UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
BACHARELADO EM SISTEMAS DE INFORMAÇÕES

JÚLIO CÉSAR DE CARVALHO BARROS

TRANSFERÊNCIA DE APRENDIZAGEM EM CNN:
Um estudo comparativo aplicado ao diagnóstico de Glaucoma

RECIFE
2023

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
BACHARELADO EM SISTEMAS DE INFORMAÇÕES

JÚLIO CÉSAR DE CARVALHO BARROS

TRANSFERÊNCIA DE APRENDIZAGEM EM CNN:
Um estudo comparativo aplicado ao diagnóstico de Glaucoma

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Pernambuco, como requisito parcial para a obtenção de título de Bacharel em Sistemas de Informações.

Área: Aprendizagem de Máquina

Orientador: Prof. Dr. Fernando Maciano de Paula Neto

RECIFE
2023

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Barros, Júlio César de Carvalho.
Transferência de Aprendizagem em CNN: Um estudo comparativo aplicado ao diagnóstico de Glaucoma / Júlio César de Carvalho Barros. - Recife, 2023.
54p : il., tab.

Orientador(a): Fernando Maciano de Paula Neto
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Informática, Sistemas de Informação - Bacharelado, 2023.
Inclui referências, apêndices.

1. Redes Neurais Convolucionais. 2. Transferência de Aprendizagem. 3. Classificação de Imagem. 4. Diagnóstico de Glaucoma. 5. Detecção por Oftalmoscopia. I. Paula Neto, Fernando Maciano de. (Orientação). II. Título.

000 CDD (22.ed.)

JÚLIO CÉSAR DE CARVALHO BARROS

TRANSFERÊNCIA DE APRENDIZAGEM EM CNN:
Um estudo comparativo aplicado ao diagnóstico de Glaucoma

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Pernambuco, como requisito parcial para a obtenção de título de Bacharel em Sistemas de Informações.

Aprovado em: 25 / 04 / 2023 .

BANCA EXAMINADORA

_____

Prof. Dr. Fernando Maciano de Paula Neto (Orientador)
Universidade Federal de Pernambuco

_____

Prof. PhD. Tsang Ing Ren (Examinador Interno)
Universidade Federal de Pernambuco

AGRADECIMENTOS

Gostaria de agradecer e dedicar esta dissertação às seguintes pessoas:

Meus pais e familiares, em especial a memória de Eurides Barbosa de Barros, por apoiarem, mesmo distantes, como podiam, dentro da desfavorável realidade enfrentada por muitos sertanejos.

A todos os amigos com quem convivi na Casa do Estudante de Pernambuco que estiveram comigo nos momentos mais difíceis. Amigos que partilham da mesma jornada de migração do interior em busca de condições melhores na capital do estado.

Aos professores do CIn, pois foram fundamentais na construção da minha formação acadêmica e profissional, sem esquecer dos demais professores que estiveram presentes desde o início.

Ao orientador pela paciência e compreensão, bem como, por fomentar a realização deste trabalho.

Por fim, mas não menos importante, gostaria de agradecer a mim mesmo, por fazer todo esse árduo trabalho, por acreditar e nunca desistir.

“Só quem suporta o processo, vive o propósito!”

Wladimir Moreira Dias

RESUMO

O glaucoma é a segunda causa mais comum de cegueira no mundo. Estima-se que já tenha afetado a visão de cerca de 80 milhões de pessoas e com projeção de crescimento para os próximos anos. Por se tratar de uma doença assintomática nas fases iniciais e incurável, apenas retardada através de tratamento, a detecção precoce é imprescindível. Nesse sentido, projetar sistemas automatizados eficazes para classificação da doença se faz relevante, tendo em vista que padrões de desconformidades podem ser observados através de exame de imagem, como o aumento da razão copo-disco, hemorragia e palidez. Este trabalho se propôs estudar a transferência de aprendizagem em Redes Neurais Convolucionais avaliando o desempenho de cinco arquiteturas de última geração (MobileNet V3, EfficientNet V2, RegNet, ConvNeXt e ResNet-RS) na classificação de diagnósticos de glaucoma. Os resultados obtidos nos extensivos experimentos realizados, utilizando o maior conjunto de dados público ACRIMA, apresentaram área sob a curva característica de operação do receptor na detecção de glaucoma de 0,9994 com intervalo de confiança de 95% entre 99,93% e 99,95%. Identificou-se superioridade dos modelos ResNet-RS em relação aos demais.

Palavras-chaves: redes neurais convolucionais; transferência de aprendizagem; classificação de imagem; diagnóstico de glaucoma; detecção por oftalmoscopia;

ABSTRACT

Glaucoma is the second most common cause of blindness in the world. It is estimated that it has already affected the vision of around 80 million people and is projected to grow in the coming years. Because it is an asymptomatic disease in the early stages and incurable, only delayed through treatment, early detection is essential. In this sense, designing effective automated systems for classifying the disease is relevant, given that patterns of non-compliance can be observed through imaging, such as increased cup-to-disk ratio, hemorrhage and pallor. This work aimed to study the transfer learning in Convolutional Neural Network evaluating the performance of five state-of-the-art architectures (MobileNet V3, EfficientNet V2, RegNet, ConvNeXt and ResNet-RS) in the classification of glaucoma diagnosis. The results obtained in the extensive experiments carried out, using the largest public dataset (ACRIMA), showed an area under the characteristic curve of operation of the receiver in the detection of glaucoma of 0.9994 with a confidence interval of 95% between 99.93% and 99.95%. The superiority of the ResNet-RS models was identified in relation to the others.

Keywords: convolutional neural networks; learning transfer; image classification; diagnosis of glaucoma; ophthalmoscopy detection;

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

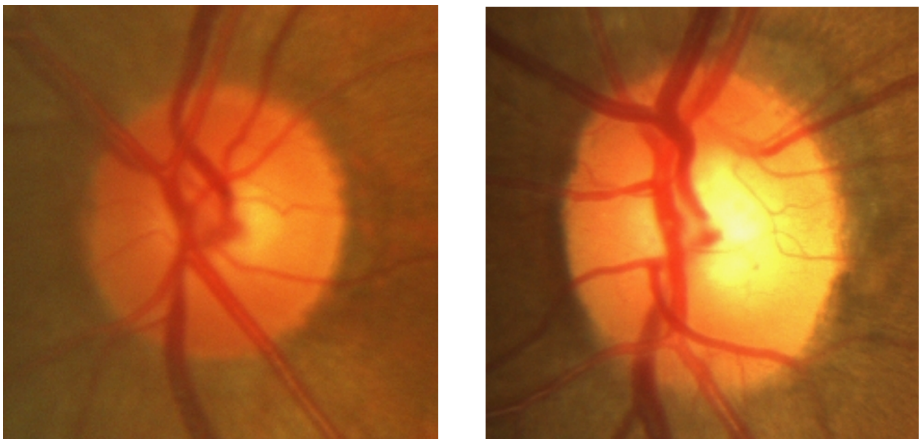
1. INTRODUÇÃO

Glaucoma é um grupo de neuropatias ópticas progressivas associadas a defeitos de campo visual característicos e alterações estruturais na cabeça do nervo óptico (LUSTHAUS, 2019). É a segunda causa mais comum de cegueira no mundo, atrás apenas da catarata, e estima-se que já tenha afetado a visão de cerca de 80 milhões de pessoas, com projeção de crescimento para 111 milhões até 2040, e que 4 milhões estejam cegas devido ao glaucoma, aponta o relatório sobre a visão da Organização Mundial de Saúde (OMS, 2019).

A detecção precoce do glaucoma é imprescindível uma vez que se trata de uma doença assintomática nas fases iniciais e não possa ser curada, apenas retardada através de tratamento. O exame de fundo de olho é uma das principais e populares modalidades para diagnosticar o glaucoma. Isso se deve por se tratar de uma abordagem não invasiva, logo adequada para triagem em larga escala, a fim de que apenas casos suspeitos passem por exames complementares e sejam acompanhados por especialistas (CHEN, X., 2015).

Estudos mostram que anormalidades na região do disco óptico, como aumento da razão copo-disco, hemorragia e palidez, fornecem evidências suficientes para a presença de glaucoma (NATARAJAN, 2021). Nesse sentido, projetar sistemas automatizados eficazes na detecção da doença se faz relevante para obtenção de uma menor taxa de erro no diagnóstico (NAWAZ, 2022).

Figura 1: Diferença visual entre um nervo óptico normal e com glaucoma



Fonte: Adaptada de DIAZ-PINTO (2019).

2. REFERENCIAL TEÓRICO

2.1. Aprendizado de máquina

O termo “Aprendizado de Máquina” tem ganhado bastante relevância na última década, embora não seja algo novo. A priori, definida como sendo o campo de estudo que possibilita aos computadores a habilidade de aprender sem explicitamente programá-los (SAMUEL, 1959). E a posteriori, compreendida pela alegação de que um programa de computador aprende pela experiência E em relação a algum tipo de tarefa T e alguma medida de desempenho P se o seu desempenho em T, conforme medido por P, melhora com a experiência E (MITCHELL, 1997).

Outra maneira de compreender o aprendizado de máquina é através da classificação da aprendizagem.

2.1.1. Origem da aprendizagem

No aprendizado supervisionado, o conjunto de dados de treinamento fornecido ao algoritmo inclui as soluções desejadas, chamadas de rótulos. O algoritmo se adapta aos dados fornecidos no intuito de realizar classificações ou previsões de conjunto de dados não rotulados (Data Science Academy, 2022). Diferentemente do aprendizado não-supervisionado, onde o conjunto de dados de treinamento fornecido ao algoritmo não inclui as soluções desejadas, os dados são analisados na busca de identificar agrupamentos ou correlações entre si. Bem como, diferente da aprendizagem semi-supervisionada que é uma abordagem híbrida entre as duas anteriores, na qual o algoritmo lida com dados de treinamento parcialmente rotulados. Outrossim, do aprendizado por reforço, onde o algoritmo - chamado de agente nesse contexto - obtém os dados através da interação com o ambiente, selecionando e executando ações, buscando maximizar recompensas e minimizar penalidades. Tal abordagem serve para identificar a melhor estratégia, chamada de política, a qual define que ação o agente deve escolher quando está em determinada situação (GÉRON, 2019, p. 13).

2.1.2. Frequência de aprendizagem

No aprendizado em lote ou por ciclo, o algoritmo, após treinado, é colocado em produção aplicando o que foi aprendido sem aprender com os novos dados que são submetidos. Para ocorrer um novo aprendizado, é necessário submeter o algoritmo ao treinamento novamente do zero com todos os dados anteriores e novos. Enquanto no aprendizado incremental, o algoritmo é treinado progressivamente, sendo fornecido os dados de forma sequencial, individual ou em pequenos lotes. Não sendo necessário submeter ao treinamento do zero ou armazenar os dados históricos.

2.1.3. Funcionamento da aprendizagem

No aprendizado baseado em instância, o algoritmo memoriza os dados de treinamento e os generaliza em novos casos, através de uma medida de similaridade, a fim de compará-los a outros exemplos aprendidos. No aprendizado baseado em modelo, o algoritmo constrói um modelo - função matemática - cujos parâmetros são ajustados à tendência dos dados de treinamento. Após treinado, o modelo pode fazer predições em novos dados.

2.1.4. Desafios

Desafios recorrentes na utilização de aprendizado de máquina são a quantidade insuficiente de dados de treinamento, dados de treinamento não representativos, dados de baixa qualidade, características irrelevantes, subajuste e sobreajuste dos dados de treinamento. Subajuste (*underfitting*) se refere a incapacidade do modelo de apreender as relações mais importante do conjunto de dados de um problema. Enquanto, sobreajuste (*overfitting*) diz respeito à ineficácia de um modelo prever novos resultados, distinto daqueles utilizados no treinamento.

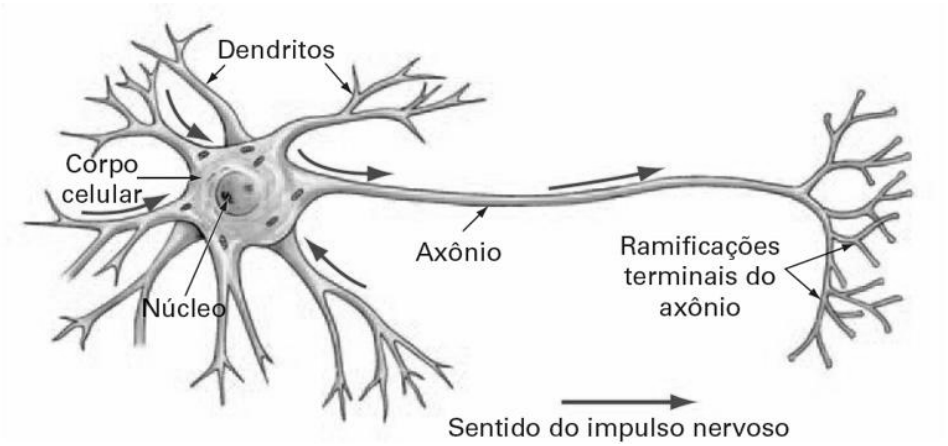
2.2. Redes Neurais Artificiais

Os algoritmos de Redes Neurais Artificiais (RNAs) são modelos de aprendizado de máquina inspirados nas redes neurais cerebrais biológicas capazes de resolver diversos problemas grandes e extremamente complexos por serem versáteis, poderosas e escaláveis.

2.2.1. Neurônios Biológicos

O neurônio biológico trata-se de uma célula encontrada principalmente no cérebro de animais, constituída por um *corpo celular* que contém a maioria dos elementos constituintes complexos da célula, e muitos prolongamentos ramificados chamados de *dendritos*, além de uma extensão bem longa chamada de *axônio*. O axônio divide-se em ramificações menores, chamadas de *telodendros*, nos quais em suas extremidades existem terminais sinápticos que estão conectados aos *dendritos* ou corpos celulares de outros neurônios (GÉRON, 2019, p. 216).

Figura 1: Representação Simplificada do Neurônio Biológico



Fonte: Data Science Academy, 2022.

Os neurônios biológicos produzem pequenos impulsos elétricos que percorrem os *axônios* e fazem as sinapses emitirem sinais químicos chamados de *neurotransmissores*. Quando um neurônio recebe uma quantidade suficiente desses estímulos em um curto intervalo, ele dispara seus próprios impulsos elétricos (Data Science Academy, 2022).

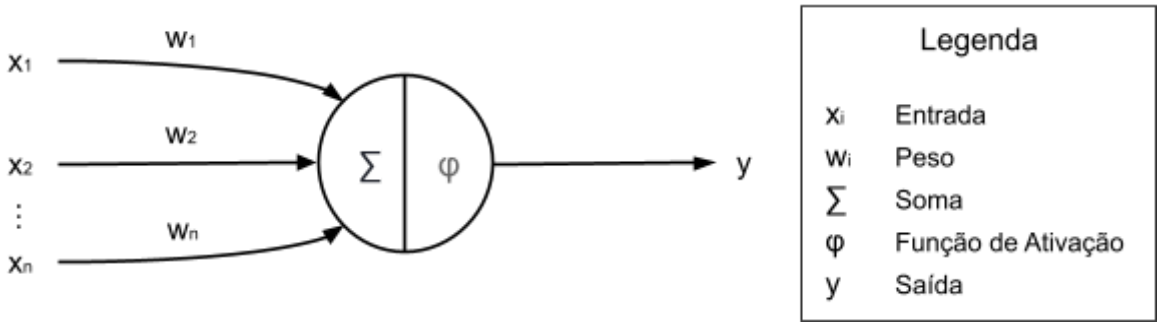
2.2.2. Neurônios Artificiais

Em 1943, McCulloch e Pitts propuseram um modelo simplificado de um neurônio biológico, simulando as características de adaptação e representação de conhecimentos baseadas em conexões. No artigo, demonstraram que com um modelo simples é possível calcular qualquer lógica proposicional (MCCULLOCH; PITTS, 1943 *apud* Data Science Academy, 2022).

2.2.3. Perceptron

Inventado por Frank Rosenblatt (1957, *apud* GÉRON, 2019, p. 218), o *perceptron* é uma das arquiteturas mais simples dos RNAs, na qual o neurônio artificial é chamado de Unidade Lógica de Limiar (TLU). As entradas e saídas são valores numéricos e cada conexão de entrada está associada a um peso. A TLU realiza a soma ponderada de suas entradas, aplica uma função de ativação a essa soma e transmite o resultado como saída.

Figura 2: Equação de saída de um Perceptron



$$y = \varphi(\sum_{i=1}^{n} x_i w_i)$$

Fonte: Adaptado de Data Science Academy (2022).

O algoritmo de treinamento de um perceptron é inspirado na Lei de Hebb (1949 *apud* GÉRON, 2019, p. 220). Essa lei pode ser sintetizada como: “Células que acionam juntas, se conectam juntas”, em outras palavras, o peso da conexão entre dois neurônios tende a aumentar quando eles ativam simultaneamente. Inspirado nisso, o algoritmo de treinamento reforça as conexões entre os neurônios de entrada atualizando o peso w_i, buscando reduzir o erro entre o valor esperado y^{real} e o valor

obtido y^{obtido} de uma entrada x_i a uma taxa de aprendizado n, conforme a equação da **Figura 3**.

Figura 3: Equação de aprendizagem do perceptron

$$w_i^{novo} = w_i^{atual} + x_i(y^{real} - y^{obtido})n$$

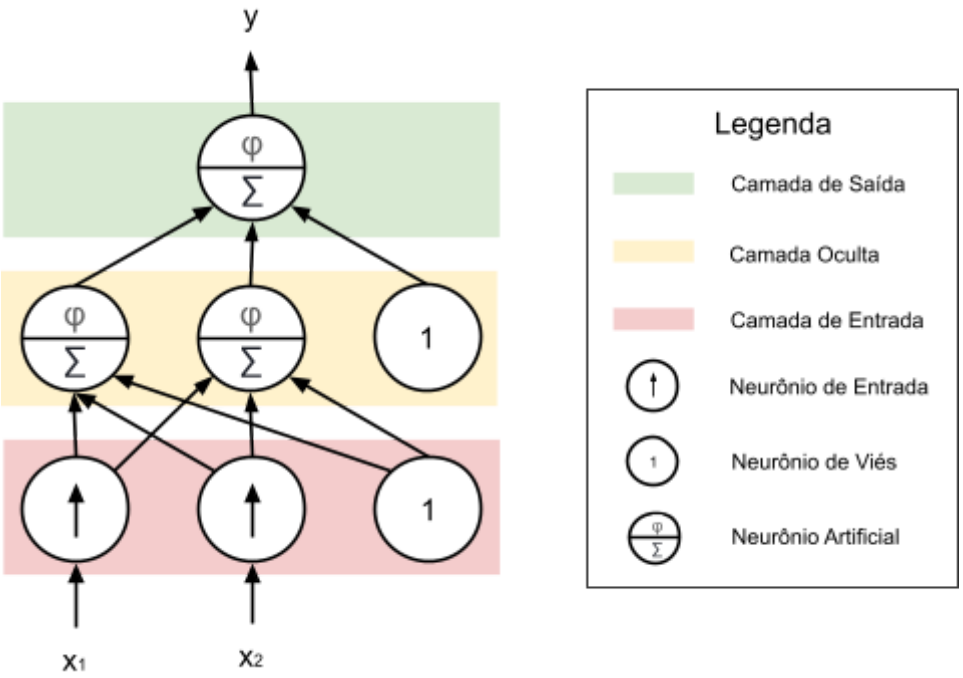
Fonte: Adaptado de GÉRON (2019)

Os perceptrons não conseguem aprender padrões complexos, como os classificadores de regressão logística, uma vez que a fronteira de decisão de cada neurônio de saída é linear. Entretanto, em 1957, Rosenblatt demonstrou com o *teorema da convergência do perceptron* que se as instâncias de treinamento forem linearmente separáveis, o algoritmo converge para uma solução. Além disso, uma série de deficiências graves das perceptrons foram ressaltadas por Marvin Minsky e Seymour Papert, em 1969, na monografia *Perceptrons*. Em especial, a incapacidade de resolver alguns problemas corriqueiros, como por exemplo, o problema de classificação *Exclusive OR* (XOR). Tais limitações só puderam ser contornadas utilizando múltiplas camadas de perceptrons (ROSENBLATT, 1957; MINSKY, PAPERT, 1969 apud GÉRON, 2019, p. 220-221).

2.2.4. Perceptron Multicamadas

Perceptron Multicamadas (MLP) é uma arquitetura complexa de RNA, composta por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída (Data Science Academy, 2022). Cada camada pode possuir uma ou mais TLUs, e podem incluir neurônios de viés com exceção da camada de saída, ilustrado na **Figura 4**. Quando uma RNA contém diversas camadas ocultas é chamada de Rede Neural Profunda (RNP).

Figura 4: Rede Neural Multicamadas



Fonte: Adaptado de GÉRON (2019).

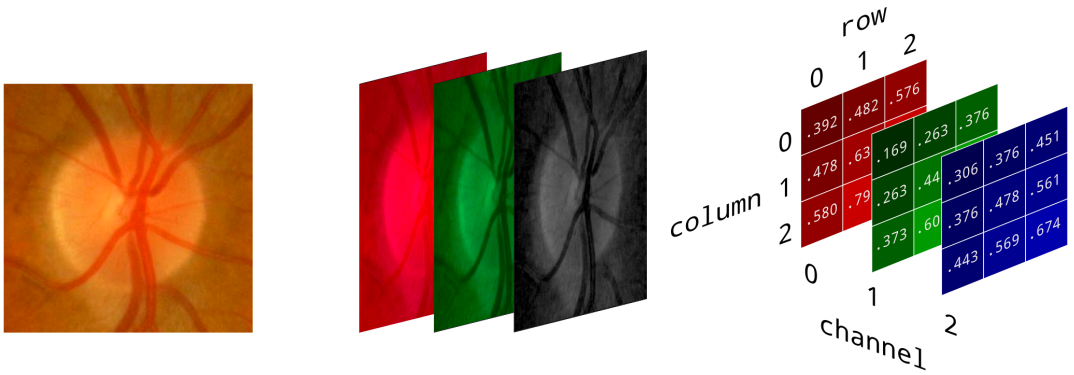
Com o aumento da complexidade da arquitetura e número de parâmetros, treinar MLPs com sucesso se tornou um grande desafio. Entretanto, David Rumelhart et. al (1986, *apud* GÉRON, 2019, p. 222) publicou um artigo com o *algoritmo de treinamento por retropropagação* de erros resolvendo o problema. Esse algoritmo consiste em duas fases: a fase de propagação, na qual as entradas são transmitidas por toda a rede e as previsões de saída são geradas; e a fase de retro-propagação, em que é calculada o gradiente da função de perda na camada de previsão e recursivamente é atualizado os pesos da rede através da regra da cadeia (Data Science Academy, 2022).

2.3. Processamento Digital de Imagens

2.3.1. Representação

Imagens são representadas digitalmente como uma matriz bidimensional (altura e largura) de pontos de cor conhecidos como *pixels*, cada *pixel* é representado como: um numérico indicando a intensidade da cor preta, para imagens em escala de cinza; ou um vetor com os valores numéricos da intensidade das cores vermelho, verde e azul - chamados de canais - para imagens coloridas.

Figura 5: Representação digital de uma imagem



Fonte: Adaptada de DIAZ-PINTO (2019).

2.3.2. Córtex Visual

A percepção de uma imagem apesar de ser uma atividade aparentemente simples para os seres humanos não é tão simples de ser explicada, muito menos reproduzida por um computador. O fato é que a percepção ocorre em grande parte fora do domínio de nossa consciência, dentro de módulos sensoriais visuais do cérebro. Uma vez que a informação sensorial chega à nossa consciência já dotada com características de alto nível. Segundo Hubel e Wiesel (1968, *apud* GÉRON, 2019, p. 345), os neurônios do córtex visual têm um pequeno campo receptivo local, ou seja, reagem apenas a estímulos visuais localizados em uma região limitada do campo visual. Os campos receptivos de diferentes neurônios podem se sobrepor e juntos, revestir todo o campo visual. Além disso, demonstraram que diferentes neurônios, ao analisar o mesmo campo receptivo, reagem de maneira distinta a depender da orientação das linhas que compõem o campo receptivo, pois cada neurônio reage somente a linhas com específicas orientações. Identificaram também que alguns neurônios possuem campos receptivos maiores e reagem a padrões

mais complexos, que são combinações dos padrões de camadas de nível anterior. Tais observações culminaram na ideia de que os neurônios de camadas mais profundas tomam como base as saídas dos neurônios de camadas vizinhas menos profundas.

2.4. Redes Neurais Convolucionais

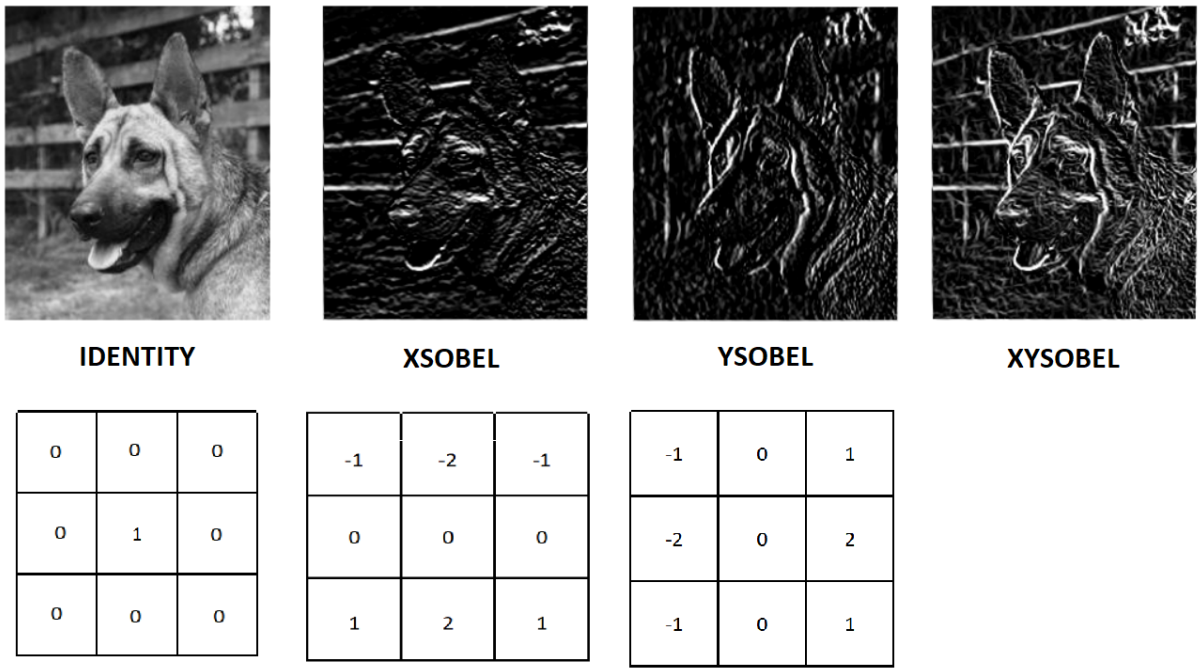
Esses estudos acerca do córtex visual serviram como inspiração para uma das redes neurais precursoras em reconhecimento de padrões de imagens, chamada de *neocognitron*, apresentada por Fukushima (1980, apud GÉRON, p. 346), que gradativamente evoluiu para o que hoje chamamos de Redes Neurais Convolucionais (em inglês, Convolutional Neural Networks ou CNN), ganhando notoriedade através do artigo publicado por LeCun (1998 *apud* NIELSEN, 2023). Tal rede é constituída por dois novos tipos de elementos: as Camadas Convolucionais e Camadas de Subamostragem.

2.4.1. Camada de Convolução

A camada de convolução é responsável por extrair características da imagem de entrada, passo a passo, através da estratégia de janela deslizante, multiplicando os pesos em cada filtro (*kernel*) pelos valores de pixels e combinando a soma para criar uma nova imagem passada para a próxima camada.

Um filtro de convolução trata-se de uma pequena matriz de pesos, cujo tamanho representa o campo receptivo, como 3x3, que quando percorre a um dado passo (*stride*) - por exemplo a cada um pixel da direita a esquerda de cima para baixo - multiplicando uma matriz de pixels 24x24 gera como resultado outra matriz de pixels, de mesmo tamanho quando preenchida com zero (*zero-padding*) bordas imaginárias ou de tamanho menor (22x22), que a depender dos pesos pode desfocar elementos, destacar relevo, detectar bordas e reconhecer características (NIELSEN, 2023). A **Figura 6** demonstra o resultado da aplicação de alguns filtros.

Figura 6: Demonstração de Filtros Convolucionais



Fonte: Própria autoria

2.4.2. Camada de Subamostragem

A camada de subamostragem desempenha o papel de redução da amostragem, ou seja, reduzir o número de dados para economizar recursos computacionais. Consiste em condensar os mapas de características obtidos pela camada anterior, em mapas de características menores, digamos que uma região de 2x2 possa ser representado como um único valor, aplicando alguma técnica como representar o agrupamento através do valor máximo (*max polling*), valor mínimo (*min polling*) ou valor médio (*average polling*). Desta forma, um mapa de características de tamanho 24x24, por exemplo, pode ser representado com menos informações e menor exatidão posicional por um mapa de características de tamanho 12x12 (NIELSEN, 2023).

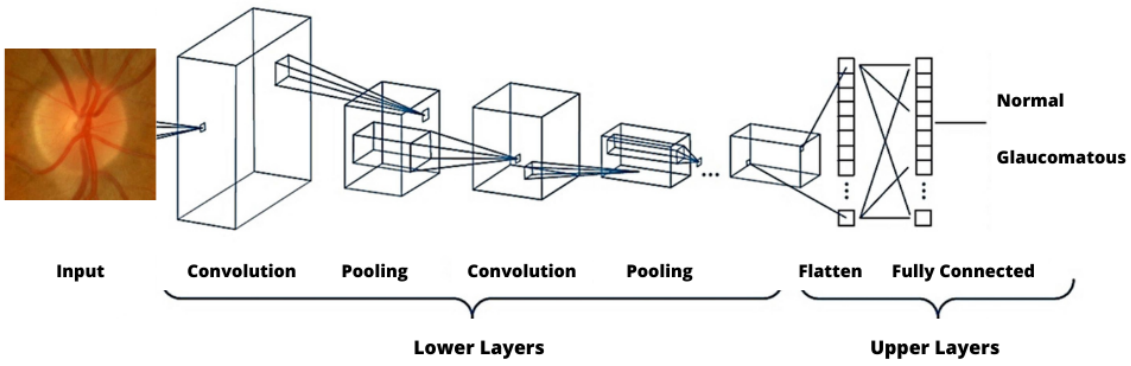
2.4.3. Camada Totalmente Conectada

Por fim, a matriz de pixels da imagem resultante das camadas anteriores passa por uma camada plana para ser convertida em uma matriz unidimensional, para então ser submetida a uma camada densa de saída que usa essa matriz como entrada para produzir o rótulo previsto aplicando a combinação linear e a função de ativação não linear (Data Science Academy, 2022).

2.4.4. Arquitetura de uma Rede Neural Convolucional

As camadas de convolução e subamostragem podem se repetir, tornando a rede ainda mais profunda, atribuindo diferentes tamanhos de mapa de características e serem combinadas com outras técnicas, ilustrado na **Figura 7**. Competições, como a *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*, têm motivado pesquisadores a criar, aperfeiçoar, testar e disponibilizar diversas abordagens de arquiteturas buscando obter melhor desempenho. Tudo isso, utilizando conjuntos de dados volumosos, por exemplo o *ImageNet* que contém mais de um milhão de imagens de mil classes diferentes de objetos.

Figura 7: Arquitetura de uma Rede Neural Convolucional



Fonte: Própria autoria

2.4.5. Transferência de Aprendizagem

Transferência de Aprendizagem é uma técnica utilizada para melhorar a eficiência do treinamento de classificadores de imagens, sem a necessidade de um conjunto de dados de treinamento expressivo, reutilizando os pesos aprendidos de um modelo pré-treinado em um banco de dados de imagens volumoso para realizar uma certa tarefa, por exemplo: distinguir cães e gatos, para outra, como: diferenciar exames de fundo de olho com ou sem glaucoma.

Isso é possível, pois as características apreendidas pelas camadas inferiores (camadas de convolução e camadas de subamostragem), como: linhas e arestas, costumam estar relacionadas. Embora seja questionável a semelhança entre a aparência de imagens não-médicas e imagens médicas, diversos estudos evidenciam as vantagens da utilização (CARNEIRO, 2015; CHEN, H., 2015; TAJBAKHSH, 2016, *apud* DIAZ-PINTO, 2019).

A técnica consiste na importação de um modelo com pesos pré-definidos, sem as camadas superiores (camada de média global e a camada densa de saída) que são responsáveis pelo aprendizado da classificação no contexto de treinamento inicial, na qual são adicionadas camadas superiores próprias para classificação em um novo contexto (GÉRON, 2019, p. 371).

Segundo o TensorFlow (2023), o treinamento utilizando a técnica é realizado em duas etapas: Na primeira, chamada de extração de recursos, as camadas inferiores são congeladas, ou seja, definidas como não treináveis. Nesta fase, apenas as camadas superiores adicionadas são treinadas para se adequar aos pesos pré-existentes; Já na segunda etapa de treinamento, chamada de ajuste fino, as camadas inferiores são descongeladas, ou seja, definidas como treináveis, além de ser utilizada uma taxa de treinamento menor que na etapa anterior. Tal fase consiste em ajustar os pesos das camadas superiores de mapas de características genéricas para mapas de características associados especificamente ao conjunto de dados de treinamento fornecido.

2.5. Técnicas de Regularização

Quanto mais parâmetros possuir um modelo, mais versátil será para se acomodar a um conjunto de dados complexos. Entretanto, essa flexibilidade também significa que a rede está propensa a se sobreajustar ao conjunto de treinamento, sendo necessárias regularizações (GÉRON, 2019, p. 280).

2.5.1. Dropout

Dropout é uma técnica para omitir a participação de um subconjunto aleatório de neurônios em uma ou mais camadas, com exceção da camada de saída, durante o treinamento a fim de reduzir a dependência exagerada de alguns neurônios, fazendo com que a rede neural se torne mais robusta e generalize melhor (KRIZHEVSKY, 2017).

2.5.2. Data Augmentation

Data Augmentation é uma técnica para expandir artificialmente o tamanho do conjunto de dados de treinamento, gerando diversas variantes realistas de cada instância de treinamento. Trata-se de uma técnica de regularização que busca reduzir o sobreajuste adicionando novas instâncias resultantes do deslocamento, rotação, redimensionamento, giro horizontal, giro vertical ou alteração das condições de iluminação das imagens existentes, de maneira aleatória, no conjunto de dados de treinamento (GÉRON, 2019, p. 360).

3. TRABALHOS RELACIONADOS

Esta seção busca apresentar a revisão bibliográfica acerca de trabalhos sobre sistemas de detecção de glaucoma baseados em aprendizado profundo.

Chen X. et al. (2015) implementou e treinou do zero um modelo de arquitetura de CNN, constituído por seis camadas, sendo quatro camadas convolucionais e duas totalmente conectadas. Esse modelo foi treinado em dois conjuntos de dados privados: ORIGA que contém 650 imagens (sendo 168 com glaucoma e 482 normais) e SCES que contém 1676 (sendo 46 com glaucoma e 1630 normais), e foram obtidos os seguintes valores de área sob a curva (AUC) 0,831 e 0,887. As principais limitações observadas neste estudo foram o desbalanceamento das amostras de cada uma das classes, bem como, a dificuldade de reproduzir o experimento uma vez que se trata de um conjunto de dados privado.

Diaz-Pinto et al. (2019), comparou diferentes modelos pré-treinados - no conjunto de dados *ImageNet* - de cinco arquiteturas de CNN (VGG16, VGG19, InceptionV3, ResNet50 e Xception) para classificação de glaucoma. Esses modelos foram treinados e avaliados em cinco conjuntos de dados públicos: ACRIMA que contém 705 imagens (sendo 396 com glaucoma e 309 normais), HRF que contém 45 imagens (sendo 27 com glaucoma e 18 normais), Drishiti-GS1 que contém 101 imagens (sendo 70 com glaucoma e 31 normais), RIM-ONE que contém 455 imagens (sendo 194 com glaucoma e 261 normais), sjchoi86-HRF que contém 401 (sendo 101 com glaucoma e 300 normais), e foram obtidos respectivamente os seguintes valores de AUC 0,7678, 0,8041, 0,8575 e 0,7739 pelo modelo Xception que apresentou melhor custo benefício entre AUC e número de parâmetros.

Natarajan et al. (2021), apresentou uma estrutura robusta para segmentação da região de interesse (ROI) e classificação do glaucoma, utilizando para classificação um modelo, baseado na arquitetura SqueezeNet, treinado nos conjuntos de dados RIGA e RIM-ONE-V2 e testadas nos conjuntos de dados ACRIMA, Drishti-GS1 e RIM-ONE-V1, obtendo os respectivos valores de AUC, 0,9999, 0,9990 e 1,0000.

Nawaz et al. (2022), realizou um trabalho robusto composto em dois estágios utilizando EfficientDet-D0 para segmentação e EfficientNet-B0 para classificação. Esta solução foi treinada no conjunto de dados ORIGA, testado nos conjuntos de dados HRF e RIM-ONE-DL, obteve valores de AUC de 0.979.

Noury et al. (2022), propôs uma sofisticada solução em aprendizado profundo para detecção de glaucoma em tomografias 3D. Na qual, validou a arquitetura de CNN 3D em três conjuntos de dados etnicamente diferentes, obtendo 0,91, 0,80, 0,94 e 0,87 de AUC, respectivamente, em base de dados de Stanford, Hong Kong, India e Nepal.

Este trabalho apresenta uma estrutura simplificada de detecção de glaucoma sendo realizado apenas o estágio de classificação, foram utilizadas como entradas para os modelos pré-treinados imagens 2D previamente recortadas na ROI, conforme descrito nas seções a seguir.

4. OBJETIVO

4.1. Objetivo Geral

Avaliar o desempenho de modelos pré-treinados, construídos a partir das mais recentes e públicas arquiteturas de CNN, re-treinando-as para classificação de diagnósticos de glaucoma.

4.2. Objetivo Específico

- Aprofundar o conhecimento científico do estudante no tema por meio de revisão da fundamentação teórica;
- Transferir aprendizagem de modelos pré-treinados para classificação de imagens em um novo contexto;
- Avaliar as métricas de desempenho obtidas dos modelos de cada uma das arquiteturas.

5. METODOLOGIA

Esta seção busca delimitar os procedimentos metodológicos de execução deste trabalho.

5.1. Revisão Bibliográfica

A revisão bibliográfica foi realizada a partir de livros e publicações científicas existentes nos principais indexadores de repositórios de periódicos científicos, como SciELO ou Google Scholar. Utilizando as palavras-chaves da referida temática, por exemplo: "*convolutional neural network*", "*glaucoma classification*", "*transfer learning*", "*deep learning*" e "*artificial intelligence",* priorizando publicações mais relevantes e recentes.

5.2. Seleção das Arquiteturas

A seleção das arquiteturas das CNN se deu através dos seguintes critérios: disponibilidade da implementação no framework Keras; adequação ao problema de classificação de imagens; data de publicação do artigo de referência até 4 anos anteriores da data de realização deste trabalho.

Nesse sentido, as arquiteturas selecionadas foram: ConvNeXt, ResNet-RS, RegNet, MobileNetV3 e EfficientNetV2, conforme pode ser observado no **Apêndice A**.

5.3. Aquisição de conjunto de dados

A escolha do conjunto de dados a ser utilizado para o treinamento e validação dos modelos foi realizada sobre os seguintes critérios: disponibilidade em repositórios de domínio público utilizado em trabalhos relacionados; maior número de amostras possíveis; adequação a proposta de classificação, ou seja, previamente segmentadas e rotuladas por especialistas.

O conjunto de dados que melhor satisfaz os critérios acima, segundo a pesquisa realizada, foi o ACRIMA, composto de 705 imagens, sendo 309 normal e 396 com glaucoma, mais detalhes da seleção no **Apêndice B**.

Tal conjunto de dados foi elaborado pelo projeto TIN2013-46751-R, fundado pelo Ministério de Economia e Competitividade da Espanha com o objetivo de

desenvolver algoritmos automáticos para avaliação de doenças da retina. No qual, todas as imagens foram obtidas em um ângulo de visão de 35º, utilizando a câmera retinal *Topcon TRC* e o sistema de captura *IMAGEnet*. A partir de pacientes - selecionados por especialistas - que consentiram previamente seguindo os padrões éticos da Declaração de Helsinque 1964 (DIAZ-PINTO, 2019).

5.4. Treinamento de modelos

O treinamento supervisionado ocorreu por transferência de aprendizagem, baseando-se no guia do TensorFlow (2023), os modelos foram submetidos ao mesmo conjunto de dados, algoritmos de otimização, função de perda, número de amostras e interações.

Para realização do treinamento, o conjunto de dados foi dividido em 80% para treinamento e 20% para teste, assim como, aumentado artificialmente adicionando variações de rotação aleatórias em até 40 graus e sendo submetido em lotes de 16 amostras - escolhidas ao acaso - de imagens com 224x224 pixels cada.

Na etapa de extração de características, na qual apenas a camada superior que substituiu a utilizada na classificação de origem é submetida ao treinamento, foram realizadas 10 interações, utilizando: taxa de aprendizagem de 10^{-4}, função de perda *Binary Cross Entropy* e otimização *Adaptive Moment Estimation (Adam)*.

Na etapa de ajuste fino, na qual a aprendizagem de características associadas especificamente ao domínio do problema, foram realizadas mais 10 interações, utilizando: taxa de aprendizagem de 10^{-3}, função de perda *Binary Cross Entropy* e otimização *Root Mean Squared Propagation (RMSprop)*.

5.5. Avaliação de Modelo

As métricas de avaliação foram calculadas da seguinte maneira:

Para cada um dos modelos re-treinados foi realizada a validação no conjunto de dados original - sem aumento artificial de dados - e a partir dos resultados foi obtida a Matriz de Confusão, tabela que permite a visualização de classificações corretas e incorretas, ou seja, verdadeiros-positivos (V_P) e verdadeiros-negativos (V_N), bem como os falsos-positivos (F_P) e os falsos-negativos (F_N). Tais valores são utilizados para calcular as métricas da **Tabela 1**:

Tabela 1: Descrição das métricas de avaliação

Métrica	Descrição	Fórmula
Acurácia	Proporção de acertos em razão de todos os itens classificados	$A = \frac{V_P + V_N}{V_P + V_N + F_P + F_N}$
Sensibilidade	Capacidade de encontrar todas as amostras positivas	$S = \frac{V_P}{V_P + F_N}$
Especificidade	Capacidade de encontrar todas as amostras negativas	$E = \frac{V_N}{V_N + F_P}$
Precisão	Capacidade de classificar como positivo somente as amostras relevantes	$P = \frac{V_P}{V_P + F_P}$
F-score	Média harmônica entre a precisão e sensibilidade	$F1 = 2 \cdot \frac{P \cdot S}{P + S}$

Fonte: Adaptado de HARRISON (2019)

Em seguida, foi obtida a curva característica de operação do receptor (em inglês, Receiver Operating Characteristic, ROC) que é uma ferramenta amplamente utilizada para avaliar classificadores. Tal ferramenta, mostra a taxa dos V_P (Sensibilidade) em relação a taxa de F_P (1 - Especificidade), na qual é calculada a área sob a curva (em inglês, Area Under the Curve, AUC) no intuito de obter uma métrica de valor único, entre 0 e 1, para comparar diferentes classificadores. Quanto mais próximo de 1 melhor e abaixo de 0.5 será considerado um modelo ruim (HARRISON, 2019).

5.6. Teste Estatístico de Hipótese

No intuito de auxiliar na identificação da arquitetura que possui o melhor desempenho através do método quantitativo, foi realizado um teste estatístico de hipótese, tendo em vista a natureza aleatória das variáveis de treinamento. Dessa forma, foi repetido o experimento 30 vezes para cada uma das arquiteturas, coletando os resultados obtidos, para calcular o valor médio das métricas dentro de um intervalo de confiança de 95% para apoiar a afirmação com base em evidências estatísticas de amostras de uma população (GRUS, 2016, p. 94), conforme a equação da **Figura 8**.

Figura 8: Equação do intervalo de confiança

$$P(\overline{X} - Z_{\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}} \leq \mu \leq \overline{X} + Z_{\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}}) = 1 - \alpha$$

Legenda:

- $\overline{X}$ a média amostral;
- $Z_{\frac{\alpha}{2}}$ valor associado ao intervalo de confiança;
- α nível de significância;
- $\frac{\sigma}{\sqrt{n}}$ desvio padrão amostral da média;
- μ a média populacional;

Fonte: Adaptado de WALPOLE (1993)

5.7. Ferramentas utilizadas

O equipamento utilizado para execução deste trabalho possui as seguintes especificações:

CPU: Intel Core i5-9300H @ 2.40GHz ~ 4.10GHz;
RAM: 16GB DDR4 2666 MHz;
GPU: NVIDIA GeForce GTX 1650 4GB GDDR6;
HD: 512GB SSD M.2 NVMe.

As ferramentas utilizadas para a implementação e treinamento dos modelos foram: Python, Anaconda, Jupyter Notebook, Scikit-Learn, Tensorflow e Keras. Ademais, o código fonte utilizado neste experimento está disponível no **Apêndice C**.

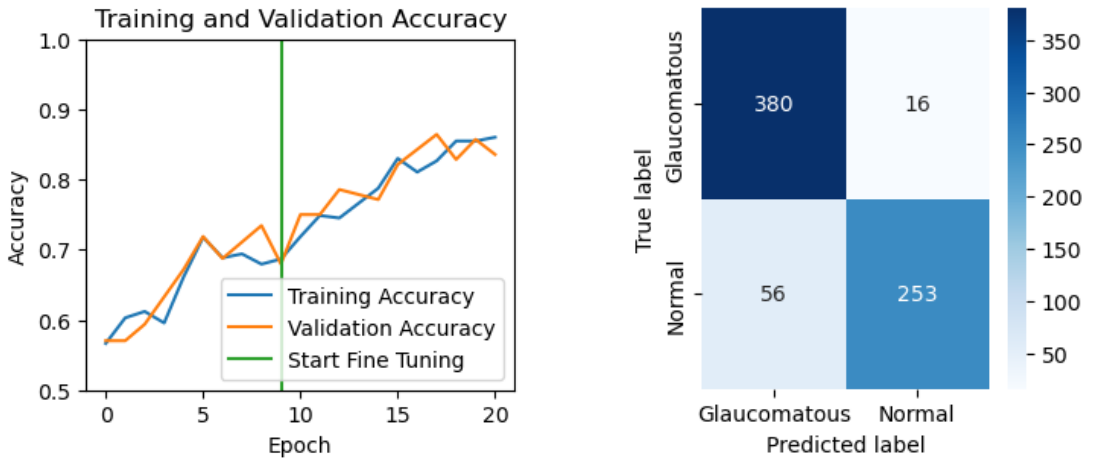
6. RESULTADOS E DISCUSSÃO

Esta seção busca apresentar e discutir os resultados obtidos.

O primeiro conjunto de modelo a ser analisado foi construído a partir da arquitetura MobileNet V3 que objetiva balancear acurácia e performance na classificação de imagens para utilização em dispositivos móveis com menor poder computacional (QIAN, 2021). Os modelos com apenas 229 camadas e 939 mil parâmetros obtiveram acurácia média de 65,22% e 84,56%, respectivamente, na primeira e segunda fase de treinamento.

Ao submeter para avaliação, um dos exemplares do modelo, das 705 amostras analisadas 633 foram previstas corretamente, sendo: 380 casos positivos de glaucoma (verdadeiros-positivos); e 253 casos normais, ou melhor, na qual não há evidências de glaucoma (verdadeiros-falsos). Apenas 72 amostras foram classificadas incorretamente, sendo: 16 falsos-negativos, imagens nas quais havia indícios para glaucoma, entretanto foram classificadas como normais; e 56 falsos-positivos, imagens normais, mas que foram classificadas como positivo para glaucoma. Os resultados podem ser visualizados na **Figura 9**.

Figura 9: Gráfico de treinamento e matriz de confusão de um modelo MobileNet V3

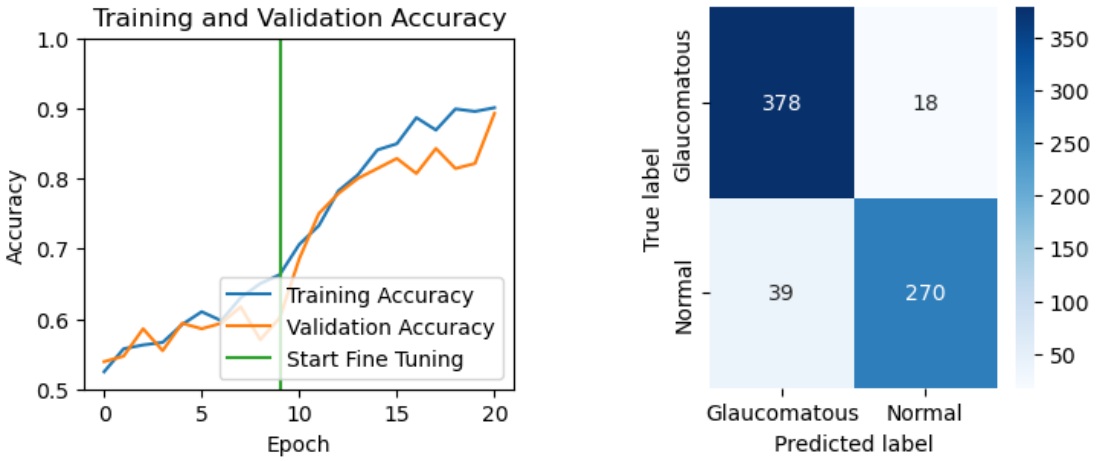


Fonte: Própria autoria

Já o segundo conjunto de modelos foi concebido através da arquitetura EfficientNet V2. Tal arquitetura tem como princípio alcançar maior velocidade de treinamento e melhor eficiência de parâmetros (TAN, 2021). Com 270 camadas e 5 milhões de parâmetros, atingiu durante o treinamento acurácia média 61,20% na primeira etapa, e 87,24% na segunda etapa.

Na avaliação, um exemplar desse conjunto, previu corretamente 648 das 705 amostras: 378 verdadeiros-positivos; e 270 verdadeiros-negativos. Apenas 57 amostras foram classificadas incorretamente, sendo: 18 falsos-negativos e 39 falsos-positivos. Os resultados podem ser visualizados na **Figura 10**.

Figura 10: Gráfico de treinamento e matriz de confusão de um modelo EfficientNet V2

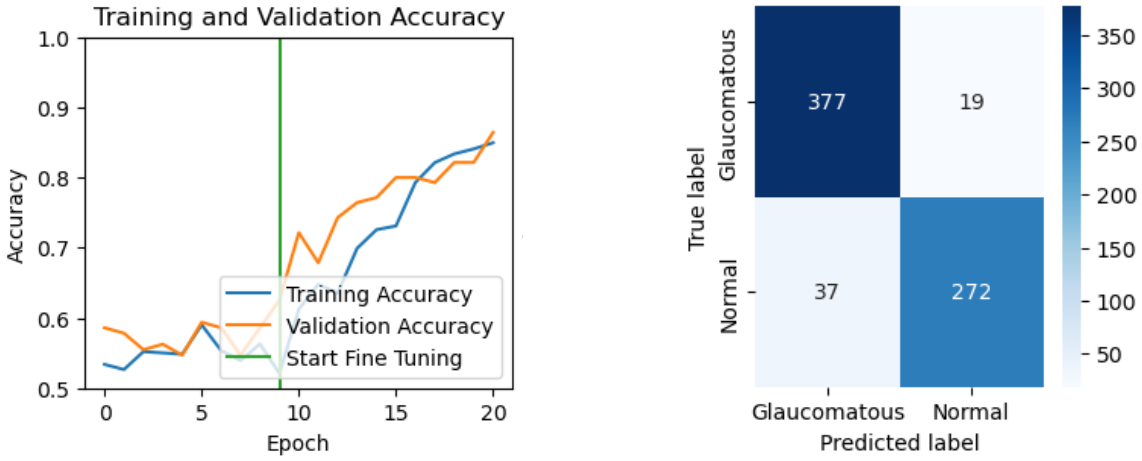


Fonte: Própria autoria

Ademais, a arquitetura RegNet que busca melhorar a escalabilidade para dotá-la de maior complexidade computacional e consequentemente poder representacional, através de estratégias de dimensionamento como aumento da largura, profundidade, resolução do modelo entre outros (DOLLÁR, 2021). A partir dela, modelos de apenas 143 camadas e 2 milhões de parâmetros, obtiveram 60,84% e 89,04% de acurácia média, respectivamente durante as etapas de extração de features e ajuste fino de treinamento.

Durante a avaliação de um exemplar deste, conforme a **Figura 11**, as previsões corretas somaram 649 amostras sendo: 377 verdadeiros-positivos e 272 verdadeiros-falsos. Já as previsões incorretas totalizaram 56 (37 falsos-positivos e 19 falsos-negativos).

Figura 11: Gráfico de treinamento e matriz de confusão de um modelo RegNet

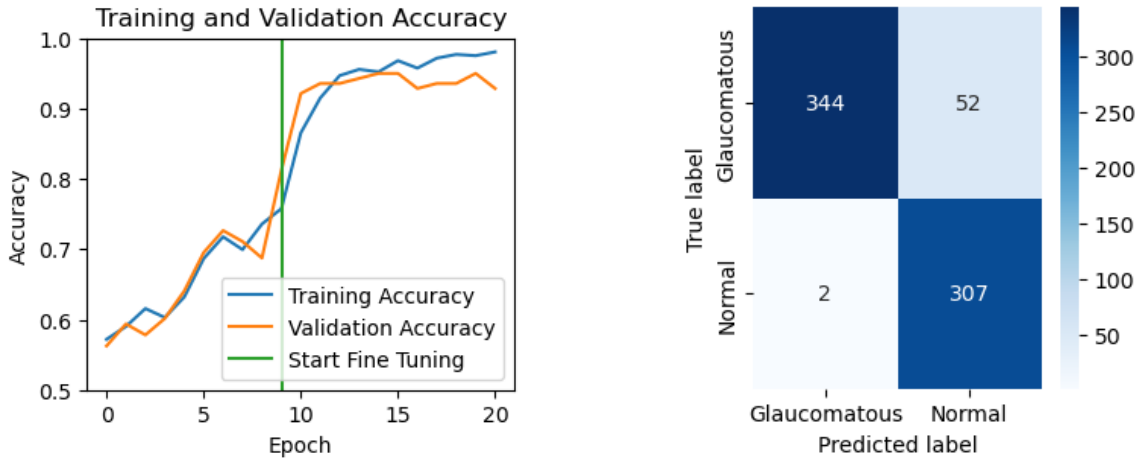


Fonte: Próprio autor

A arquitetura ConvNeXt almeja superioridade de acurácia e escalabilidade através de abordagens híbridas entre os vieses indutivos inerentes da convolução e transformers, técnica utilizada na aprendizagem de contextos em dados sequenciais, como no processamento de linguagem natural (LIU, 2022). Com 295 camadas e 87 milhões de parâmetros, obteve 75,82% de acurácia média na primeira etapa e 96,31% de acurácia média na segunda etapa de treinamento.

Na avaliação de um exemplar do modelo em questão, de acordo com a **Figura 12**, foram previstas corretamente 651 amostras (344 verdadeiros-positivos e 307 verdadeiros-negativos) das 705 do conjunto de dados. Enquanto as previsões incorretas, somente 54 (52 falsos-negativos e 2 falsos-positivos).

Figura 12: Gráfico de treinamento e matriz de confusão de um modelo ConvNeXt

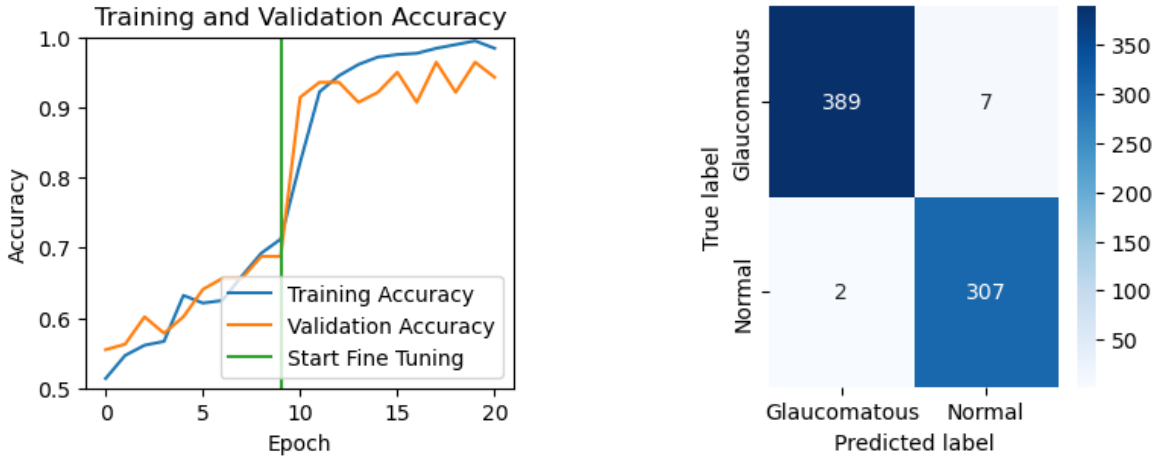


Fonte: Própria autoria

A arquitetura ResNet-RS é motivada pela obtenção de superioridade de acurácia com maior foco em estratégias de treinamento e escalabilidade ponderadas do que em mudanças arquitetônicas (BELLO, 2021). Com impressionantes 1471 camadas e 128 milhões de parâmetros obteve acurácia média 73,04% ainda na primeira fase de treinamento e 98,31% na segunda fase de treinamento.

Um exemplar desse conjunto, durante a avaliação, obteve 696 classificações corretas (389 verdadeiros-positivos e 307 verdadeiros-negativos) e apenas 9 classificações incorretas (2 falsos-positivos e 7 falsos-negativos), observáveis na **Figura 13**.

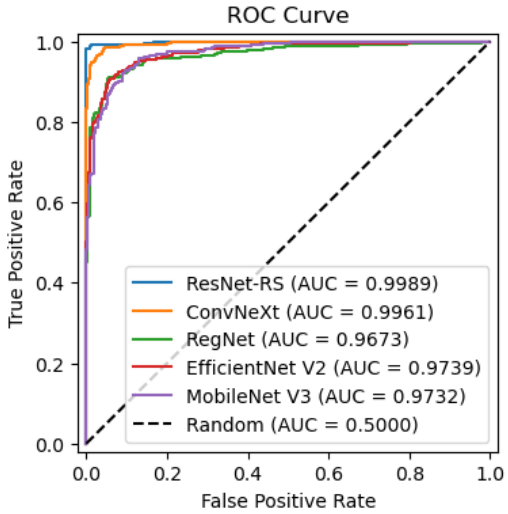
Figura 13: Gráfico de treinamento e matriz de confusão de um modelo ResNet-RS



Fonte: Própria autoria

Em geral, os resultados obtidos pelas arquiteturas nesse experimento foram bastante competitivos, sobretudo quando comparado uma única amostra de cada arquitetura, a diferença é quase imperceptível, vistas na **Figura 14**. Nesse sentido, a fim de compará-los com maior precisão foram treinados 30 exemplares de cada arquitetura e calculadas as métricas de avaliação em um intervalo de confiança de 95%. Conforme observado na **Tabela 2**, dois modelos se destacaram entre os demais, sendo eles: ConvNeXt e ResNet-RS.

Figura 14: Comparação ROC AUC entre os modelos



Fonte: Própria autoria

Tabela 2: Resultados das métricas de avaliação

Modelos	Parâmetros	Acurácia	Sensibilidade	Especificidade	Precisão	F-Score	AUC
MobileNet V3	939.697	0,8930 (0,0066)	0,9495 (0,0059)	0,8207 (0,0170)	0,8727 (0,0102)	0,9091 (0,051)	0,9641 (0,0032)
EfficientNet V2	5.920.593	0,9020 (0,0039)	0,9660 (0,0032)	0,8201 (0,0107)	0,8735 (0,0065)	0,9173 (0,0030)	0,9702 (0,0008)
RegNet	2.337.009	0,9086 (0,0047)	0,9462 (0,0095)	0,8604 (0,0153)	0,8980 (0,0094)	0,9209 (0,0039)	0,9693 (0,0018)
ConvNeXt	87.567.489	0,9564 (0,0044)	0,9337 (0,0092)	0,9854 (0,0026)	0,9881 (0,0020)	0,9599 (0,0043)	0,9956 (0,0003)
ResNet-RS	128.057.569	**0,9798 (0,0040)**	**0,9708 (0,0082)**	**0,9914 (0,0030)**	**0,9932 (0,0023)**	**0,9817 (0,0037)**	**0,9994 (0,0001)**

Fonte: Própria Autoria

7. CONCLUSÃO

Esta seção busca apresentar as considerações finais e indicar trabalhos futuros.

Neste trabalho, foi apresentado uma estrutura de avaliação para 5 diferentes arquiteturas de última geração de CNN, baseados em transferência de aprendizagem para detecção automática de glaucoma, na qual foi possível comparar o desempenho dos modelos, a partir de métricas amplamente utilizadas em trabalhos relacionados na base de dados pública ACRIMA. No experimento realizado o modelo ResNet-RS apresentou os melhores resultados em comparação com os demais, obtendo 0,9994 de AUC com intervalo de confiança de 95% de 99,93-99,95%.

Ademais, embora os resultados sejam promissores e apresentem uma melhoria significativa quando comparados aos trabalhos relacionados, é de conhecimento as limitações deste trabalho. Dada a dificuldade em generalizar para conjuntos de dados distintos dos usados para treinamento, mesmo com a utilização do maior banco de dados público disponível e aumento de dados para realizar o ajuste-fino dos modelos, tendo em vista a diversidade entre os pacientes, qualidade das imagens e diferentes critérios de avaliação utilizados para rotulação do banco de dados.

Em trabalhos futuros, planeja-se disponibilizar os modelos treinados em uma plataforma online para fins de demonstração, contemplar a etapa de segmentação da ROI de imagens, assim como, realizar estudos acerca de técnicas e processos que permitam melhor compreensão dos resultados obtidos pelas inteligências artificiais explicáveis (em inglês, *eXplainable Artificial Intelligence*, XAI). Outrossim, recomenda-se a elaboração de um banco de dados volumoso com amostras de exame de fundo de olho que melhor representem a diversidade étnica nacional.

REFERÊNCIAS

BELLO, Irwan et al. Revisiting resnets: Improved training and scaling strategies. **Advances in Neural Information Processing Systems**, v. 34, p. 22614-22627, 2021.

CHEN, Xiangyu et al. Glaucoma detection based on deep convolutional neural network. In: **2015 37th annual international conference of the IEEE engineering in medicine and biology society (EMBC)**. IEEE, 2015. p. 715-718.

Data Science Academy. **Deep Learning Book**, 2022. Disponível em: <https://www.deeplearningbook.com.br/>. Acesso em: 20 de dezembro de 2022.

DIAZ-PINTO, Andres et al. CNNs for automatic glaucoma assessment using fundus images: an extensive validation. **Biomedical engineering online**, v. 18, p. 1-19, 2019.

DOLLÁR, Piotr; SINGH, Mannat; GIRSHICK, Ross. Fast and accurate model scaling. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2021. p. 924-932.

GÉRON, Aurélien. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow**. Alta Books, 2019.

GRUS, Joel. **Data science do zero**. Alta Books, 2016.

HARRISON, Matt. **Machine Learning–Guia de Referência Rápida: Trabalhando com dados estruturados em Python**. Novatec Editora, 2019.7

LI, F. F.; JOHNSON, J.; YEUNG, S. **Convolutional neural networks: architectures, convolution/pooling layers**. Course notes, CS231n: Convolutional Neural Networks for Visual Recognition, Spring Quarter, Department of Computer Science. 2023.

LIU, Zhuang et al. A convnet for the 2020s. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2022. p. 11976-11986.

LUSTHAUS, Jed; GOLDBERG, Ivan. Current management of glaucoma. **Medical Journal of Australia**, v. 210, n. 4, p. 180-187, 2019.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. **Communications of the ACM**, v. 60, n. 6, p. 84-90, 2017.

MITCHELL, Tom M.; MITCHELL, Tom M. **Machine learning**. New York: McGraw-hill, 1997.

NATARAJAN, Deepa et al. A deep learning framework for glaucoma detection based on robust optic disc segmentation and transfer learning. **International Journal of Imaging Systems and Technology**, v. 32, n. 1, p. 230-250, 2021.

NAWAZ, Marriam et al. An efficient deep learning approach to automatic glaucoma detection using optic disc and optic cup localization. **Sensors**, v. 22, n. 2, p. 434, 2022.

NIELSEN, M. Chapter 6, Deep Learning. Neural Networks and Deep Learning. Disponível em: <http://neuralnetworksanddeeplearning.com/chap6.html>. Acesso em: 04 de abril de 2023.

NOURY, Erfan et al. Deep Learning for Glaucoma Detection and Identification of Novel Diagnostic Areas in Diverse Real-World Datasets. **Translational Vision Science & Technology**, v. 11, n. 5, p. 11-11, 2022.

OMS et. al. Relatório Mundial sobre a visão. 2019. Disponível em: <https://www.who.int/publications/i/item/9789241516570>. Acesso em 09 de novembro de 2022.

QIAN, Siying; NING, Chenran; HU, Yuepeng. MobileNetV3 for image classification. In: **2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering** (ICBAIE). IEEE, 2021. p. 490-497.

SAMUEL, Arthur L. Machine learning. **The Technology Review**, v. 62, n. 1, p. 42-45, 1959.

TAN, Mingxing; LE, Quoc. Efficientnetv2: Smaller models and faster training. In: **International conference on machine learning**. PMLR, 2021. p. 10096-10106.

TensorFlow, Guia: Transferência de aprendizado e ajuste fino. 2023. Disponível em: <https://www.tensorflow.org/tutorials/images/transfer_learning>. Acesso em: 06 de março de 2023.

WALPOLE, Ronald E. et al. Probability and statistics for engineers and scientists. **New York: Macmillan**, 1993.

APÊNDICE A

PUBLICAÇÕES DE REFERÊNCIA DAS ARQUITETURAS DE CNN

Arquitetura	Artigo	Ano
ConvNeXt	LIU, Zhuang et al. A convnet for the 2020s. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2022. p. 11976-11986.	2022
ResNet-RS	BELLO, Irwan et al. Revisiting resnets: Improved training and scaling strategies. **Advances in Neural Information Processing Systems**, v. 34, p. 22614-22627, 2021.	2021
RegNet	DOLLÁR, Piotr; SINGH, Mannat; GIRSHICK, Ross. Fast and accurate model scaling. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition**. 2021. p. 924-932.	2021
MobileNet V3	QIAN, Siying; NING, Chenran; HU, Yuepeng. MobileNetV3 for image classification. In: **2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)**. IEEE, 2021. p. 490-497.	2021
EfficientNet V2	TAN, Mingxing; LE, Quoc. Efficientnetv2: Smaller models and faster training. In: **International conference on machine learning**. PMLR, 2021. p. 10096-10106.	2021
EfficientNet	TAN, Mingxing; LE, Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks. In: **International conference on machine learning**. PMLR, 2019. p. 6105-6114.	2019
NASNet	ZOPH, Barret et al. Learning transferable architectures for scalable image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2018. p. 8697-8710.	2018
MobileNet V2	SANDLER, Mark et al. Mobilenetv2: Inverted residuals and linear bottlenecks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2018. p. 4510-4520.	2018
Xception V1	CHOLLET, François. Xception: Deep learning with depthwise separable convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2017. p. 1251-1258.	2017
MobileNet	HOWARD, Andrew G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. **arXiv preprint arXiv**:1704.04861, 2017.	2017
Inception-ResNet V2	SZEGEDY, Christian et al. Inception-v4, inception-resnet and the impact of residual connections on learning. In: **Proceedings of the AAAI conference on artificial intelligence**. 2017.	2017
DenseNet	HUANG, Gao et al. Densely connected convolutional networks. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2017. p. 4700-4708.	2017
ResNet V2	HE, Kaiming et al. Identity mappings in deep residual networks. In: **Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. Springer International Publishing**, 2016. p. 630-645.	2016
ResNet / ResNet50	HE, Kaiming et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016. p. 770-778.	2016
Inception V3	SZEGEDY, Christian et al. Rethinking the inception architecture for computer vision. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016. p. 2818-2826.	2016
VGG16 / VGG19	SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.	2014

Fonte: Próprio Autor

APÊNDICE B

PRINCIPAIS CONJUNTOS DE DADOS PÚBLICOS EXISTENTES

Nome	Artigo	Normal	Glaucoma	Total	Finalidade
RIGA	ALMAZROA, Ahmed et al. Retinal fundus images for glaucoma analysis: the RIGA dataset. In: **Medical Imaging 2018: Imaging Informatics for Healthcare, Research, and Applications**. SPIE, 2018. p. 55-62.	-	-	750	Segmentação
ACRIMA	DIAZ-PINTO, Andres et al. CNNs for automatic glaucoma assessment using fundus images: an extensive validation. **Biomedical engineering online**, v. 18, p. 1-19, 2019.	309	396	705	Classificação
Drishti-S1	SIVASWAMY, Jayanthi et al. A comprehensive retinal image dataset for the assessment of glaucoma from the optic nerve head analysis. **JSM Biomedical Imaging Data Papers**, v. 2, n. 1, p. 1004, 2015.	31	70	101	Classificação
RIM-ONEv1	BATISTA, Francisco et al. Rim-one dl: A unified retinal image database for assessing glaucoma using deep learning. **Image Analysis & Stereology**, v. 39, n. 3, p. 161-167, 2020.	118	51	169	Classificação
RIM-ONEv2	BATISTA, Francisco et al. Rim-one dl: A unified retinal image database for assessing glaucoma using deep learning. **Image Analysis & Stereology**, v. 39, n. 3, p. 161-167, 2020.	255	200	455	Classificação

Fonte: Próprio Autor

APÊNDICE C

CÓDIGO FONTE UTILIZADO NO EXPERIMENTO

glaucoma.py

C: > Users > JULIO > Documents > GitHub > CNNs > glaucoma.py > generate_augmented_training_dataset

```python
import os
import math
import tensorflow as tf
import keras as K
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Dropout, GlobalAveragePooling2D

print("TensorFlow Version:", tf.__version__)
print("Keras Version:", K.__version__)

DATASET_DIR = "glaucoma_datasets"

def resolve_path(relative_path):
    return os.path.join(DATASET_DIR, relative_path)

```

```
20
21  def generate_augmented_training_dataset(directory_path, target_size = (224, 224), batch_size = 16):
22      train_datagen = ImageDataGenerator(
23          validation_split=0.2,
24          horizontal_flip = True,
25          vertical_flip = True,
26          rotation_range=40,
27          fill_mode='nearest')
28
29      training_set = train_datagen.flow_from_directory(
30          directory_path,
31          subset='training',
32          class_mode = 'binary',
33          target_size = target_size,
34          batch_size = batch_size,
35          shuffle=True)
36
37      validation_set = train_datagen.flow_from_directory(
38          directory_path,
39          subset='validation',
40          class_mode = 'binary',
41          target_size = target_size,
42          batch_size = batch_size,
43          shuffle=True)
44
45      return training_set, validation_set
46
```

```
46
47  def generate_test_dataset(directory_path, target_size = (224, 224), batch_size = 16):
48      test_datagen = ImageDataGenerator()
49
50      test_set = test_datagen.flow_from_directory(
51          directory_path,
52          class_mode = 'binary',
53          target_size = target_size,
54          batch_size = batch_size,
55          shuffle = False)
56
57      return test_set
58
```

```
84
85  def recompile_model(base_model, preprocess_input, training_set, input_shape = (224, 224, 3),
86                      base_learning_rate = 0.0001, show_summary = False):
87      if show_summary:
88          base_model.summary()
89
90      # Freeze convolutional layers
91      base_model.trainable = False
92
93      # New prediction layer
94      image_batch, label_batch = training_set.next()
95      feature_batch = base_model(image_batch)
96
97      global_average_layer = GlobalAveragePooling2D()
98      feature_batch_average = global_average_layer(feature_batch)
99
100     prediction_layer = Dense(1)
101     prediction_batch = prediction_layer(feature_batch_average)
102
103     # Model Assembly
104     inputs = Input(shape=input_shape)
105     x = preprocess_input(inputs)
106     x = base_model(x, training=False)
107     x = global_average_layer(x)
108     x = Dropout(0.5)(x)
109     outputs = prediction_layer(x)
110     classifier = Model(inputs, outputs)
111
112     if show_summary:
113         classifier.summary()
114
115     classifier.compile(
116         loss= tf.keras.losses.BinaryCrossentropy(from_logits=True),
117         optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_rate),
118         metrics=['accuracy'])
119
120     return classifier
121
```

```python

def load_convnext(input_shape=(224,224,3)):
    from tensorflow.keras.applications.convnext import ConvNeXtBase
    convnext = ConvNeXtBase(input_shape=input_shape, weights='imagenet', include_top=False)
    return convnext, tf.keras.applications.convnext.preprocess_input

def load_efficientnet(input_shape=(224,224,3)):
    from tensorflow.keras.applications.efficientnet_v2 import EfficientNetV2B0
    efficientnet = EfficientNetV2B0(input_shape=input_shape, weights='imagenet', include_top=False)
    return efficientnet, tf.keras.applications.efficientnet_v2.preprocess_input

def load_mobilenet(input_shape=(224,224,3)):
    from tensorflow.keras.applications import MobileNetV3Small
    mobilenet = MobileNetV3Small(input_shape=input_shape, weights='imagenet', include_top=False)
    return mobilenet, tf.keras.applications.mobilenet_v3.preprocess_input

def load_regnet(input_shape=(224,224,3)):
    from tensorflow.keras.applications.regnet import RegNetX002
    regnet = RegNetX002(input_shape=input_shape, weights='imagenet', include_top=False)
    return regnet, tf.keras.applications.regnet.preprocess_input

def load_resnet_rs(input_shape=(224,224,3)):
    from tensorflow.keras.applications.resnet_rs import ResNetRS270
    resnet_rs = ResNetRS270(input_shape=input_shape, weights='imagenet', include_top=False)
    return resnet_rs, tf.keras.applications.resnet_rs.preprocess_input

```

```
121
122 def fit_model(classifier, training_set, validation_set, epochs = 10):
123     history = classifier.fit(
124         training_set,
125         validation_data = validation_set,
126         epochs = epochs,
127         steps_per_epoch = training_set.samples // training_set.batch_size,
128         validation_steps = validation_set.samples // validation_set.batch_size)
129
130     return history
131
```

```
131
132 def plot_training_validation_accuracy(history):
133     epochs = history.params['epochs']
134     epochs_range = range(epochs)
135
136     acc = history.history['accuracy']
137     val_acc = history.history['val_accuracy']
138
139     plt.figure(figsize=(8, 3))
140     plt.subplot(1, 2, 1)
141     plt.plot(epochs_range, acc, label='Training Accuracy')
142     plt.plot(epochs_range, val_acc, label='Validation Accuracy')
143     plt.xlabel('Epoch')
144     plt.ylabel('Accuracy')
145     plt.ylim([0.5, 1])
146     plt.legend(loc='lower right')
147     plt.title('Training and Validation Accuracy')
148
149     loss = history.history['loss']
150     val_loss = history.history['val_loss']
151
152     plt.subplot(1, 2, 2)
153     plt.plot(epochs_range, loss, label='Training Loss')
154     plt.plot(epochs_range, val_loss, label='Validation Loss')
155     plt.xlabel('Epoch')
156     plt.ylabel('Loss')
157     plt.legend(loc='lower right')
158     plt.title('Training and Validation Loss')
159     plt.show()
160
```

```
160
161 def fine_tune(base_model, classifier, training_set, validation_set, history,
162               fine_tune_at=100, base_learning_rate = 0.0001, initial_epochs = 10,
163               fine_tune_epochs = 10, show_summary = False):
164     # Unfreeze all layers starting from 'fine_tune_at' layer
165     base_model.trainable = True
166     for layer in base_model.layers[:fine_tune_at]:
167         layer.trainable = False
168
169     classifier.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
170               optimizer = tf.keras.optimizers.RMSprop(learning_rate=base_learning_rate/10),
171               metrics=['accuracy'])
172
173     total_layers = len(base_model.layers)
174     print(f"Fine adjustment applied starting from {fine_tune_at} of {total_layers}")
175     if show_summary:
176         classifier.summary()
177         print(len(classifier.trainable_variables))
178
179     total_epochs =  initial_epochs + fine_tune_epochs
180
181     history_fine = classifier.fit(training_set,
182                                   epochs=total_epochs,
183                                   initial_epoch=history.epoch[-1],
184                                   validation_data=validation_set)
185
186     return history_fine
187
```

```
187
188 def plot_fine_tune_training_validation_accuracy(history, history_fine):
189     initial_epochs = history.params['epochs']
190
191     acc = history.history['accuracy']
192     val_acc = history.history['val_accuracy']
193
194     acc += history_fine.history['accuracy']
195     val_acc += history_fine.history['val_accuracy']
196
197     plt.figure(figsize=(8, 3))
198     plt.subplot(1, 2, 1)
199     plt.plot(acc, label='Training Accuracy')
200     plt.plot(val_acc, label='Validation Accuracy')
201     plt.xlabel('Epoch')
202     plt.ylabel('Accuracy')
203     plt.ylim([0.5, 1])
204     plt.plot([initial_epochs - 1,initial_epochs - 1], plt.ylim(), label='Start Fine Tuning')
205     plt.legend(loc='lower right')
206     plt.title('Training and Validation Accuracy')
207
208     loss = history.history['loss']
209     val_loss = history.history['val_loss']
210
211     loss += history_fine.history['loss']
212     val_loss += history_fine.history['val_loss']
213
214     plt.subplot(1, 2, 2)
215     plt.plot(loss, label='Training Loss')
216     plt.plot(val_loss, label='Validation Loss')
217     plt.xlabel('Epoch')
218     plt.ylabel('Loss')
219     plt.ylim([0, 1.0])
220     plt.plot([initial_epochs - 1,initial_epochs - 1], plt.ylim(), label='Start Fine Tuning')
221     plt.legend(loc='upper right')
222     plt.title('Training and Validation Loss')
223     plt.show()
224
```

```
224
225 def get_real_and_predicted_labels(model, validation_generator):
226     predictions_list = np.array([])
227     labels_list = np.array([])
228
229     for i in range(len(validation_generator)):
230         x_batch, y_batch = validation_generator[i]
231
232         # Get the model prediction
233         predictions = model.predict_on_batch(x_batch).flatten()
234
235         # Get predicted labels
236         predictions = tf.nn.sigmoid(predictions)
237         predictions = tf.where(predictions < 0.5, 0, 1)
238         predictions_list = np.concatenate([predictions_list, predictions])
239
240         # Get expected labels
241         labels = y_batch.flatten()
242         labels_list = np.concatenate([labels_list, labels])
243
244     return [labels_list, predictions_list]
245
```

```
245
246 def plot_confusion_matrix(classifier, test_set):
247     all_labels, all_predictions = get_real_and_predicted_labels(classifier, test_set)
248     conf_matrix = tf.math.confusion_matrix(all_labels, all_predictions)
249
250     sorted_names = sorted(test_set.class_indices.items(), key=lambda x: x[1])
251     class_names = list(map(lambda x: x[0], sorted_names))
252
253     plt.figure(figsize=(3, 3))
254     sns.heatmap(
255         conf_matrix,
256         annot=True,
257         fmt="d",
258         cmap=plt.cm.Blues,
259         xticklabels=class_names,
260         yticklabels=class_names)
261
262     plt.tight_layout()
263     plt.ylabel('True label')
264     plt.xlabel('Predicted label')
265     plt.show()
266
```

```
266
267 def get_real_labels_and_predicted_probabilities(model, validation_generator):
268     labels_list = np.array([])
269     probabilities_list = np.array([])
270
271     for i in range(len(validation_generator)):
272         x_batch, y_batch = validation_generator[i]
273
274         # Get the model prediction
275         predictions = model.predict_on_batch(x_batch).flatten()
276
277         # Get probability of the diagnosis being positive
278         probabilities = tf.nn.sigmoid(predictions)
279         probabilities_list = np.concatenate([probabilities_list, probabilities])
280
281         # Get expected labels
282         labels = y_batch.flatten()
283         labels_list = np.concatenate([labels_list, labels])
284
285     return [labels_list, probabilities_list]
286
```

```
286
287 def plot_roc_curve(classifiers, test_set):
288     # Plot ROC curve
289     plt.figure(figsize=(4, 4))
290     plt.plot([0, 1], [0, 1], 'k--', label=f"Random (AUC = {.5:.4f})")
291
292     for classifier_name, classifier in classifiers:
293         labels, probabilities = get_real_labels_and_predicted_probabilities(classifier, test_set)
294         fpr, tpr, thresholds = roc_curve(labels, probabilities)
295         auc_score = roc_auc_score(labels, probabilities)
296         plt.plot(fpr, tpr, label=f"{classifier_name} (AUC = {auc_score:.4f})")
297
298     plt.legend(loc='lower right')
299     plt.xlabel('False Positive Rate')
300     plt.xlim([0, 1])
301     plt.ylabel('True Positive Rate')
302     plt.ylim([0, 1])
303     plt.title('ROC Curve')
304     plt.show()
305
```

```
305
306 def export_classifier(classifier, filepath):
307     model_json = classifier.to_json()
308     with open(f'{filepath}.json', "w") as json_file:
309         json_file.write(model_json)
310     classifier.save_weights(f'{filepath}.h5')
311
```

Fonte: Adaptado de TensorFlow (2023)