

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

IGOR MACHADO CRUZ GUIMARÃES OLIVEIRA
Orientador: Pedro Henrique Lopes Silva

**CLASSIFICAÇÃO DE CÉLULAS INFECTADAS POR MALÁRIA
UTILIZANDO REPRESENTAÇÕES PROFUNDAS**

Ouro Preto, MG
2025

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

IGOR MACHADO CRUZ GUIMARÃES OLIVEIRA

**CLASSIFICAÇÃO DE CÉLULAS INFECTADAS POR MALÁRIA UTILIZANDO
REPRESENTAÇÕES PROFUNDAS**

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Pedro Henrique Lopes Silva

Ouro Preto, MG
2025

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

O48c Oliveira, Igor Machado Cruz Guimaraes.
Classificação de células infectadas por malária utilizando
representações profundas. [manuscrito] / Igor Machado Cruz Guimaraes
Oliveira. - 2025.
55 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Pedro Henrique Lopes Silva.
Monografia (Bacharelado). Universidade Federal de Ouro Preto.
Instituto de Ciências Exatas e Biológicas. Graduação em Ciência da
Computação .

1. Inteligência artificial. 2. Redes Neurais Convolucionais. 3. Imagens
como recursos de informação. 4. Malária. I. Silva, Pedro Henrique Lopes.
II. Universidade Federal de Ouro Preto. III. Título.

CDU 004.8

Bibliotecário(a) Responsável: Soraya Fernanda Ferreira e Souza - SIAPE: 1.763.787



FOLHA DE APROVAÇÃO

Igor Machado Cruz Guimarães Oliveira

Classificação de Células Infectadas por Malária Utilizando Representações Profundas

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 2 de Abril de 2025.

Membros da banca

Pedro Henrique Lopes Silva (Orientador) - Doutor - Universidade Federal de Ouro Preto
Guilherme Augusto Lopes Silva (Examinador) - Mestre - Programa de Pós-Graduação em Ciência da Computação - UFOP
Pablo Martins Coelho (Examinador) - Bacharel - Programa de Pós-Graduação em Ciência da Computação - UFOP

Pedro Henrique Lopes Silva, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 2/04/2025.



Documento assinado eletronicamente por **Pedro Henrique Lopes Silva, PROFESSOR DE MAGISTERIO SUPERIOR**, em 02/04/2025, às 14:08, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0886293** e o código CRC **8617F2E7**.

*Dedico esse trabalho a todos que apoiaram minha jornada até aqui, em especial meus pais
Célio da Conceição Guimarães Oliveira e Rosana Fátima Machado Cruz Guimarães.*

Agradecimentos

Primeiramente quero agradecer a minha família, que além de sempre apoiar minhas decisões, me proporcionaram a oportunidade de estudar o que queria. Aos meus pais, Célio e Rosana, pelos ensinamentos e todo apoio que sempre me deram, e às minhas irmãs Juliane e Júlia, que sempre estiveram comigo. Amo todos vocês.

A minha namorada Ana Luiza, que me apoiou durante toda minha graduação em todos os momentos difíceis, mas também vibrou comigo em todas as minhas conquistas. Juntos passamos por momentos importantes nesse tempo, que fortaleceu ainda mais o nosso vínculo. Muito obrigado, eu te amo!

Ao meu professor orientador, Pedro Henrique Lopes Silva, pela orientação e apoio não só durante a monografia, mas durante toda a graduação. O resultado desse trabalho se deve a atenção, paciência e compromisso que teve comigo, muito obrigado!

Agradeço também aos meus amigos da Lokomotiva que me acompanharam durante a graduação, Arthur (Negrão/ Tonton), Paulo Henrique (Paulinho/ Bovs), Gabriel (Gabrielzinho), Matheus (Mafioso/ Rato). Juntos vivemos momentos que me recordarei para o resto de minha vida.

“Eu sei que a vida acaba, e lá no final da vida quando eu olhar para trás,
eu quero comemorar, e não me lamentar.”

Eduardo Marinho (2022)

Resumo

A malária é uma doença infecciosa, especialmente em países emergentes. Este trabalho teve como objetivo principal desenvolver e avaliar um modelo de classificação para identificar *Red Blood Cells* (RBC) infectadas e não infectadas por malária, utilizando a arquitetura de rede neural convolucional EfficientNet-B0 e o *Vision Transformer* (ViT), explorando o uso de representações profundas. Foram aplicadas técnicas como *Transfer Learning* (TL) e *Cosine Decay* para otimizar o treinamento do modelo EfficientNet-B0. A metodologia envolveu dois cenários de avaliação: o primeiro com validação cruzada estratificada, alcançando uma acurácia de 94% com EfficientNet-B0 e TL, e o segundo com uma divisão em treino, validação e teste baseada na literatura, onde o EfficientNet-B0 atingiu 96% de acurácia com TL e *Cosine Decay*. A aplicação de *Super Resolution* (SR) como pré-processamento, utilizando o *Enhanced Super-Resolution Generative Adversarial Network* (ESRGAN), combinada com o ViT, resultou em uma acurácia de 97%. Os resultados indicam que o uso de TL e *Cosine Decay* foi crucial para otimizar o desempenho do EfficientNet-B0, especialmente no *recall* para a classe infectada. Adicionalmente, o ViT, precedido por SR, demonstrou resultados competitivos, equiparando-se ao estado da arte. Conclui-se que os modelos propostos, representam abordagens eficazes e promissoras para a classificação de malária, com desempenhos competitivos em relação à literatura existente.

Palavras-chave: Malária. Classificação de Imagens. EfficientNet-B0. *Vision Transformer*. *Super-Resolution*.

Abstract

Malaria is an infectious disease, especially in emerging countries. This study aimed to develop and evaluate a classification model to identify Red Blood Cells (RBC) infected and non-infected by malaria, using the EfficientNet-B0 convolutional neural network architecture and Vision Transformer (ViT), exploring the use of deep representations. Techniques such as Transfer Learning (TL) and Cosine Decay were applied to optimize the training of the EfficientNet-B0 model. The methodology involved two evaluation scenarios: the first with stratified cross-validation, achieving an accuracy of 94% with EfficientNet-B0 and TL, and the second with a training, validation, and test split based on the literature, where EfficientNet-B0 reached 96% accuracy with TL and Cosine Decay. The application of Super Resolution (SR) as preprocessing, using Enhanced Super-Resolution Generative Adversarial Network (ESRGAN), combined with ViT, resulted in 97% accuracy. The results indicate that the use of TL and Cosine Decay was crucial to optimizing the performance of EfficientNet-B0, especially in recall for the infected class. Additionally, ViT, preceded by SR, demonstrated competitive results, matching the state-of-the-art. It is concluded that the proposed models represent effective and promising approaches for malaria classification, achieving competitive performances compared to the existing literature.

Keywords: *Malaria. Image Classification. EfficientNet-B0. Vision Transformer. Super-Resolution.*

Lista de Ilustrações

Figura 1.1 – Ciclo de transmissão da malária.	1
Figura 2.1 – Exemplo de uma rede neural.	8
Figura 2.2 – Representação de como o <i>backpropagation</i> funciona através do ajuste de pesos.	10
Figura 2.3 – Exemplo de um processo realizado por uma CNN.	11
Figura 2.4 – Exemplo de como o modelo pode se adaptar aos dados de teste.	12
Figura 2.5 – Processo de funcionamento da técnica de <i>Dropout</i>	13
Figura 2.6 – Exemplo do resultado da aplicação da técnica de <i>Data Augmentation</i> . Em (a) está a imagem original, em (b) foi aplicada uma rotação de 180°, em (c) a conversão para escala de cinza, em (d) um zoom-in na imagem e em (e) a pixelização combinada com um espelhamento horizontal.	14
Figura 2.7 – Estratégias de Escalonamento na Arquitetura EfficientNet.	14
Figura 2.8 – Arquitetura base da rede EfficientNet-B0.	16
Figura 2.9 – Exemplo do funcionamento de <i>Transfer Learning</i>	16
Figura 2.10–Figura que ilustra o processo de validação cruzada <i>K-fold</i>	17
Figura 2.11–Ilustração do processo de <i>Super Resolution</i>	19
Figura 2.12–Estrutura interna do bloco <i>Residual-in-Residual Dense Block</i>	21
Figura 2.13–Arquitetura do modelo ESRGAN.	21
Figura 2.14–Estrutura do ViT, que divide imagens em <i>patches</i> e utiliza um codificador baseado em <i>Transformer</i> para extrair características globais.	23
Figura 2.15–Arquiteturas avaliadas por Harahap et al. (2021).	25
Figura 3.1 – Exemplos de imagens contidas na base de dados. As imagens (a) e (b) representam células infectadas e (c) e (d), células não infectadas.	28
Figura 3.2 – Figura que ilustra o fluxo do modelo.	32
Figura 3.3 – Função <i>Sigmoid</i>	33
Figura 3.4 – Exemplo de imagem original (a) e sua versão aumentada por data augmentation (b).	35
Figura 3.5 – Exemplo de imagem original (a) e sua versão após o processo de SR (b).	36

Lista de Tabelas

Tabela 2.1 – Resultados obtidos nos três cenários propostos.	24
Tabela 4.1 – Resultados obtidos com <i>Transfer Learning</i> (TL) para 5 <i>Folds</i> para os dois cenários de configuração de <i>learning rate</i>	41
Tabela 4.2 – Resultados obtidos sem e com <i>Transfer Learning</i> (TL) utilizando o modelo EfficientNet-B0 para 5 <i>Folds</i>	43
Tabela 4.3 – Parâmetros usados para realizar o <i>data augmentation</i>	45
Tabela 4.4 – Resultados do modelo proposto com e sem <i>data augmentation</i>	46
Tabela 4.5 – Resultados do modelo proposto com diferentes otimizadores para SR.	46
Tabela 4.6 – Resultados obtidos com o treinamento do ViT.	47
Tabela 4.7 – Comparação entre modelos da literatura com os modelos testados.	47

Lista de Algoritmos

2.1	Definição da Taxa de Aprendizado com <i>Cosine Decay</i>	18
-----	--------------------------------------------------------------------	----

Lista de Abreviaturas e Siglas

- CNN** *Convolutional Neural Network*. ix, 2, 6, 10–12, 15, 19, 22, 24–28, 30, 31, 49
- ESRGAN** *Enhanced Super-Resolution Generative Adversarial Network*. vii–ix, xiv, 20–22, 46, 49, 50
- FN** *False Negatives*. 36
- FP** *False Positives*. 36
- GAN** *Generative Adversarial Networks*. 19
- IA** *Inteligência Artificial*. 6, 22
- LoG** *Laplacian of Gaussian*. 28, 29
- MBCONV** *Mobile Inverted Bottleneck Conv*. 15
- RaGAN** *Relativistic average Discriminator*. 21
- RBC** *Red Blood Cells*. vii, viii, 2, 7, 28, 29, 35, 44, 46, 49
- RRDB** *Residual-in-Residual Dense Block*. ix, 20, 21
- SE** *Squeeze and Excitation*. 15
- SR** *Super Resolution*. vii–x, xiv, xv, 3–5, 19–22, 31, 34–36, 39, 46–50
- SRCNN** *Super-Resolution Convolutional Neural Network*. 19
- SRGAN** *Super-Resolution Generative Adversarial Network*. 19, 20
- TL** *Transfer Learning*. vii, viii, x, xiv, 2, 3, 15, 23, 24, 40–44, 49
- TN** *True Negatives*. 36
- TP** *True Positives*. 36
- ViT** *Vision Transformer*. vii–ix, xiv, xv, 3, 4, 22, 23, 26, 31, 34, 35, 39, 40, 46–48, 50

Lista de Símbolos

Σ	Somatório
∇	Gradiente (operador nabla)
π	Letra grega minúscula pi
α	Letra grega minúscula alfa
σ	Letra grega minúscula sigma
η	Letra grega minúscula eta
ϕ	Letra grega minúscula phi
∂	Derivada parcial (símbolo del)

Sumário

1	Introdução	1
1.1	Justificativa	3
1.2	Objetivos	4
1.3	Organização do Trabalho	5
2	Revisão Bibliográfica	6
2.1	Fundamentação Teórica	6
2.1.1	Inteligência Artificial e <i>Machine Learning</i>	6
2.1.2	<i>Deep Learning</i>	7
2.1.3	Backpropagation	9
2.1.4	<i>Convolutional Neural Network (CNN)</i>	10
2.1.5	<i>Overfitting</i>	12
2.1.6	<i>Dropout</i>	12
2.1.7	<i>Data Augmentation</i>	13
2.1.8	Família das arquiteturas EfficientNet	14
2.1.9	<i>Transfer Learning</i>	16
2.1.10	<i>StratifiedKFold</i>	17
2.1.11	<i>Cosine Decay</i>	18
2.1.12	<i>Super Resolution</i>	19
2.1.13	<i>Enhanced Super-Resolution Generative Adversarial Network (ESRGAN)</i>	20
2.1.14	<i>Vision Transformer (ViT)</i>	22
2.2	Trabalhos Relacionados	23
3	Base de dados e Metodologia Proposta	28
3.1	Base de dados	28
3.1.1	Pré-processamento da base	30
3.1.2	Divisão dos dados	30
3.2	Modelos Propostos	31
3.2.1	EfficientNet-B0	31
3.2.2	<i>Vision Transformer (ViT)</i>	34
3.2.3	Estratégia com <i>Data Augmentation</i>	34
3.2.4	Estratégia com <i>Super Resolution</i>	35
3.2.5	Métricas	36
4	Experimentos e Resultados	39
4.1	Setup dos experimentos	39
4.2	Avaliação com validação cruzada estratificada	40
4.3	Avaliação do <i>Transfer Learning (TL)</i>	44
4.4	Avaliação do <i>Data Augmentation</i>	45

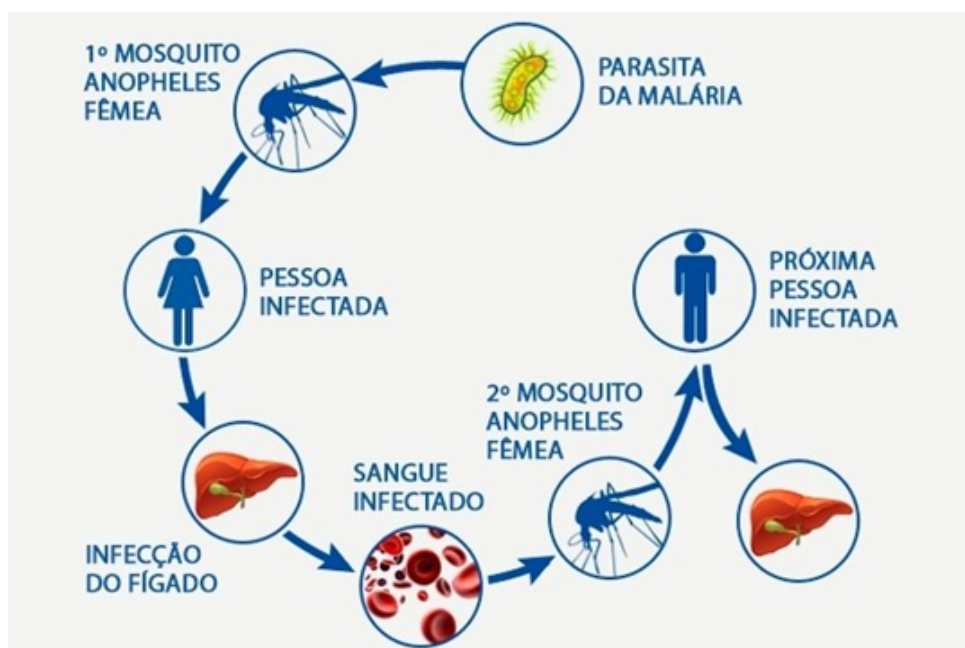
4.5	Avaliação do <i>Super Resolution</i> (SR)	46
4.6	Avaliação do <i>Vision Transformer</i> (ViT)	46
4.7	Comparação com a literatura	47
5	Considerações Finais	49
5.1	Conclusão	49
5.2	Trabalhos Futuros	50
	Referências	51

1 Introdução

A malária é uma doença infecciosa causada pelo parasita do gênero *Plasmodium*, transmitida para os seres humanos através da picada de mosquitos fêmeas do gênero *Anopheles*, popularmente conhecidos como mosquitos-prego (Secretaria de Estado da Saúde do Pará, 2023). A malária é uma doença que afeta milhares de pessoas no mundo, especialmente em regiões tropicais e subtropicais, como a África subsaariana, o Sudeste Asiático e partes da América do Sul (Secretaria de Estado da Saúde do Pará, 2023). No Brasil, a região Amazônica se destaca com o maior número de casos, isso se deve ao fato de existirem muitos locais com água limpa e parada, ambientes esses propícios para a proliferação do mosquito (Secretaria de Estado da Saúde do Pará, 2023).

A transmissão da malária ocorre quando uma fêmea do mosquito *Anopheles* infectada pica uma pessoa, injetando parasitas do gênero *Plasmodium* diretamente na corrente sanguínea como visto na Figura 1.1. Os sintomas iniciais da malária surgem entre sete e 30 dias após a infecção e incluem febre, calafrios, sudorese, dores de cabeça, fadiga e dores musculares (SHEKAR; REVATHY; GOUD, 2020). Se não tratada rapidamente, a malária pode evoluir para formas graves, causando complicações como anemia severa, danos a órgãos, disfunção cerebral e, em casos críticos, morte (Secretaria de Estado da Saúde do Pará, 2023). O diagnóstico da malária é geralmente realizado através da análise de uma amostra de sangue, procurando a presença dos parasitas *Plasmodium*.

Figura 1.1 – Ciclo de transmissão da malária.



Fonte: (Secretaria de Estado da Saúde do Pará, 2023).

O diagnóstico precoce e preciso da malária é crucial para o tratamento eficaz e a redução dos efeitos adversos da doença. O método tradicional, embora seja o padrão, possui várias limitações, incluindo a necessidade de técnicos qualificados, a variabilidade dos resultados devido à subjetividade do observador e o tempo necessário para a análise detalhada das amostras (REDDY; JULIET, 2019). Além disso, os laboratórios de saúde frequentemente enfrentam problemas com a disponibilidade de recursos, entre eles recursos computacionais, se considerarmos as regiões onde a malária é mais agravante. Essas limitações podem levar a falsos negativos, atrasos no tratamento e, conseqüentemente, a um aumento na severidade da doença e nas taxas de mortalidade (REDDY; JULIET, 2019).

Com o avanço das técnicas de inteligência artificial, especialmente no campo do *Deep Learning*, surgem novas oportunidades para aprimorar os métodos de classificação da malária. A utilização de *Deep Learning* para o diagnóstico da malária apresenta várias vantagens (REDDY; JULIET, 2019). Primeiramente, os modelos podem ser treinados para detectar o parasita com alta precisão, diminuindo a dependência de técnicos especializados e minimizando erros humanos (REDDY; JULIET, 2019). Além disso, esses modelos podem processar grandes quantidades de amostras em um curto espaço de tempo, facilitando o diagnóstico em massa e melhorando a eficiência da saúde pública e privada (REDDY; JULIET, 2019).

Nesse contexto, a arquitetura EfficientNet-B0 apresenta-se como uma arquitetura em potencial para a geração de um modelo que possa automatizar os diagnósticos de malária, visto que essa *Convolutional Neural Network* (CNN) já apresentou resultados positivos na detecção e classificação em outras áreas da saúde, como por exemplo no estudo de Shah et al. (2022), que utilizou a EfficientNet para a detecção de tumor cerebral. EfficientNet é uma família de modelos de redes neurais convolucionais que se destaca pela sua eficiência e precisão, obtendo bons resultados de classificação utilizando imagens (TAN; LE, 2019). A arquitetura EfficientNet-B0, em particular, combina simplicidade e eficácia, quando comparada com outros exemplos, oferecendo uma solução que pode ser implementada em sistemas com recursos limitados (KLINGLER, 2024) (TAN; LE, 2019).

Uma das hipóteses deste trabalho é avaliar se a EfficientNet-B0 é capaz de classificar corretamente imagens de *Red Blood Cells* (RBC) segmentadas entre as que contêm e as que não contêm malária. A pesquisa busca avaliar a eficácia da EfficientNet-B0 em diferentes cenários, comparando seu desempenho com e sem o uso de *Transfer Learning* (TL). O uso de TL é justificado por sua capacidade de acelerar o processo de treinamento, o que é interessante em ambientes com recursos computacionais limitados. Dessa forma, a combinação da EfficientNet-B0 com o TL é uma solução que pode ser prática em laboratórios com restrições de infraestrutura. Além disso, os resultados serão comparados com o estado da arte para determinar a eficácia relativa da arquitetura na identificação automática de malária em diferentes condições.

Utilizou-se a técnica de validação cruzada no modelo com EfficientNet-B0 (*stratified K-fold*) para garantir que cada subconjunto de dados mantivesse a proporção original das classes.

Os resultados obtidos até o momento são promissores: a acurácia alcançada foi de cerca de 95% sem a utilização de pesos pré-treinados na *ImageNet* e aproximadamente 94% utilizando a técnica de TL com pesos da *ImageNet*. Isso indica que, apesar das vantagens do TL, o ajuste fino dos pesos para o contexto específico da malária é necessário para alcançar melhores resultados.

Nesse mesmo contexto, o *Vision Transformer* (ViT) surge como uma abordagem promissora para a classificação de imagens, apresentando resultados competitivos e, em alguns casos, superando arquiteturas tradicionais baseadas em redes neurais convolucionais (DOSOVITSKIY et al., 2020). Sua capacidade de modelar relações globais dentro da imagem o torna uma alternativa interessante para a identificação de padrões sutis, o que pode ser interessante na classificação de malária. No entanto, o ViT é uma rede mais complexa em termos computacionais, o que pode impactar a eficiência do treinamento, exigindo estratégias adequadas para otimização ViT. Ainda assim, sua crescente adoção na literatura apoia outra hipótese desse estudo, de que existe um potencial para alcançar ou até mesmo superar os resultados estado da arte na tarefa de classificação de glóbulos vermelhos infectados por malária, como por exemplo o estudo de Minarno et al. (2023).

Além disso, considerando que as imagens utilizadas no estudo correspondem a glóbulos vermelhos – estruturas pequenas que podem conter detalhes sutis –, a aplicação de técnicas de *Super Resolution* (SR) para aumentar a resolução das imagens pode contribuir para a melhoria dos resultados (WANG et al., 2018). Métodos de *Super Resolution* (SR) buscam reconstruir detalhes finos em imagens de baixa resolução, permitindo uma melhor definição das características relevantes para a classificação. Dessa forma, este trabalho inclui a aplicação dessa técnica para avaliar seu impacto no desempenho dos modelos EfficientNet-B0 e ViT. A hipótese é que o aprimoramento da resolução das imagens possa favorecer a extração de características discriminantes, potencialmente melhorando a acurácia dos modelos na identificação de células infectadas.

1.1 Justificativa

A malária continua a ser uma das principais doenças infecciosas, e sua detecção precoce e precisa é importante para o tratamento da doença e, conseqüentemente, a redução dos casos de mortalidade associados à mesma (MINARNO et al., 2023).

A escolha deste tema, também engloba a tentativa de aplicar a tecnologia de forma a investigar possibilidades para futuros panoramas relacionados à classificação de imagens na área da saúde utilizando *machine learning*, em especial nesse estudo, a malária. Além disso, a escolha da arquitetura EfficientNet-B0 se justifica por sua eficiência em termos de custo computacional (TAN; LE, 2019), tornando-a uma opção viável para implementação em dispositivos com recursos limitados, como clínicas de saúde em países emergentes. Essa característica permite que o modelo seja acessível e escalável, possibilitando sua adoção em larga escala em regiões onde a malária é

endêmica.

Além do EfficientNet-B0, a inclusão do *Vision Transformer* (ViT) se justifica devido ao seu potencial em tarefas de classificação de imagens, tendo alcançado resultados positivos no contexto de classificação de imagens (DOSOVITSKIY et al., 2020). Embora seja uma arquitetura mais custosa computacionalmente, o ViT apresenta uma abordagem diferenciada para a extração de características, modelando relações globais dentro da imagem, o que pode contribuir para um aumento na precisão da classificação de glóbulos vermelhos infectados. Assim, sua avaliação neste estudo permite investigar se essa abordagem pode superar o estado da arte na classificação de malária.

Adicionalmente, a aplicação da técnica de *Super Resolution* (SR) se justifica pelo fato de que as imagens analisadas correspondem a glóbulos vermelhos, estruturas pequenas onde detalhes importantes podem ser perdidos devido à resolução limitada. O uso de SR pode aprimorar a resolução dessas imagens, permitindo uma extração mais eficiente de características relevantes para a classificação. Avaliar seu impacto nos modelos testados pode demonstrar como a melhoria da resolução influencia a precisão da detecção da malária.

Portanto, este estudo tem o potencial de contribuir não apenas para a melhoria do diagnóstico da malária, mas também para a criação de soluções tecnológicas mais acessíveis e eficientes na área da saúde.

1.2 Objetivos

O principal objetivo deste trabalho é estudar e desenvolver modelos baseados em *deep learning* para a identificação automática de parasitas do gênero *Plasmodium* em imagens de esfregaços de sangue. Para isso, utilizam-se as arquiteturas EfficientNet-B0 e ViT, comparando seus respectivos desempenhos na tarefa de diagnóstico da malária. Através da aplicação de técnicas de *deep learning* e aprimoramento de dados, busca-se criar um modelo capaz de automatizar esse processo, minimizando os desafios encontrados no diagnóstico convencional e potencialmente superando os resultados obtidos na literatura.

Para se alcançar os resultados desejados, deve-se cumprir os seguintes objetivos:

- **Seleção e preparação do conjunto de dados.** Será utilizada uma base de imagens de esfregaços sanguíneos contendo parasitas do gênero *Plasmodium*, com classificações fornecidas por especialistas. Além de garantir que a base seja representativa e equilibrada, serão aplicadas técnicas de processamento para padronização e adequação das imagens ao treinamento dos modelos.
- **Avaliação do desempenho dos modelos.** O desempenho das arquiteturas EfficientNet-B0 e ViT será avaliado com diferentes abordagens. Primeiramente, será aplicada a técnica de validação cruzada estratificada para obter uma análise inicial da performance do modelo

EfficientNet-B0. Em seguida, os resultados serão comparados utilizando a mesma divisão da base de dados presente em estudos anteriores, permitindo uma avaliação direta em relação à literatura. As métricas analisadas incluem acurácia, precisão e revocação.

- **Aplicação de técnicas de aprimoramento de dados e modelo.** Serão aplicadas estratégias como *Super Resolution* e *Data Augmentation* para avaliar seu impacto no desempenho dos modelos. A eficácia dessas abordagens será analisada comparando os resultados obtidos com e sem a aplicação dessas técnicas, considerando métricas como acurácia, precisão e revocação.
- **Análise comparativa dos resultados.** Por fim, os resultados obtidos serão comparados, além de uma análise das diferentes configurações dos modelos para determinar qual abordagem oferece o melhor desempenho. Esses resultados também podem ser comparados com estudos considerados estado da arte na literatura, que utilizaram metodologias similares, e a mesma base de dados, a fim de avaliar a eficácia da abordagem proposta.

1.3 Organização do Trabalho

O [Capítulo 2](#) oferece uma revisão da literatura relacionada ao tema, abordando tanto os estudos que aplicam diferentes técnicas de *deep learning* para classificação da malária quanto a base teórica necessária para compreender as técnicas utilizadas e o estudo desenvolvido. O [Capítulo 3](#) detalha a metodologia adotada, descrevendo as etapas do desenvolvimento e a implementação dos modelos propostos. No [Capítulo 4](#), são apresentados e discutidos os resultados obtidos nos experimentos. Finalmente, o [Capítulo 5](#) contém as conclusões finais e sugere opções para pesquisas futuras.

2 Revisão Bibliográfica

Este capítulo proporciona uma base teórica utilizada para a compreensão dos conceitos e metodologias empregadas no presente estudo. Em seguida, são apresentados trabalhos relacionados, com o objetivo de contextualizar o tema a ser abordado: a aplicação de *Deep Learning* para a classificação do parasita da malária em células sanguíneas.

2.1 Fundamentação Teórica

Neste capítulo, serão abordados os conceitos teóricos essenciais para a compreensão do estudo desenvolvido. Em um primeiro momento, serão discutidos os fundamentos de inteligência artificial e *machine learning*. Na sequência, será contemplada a área de *deep learning*, com uma explicação sobre o que são e como funcionam as redes neurais artificiais, mais especificamente as **CNNs**. Após isso, será discutida a aplicação da técnica de *transfer learning* e a arquitetura *EfficientNet*, que foi a escolhida para ser utilizada no estudo. Por fim, será abordada a técnica *StratifiedKFold*, explicando seu papel na validação cruzada estratificada e como ela contribui para a avaliação mais robusta dos modelos de *machine learning*.

2.1.1 Inteligência Artificial e *Machine Learning*

A **Inteligência Artificial (IA)** é um campo de pesquisa criado e amplamente estudado, o qual atua na reprodução de padrões de comportamento semelhantes aos humanos por meio de técnicas, algoritmos e programas computacionais (PEREIRA, 2023). Atualmente existem vários subcampos dentro da área de **IA**, que avaliam métodos matemáticos e algoritmos para “ensinar” máquinas a executarem determinadas tarefas (PEREIRA, 2023). A **IA** é aplicada em diferentes áreas da sociedade, como na saúde, finanças e transporte (PANNU, 2015), com o objetivo de ser uma ferramenta que possa facilitar e até mesmo aprimorar todos esses campos.

Uma das áreas de **IA** é *Machine Learning*. Ela se concentra no desenvolvimento de algoritmos e modelos estatísticos que capacitam sistemas computacionais a realizar tarefas complexas sem instruções diretas, baseando-se em padrões e análises extraídas de grandes volumes de dados (PEREIRA, 2023). Essa tecnologia é essencial para empresas e diferentes setores da sociedade, como manufatura, saúde, finanças, varejo e mídia, onde é aplicada em uma variedade de cenários, desde manutenção preditiva e diagnóstico médico até análise de riscos financeiros e personalização de experiências de compra (PEREIRA, 2023). Ao automatizar processos e analisar rapidamente grandes conjuntos de dados, *machine Learning* impulsiona o crescimento e resolve desafios complexos, oferecendo um potencial significativo para aprimorar a eficiência e a inovação em diversos campos (PEREIRA, 2023).

Existem quatro tipos principais de aprendizado em *Machine Learning*: supervisionado, não supervisionado, semi-supervisionado e por reforço. No aprendizado supervisionado, o conjunto de dados é rotulado, ou seja, os dados de entrada já possuem seus valores de saída correspondentes definidos, permitindo ao modelo aprender a relação entre os dois (JANIESCH; ZSCHECH; HEINRICH, 2021). Isso é usado em problemas de regressão, onde valores numéricos são previstos, e em problemas de classificação, onde a saída é uma classe já categorizada. Exemplos de técnicas supervisionadas incluem regressão linear, árvores de decisão e máquinas de vetores de suporte (SVM) (JANIESCH; ZSCHECH; HEINRICH, 2021).

No aprendizado não supervisionado, não há rótulos ou respostas predefinidas; em vez disso, o sistema busca padrões nos dados para agrupá-los ou reduzir sua dimensionalidade (JANIESCH; ZSCHECH; HEINRICH, 2021). Um tipo de técnica comum são os algoritmos de clusterização, como o K-Means (JANIESCH; ZSCHECH; HEINRICH, 2021).

Já o aprendizado semi-supervisionado combina características dos dois anteriores, utilizando uma pequena quantidade de dados rotulados juntamente com uma grande quantidade de dados não rotulados para melhorar o treinamento do modelo (ENGELEN; HOOS, 2020). Esse tipo de aprendizado é empregado quando a rotulagem de dados é cara ou demorada (ENGELEN; HOOS, 2020). Técnicas populares incluem *Self-training*, que permite ao modelo treinar iterativamente usando seus próprios rótulos preditivos, e *Graph-based methods*, que utilizam relações entre os dados para propagar rótulos (CHAPELLE; SCHÖLKOPF; ZIEN, 2006).

No aprendizado por reforço, o modelo aprende a tomar decisões através de tentativa e erro, visando maximizar uma recompensa definida. É utilizado em aplicações como robótica e jogos, com técnicas populares como o Q-Learning e as redes neurais profundas de reforço (Deep Q-Networks). (JANIESCH; ZSCHECH; HEINRICH, 2021).

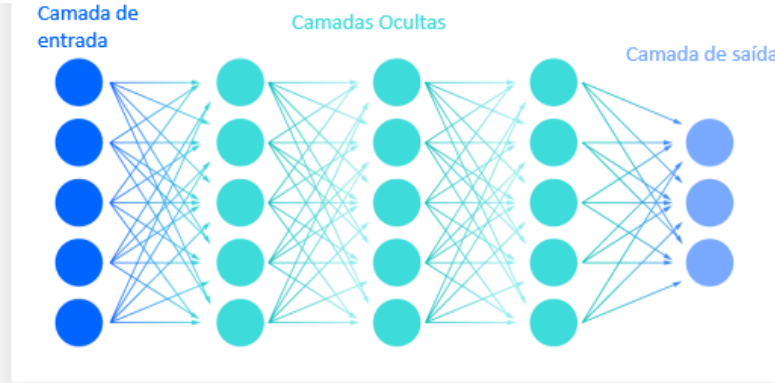
Uma das técnicas de aprendizado de máquina é *Deep Learning*. Ela se destaca pela sua capacidade de aprender padrões dos dados fornecidos de forma independente, o que elimina a necessidade de extração manual de características dos dados (LECUN; BENGIO; HINTON, 2015). No contexto do trabalho atual, o tipo de aprendizado utilizado é o supervisionado, onde o modelo irá aprender a partir de dados rotulados, com a finalidade de ser capaz de classificar imagens de RBC a partir de amostras pré-classificadas.

2.1.2 *Deep Learning*

A base de *Deep Learning*, são as redes neurais artificiais. Essas se inspiram no cérebro humano, com neurônios artificiais (nós) interconectados em três tipos de camadas principais: entrada, oculta e saída, como mostrado na Figura 2.1. A camada de entrada são os dados que serão utilizados. As camadas ocultas, que podem ser múltiplas, computam os dados recebidos da camada de entrada ou de outras camadas ocultas. A camada de saída fornece o resultado final do processamento, podendo ter um ou vários nós conforme a complexidade do problema. As

redes simples têm três camadas, enquanto as redes profundas possuem várias camadas ocultas e milhões de neurônios (LECUN; BENGIO; HINTON, 2015).

Figura 2.1 – Exemplo de uma rede neural.



Fonte: adaptado de (IBM, 2024).

Cada neurônio (j) em uma camada (l) recebe valores entradas, após isso é realizado uma soma ponderada dessas entradas com os pesos associados a elas, adiciona um viés (*bias*) e aplica-se uma função de ativação ao resultado (LECUN; BENGIO; HINTON, 2015). Este processo é descrito pela seguinte expressão matemática (LECUN; BENGIO; HINTON, 2015) :

$$z_j^{(l)} = \sum_{i=1}^n w_{ij}^{(l)} x_i^{(l-1)} + b_j^{(l)}, \quad (2.1)$$

onde:

- $z_j^{(l)}$ é a soma ponderada do neurônio j na camada l ,
- $w_{ij}^{(l)}$ é o peso associado à conexão entre o neurônio i na camada $(l - 1)$ e o neurônio j na camada l ,
- $x_i^{(l-1)}$ é a saída do neurônio i na camada $(l - 1)$,
- $b_j^{(l)}$ é o viés (*bias*) do neurônio j na camada l .

Com a soma ponderada dos pesos, aplica-se uma função de não-linearidade. Esta operação pode ser definida pela seguinte equação:

$$a_j^{(l)} = f(z_j^{(l)}), \quad (2.2)$$

onde:

- $f(\cdot)$ é a função de ativação aplicada à soma ponderada $z_j^{(l)}$,
- $a_j^{(l)}$ é a saída (ativação) do neurônio j na camada l .

As funções de ativação são basicamente o processamento da soma em cada neurônio, e sua escolha varia com a natureza do problema e as características dos dados trabalhados (LECUN; BENGIO; HINTON, 2015). Entre as opções comuns de funções de ativação estão a linear, que mantém a saída proporcional à entrada; a sigmoide, que comprime a saída entre 0 e 1 (adequada para problemas de classificação binária); a ReLU (*Rectified Linear Unit*), que é não linear e estabelece a saída como zero para entradas negativas, ajudando a rede a aprender características mais complexas; e a tangente hiperbólica (Tanh), semelhante à sigmoide, mas comprimindo a saída entre -1 e 1 (LECUN; BENGIO; HINTON, 2015).

Por outro lado, a função de perda é vital para avaliar o desempenho da rede, medindo a diferença entre as previsões e os valores reais (WANG et al., 2020). Existem algumas opções para a função de perda, com isso, o importante é avaliar o modelo trabalhado e o objetivo desejado, para que os resultados sejam positivamente influenciados. O Erro Quadrático Médio (*Mean Squared Error* - MSE) é comumente usado em problemas de regressão, enquanto a Entropia Cruzada (*Cross-Entropy Loss*) é eficaz em problemas de classificação (LECUN; BENGIO; HINTON, 2015). Reduzir a perda é essencial para melhorar a precisão do modelo, ajustando os parâmetros da rede para que suas previsões se aproximem cada vez mais dos valores reais (WANG et al., 2020). Esse ajuste é feito através do processo de retropropagação (*backpropagation*).

2.1.3 Backpropagation

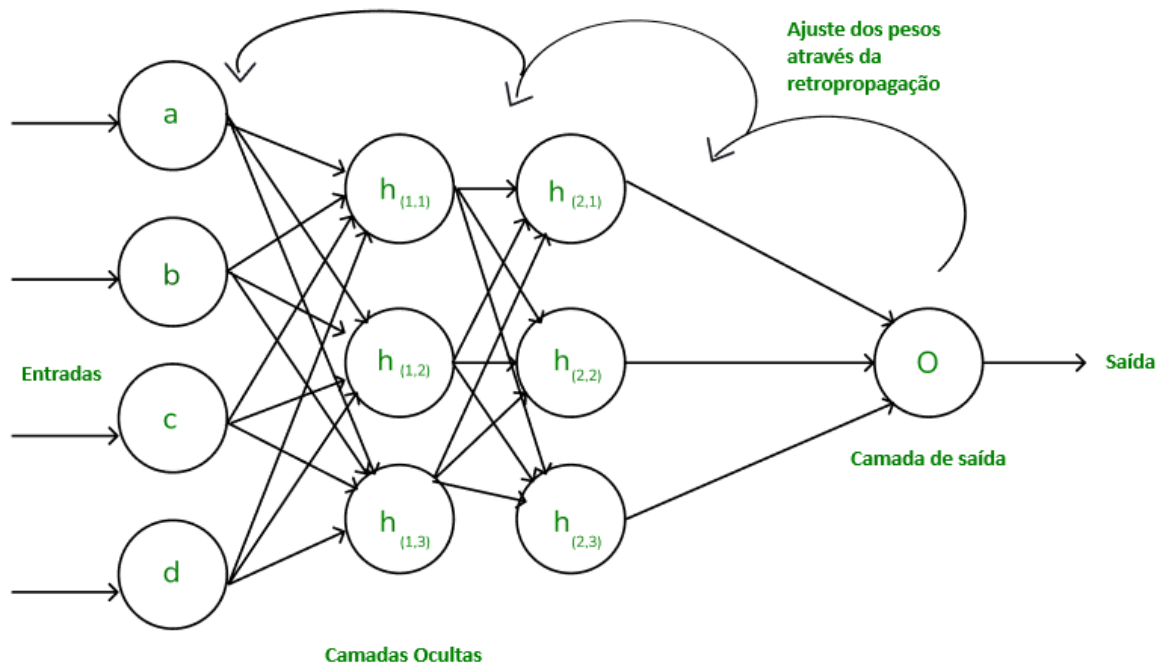
O algoritmo de *backpropagation* (ROJAS; ROJAS, 1996) é empregado para calcular o gradiente da função de perda em relação aos pesos da rede, permitindo que esses pesos sejam ajustados para minimizar a perda. Isso é realizado usando uma técnica de otimização, como o gradiente descendente. A Figura 2.2 ilustra o processo de *backpropagation*, mostrando como o erro é propagado através das camadas da rede e como os pesos são ajustados em resposta a esse erro.

O *learning rate*, representado por η , controla a magnitude do ajuste dos pesos a cada iteração, ou seja, define o quão rapidamente ou lentamente o modelo corrige os seus pesos. Um *learning rate* muito alto pode fazer com que o modelo aprenda de forma muito rápida, possivelmente saltando por cima de mínimos locais e não convergindo adequadamente. Por outro lado, um *learning rate* muito baixo pode tornar o processo de treinamento extremamente lento e prender o modelo em mínimos locais, sem alcançar a solução ideal (ROJAS; ROJAS, 1996). Portanto, a escolha de um *learning rate* apropriado é crucial para a eficiência e eficácia do treinamento da rede neural e sua aplicação na atualização dos pesos é representada pela seguinte fórmula (ROJAS; ROJAS, 1996):

$$w_{\text{new}} = w_{\text{old}} - \eta \times \nabla L \quad (2.3)$$

onde:

Figura 2.2 – Representação de como o *backpropagation* funciona através do ajuste de pesos.



Fonte: adaptado de (GeeksforGeeks, 2024).

- w_{new} é o novo peso,
- w_{old} é o peso anterior,
- n é o *learning rate*,
- ∇L é o gradiente da função de perda em relação aos pesos.

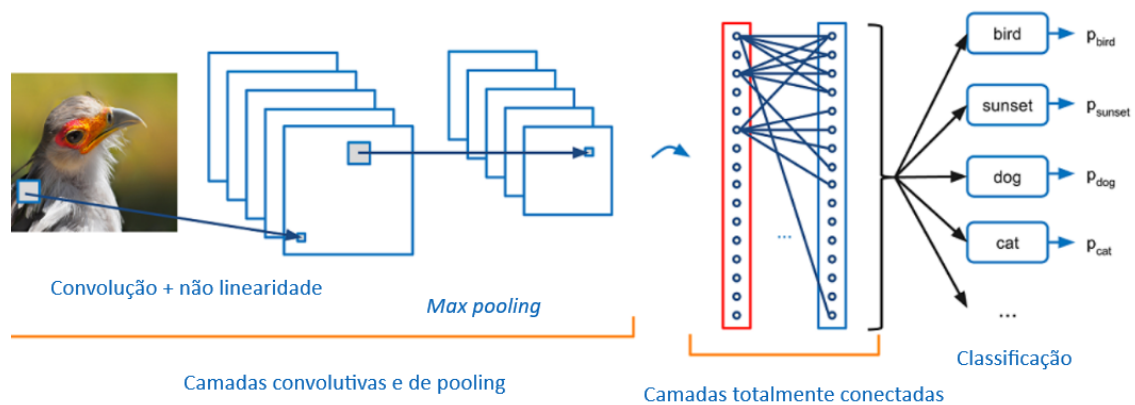
2.1.4 Convolutional Neural Network (CNN)

As Redes Neurais Convolucionais (*Convolutional Neural Network (CNN)*) são um tipo especial de rede neural desenvolvidas para processar imagens (O'SHEA; NASH, 2015). Elas têm causado um grande impacto na área de visão computacional, sendo essenciais em diversas aplicações, desde o reconhecimento de objetos (SHEHZADI et al., 2024) até a segmentação de imagens (YAO et al., 2024) e detecção de rostos (KANNA et al., 2024). O que torna as CNNs tão poderosas é o uso de filtros convolucionais para identificar características específicas das imagens em diferentes partes e escalas (O'SHEA; NASH, 2015).

Outro ponto importante é que as CNNs podem ter várias camadas convolucionais e totalmente conectadas em sequência. À medida que a informação é passada pela rede, essas camadas extraem características cada vez mais complexas, permitindo que a rede aprenda representações hierárquicas dos dados (POLONI, 2022).

A camada convolutiva em uma **CNN** consiste em um conjunto de filtros convolucionais (*kernel*) que são aplicados à entrada para extrair características. Cada filtro é uma matriz de pesos que desliza (ou convolui) sobre a entrada, realizando operações de multiplicação entre os valores dos pixels da entrada e os pesos do filtro em cada posição. O resultado dessa operação é então somado para produzir um único valor na saída, que representa a ativação do filtro naquela posição da entrada (KRICHEN, 2023). Esses filtros são como pequenas janelas que percorrem a imagem, destacando bordas, texturas e formas importantes.

Figura 2.3 – Exemplo de um processo realizado por uma **CNN**.



Fonte: adaptado de (BOOK, 2024).

A principal função da camada de *pooling* em uma **CNN** é reduzir a dimensionalidade dos mapas de características Krichen (2023). Isso é feito dividindo a entrada em regiões sobrepostas e aplicando uma operação de agregação em cada região. A operação de *pooling* mais comum é o *max pooling*, onde o valor máximo em cada região é selecionado e mantido na saída, descartando os outros valores. Outra opção é o *average pooling*, onde a média dos valores da região é calculada. O tamanho da região de *pooling* e o passo (*stride*) determinam a quantidade de sobreposição entre as regiões e a redução na dimensionalidade. Essa redução de dimensionalidade torna a representação dos dados mais compacta e eficiente computacionalmente, ao mesmo tempo que preserva as características mais importantes da entrada.

Além das camadas convolucionais e de *pooling*, uma **CNN** também pode incluir outras camadas importantes para melhorar o desempenho e a capacidade de generalização do modelo. A camada de *batch normalization* desempenha um papel fundamental na normalização da saída das camadas anteriores, reduzindo a covariância interna e melhorando a estabilidade do treinamento (KRICHEN, 2023). A camada de *flatten* é essencial para transformar os mapas de características tridimensionais em um vetor unidimensional, preparando-os para alimentar as camadas totalmente conectadas. Por fim, as camadas totalmente conectadas recebem os vetores de características e realizam as operações finais de processamento para gerar as previsões finais da rede (KRICHEN, 2023).

Com o impacto significativo das **CNNs** no campo da visão computacional, várias arqui-

teturas conhecidas foram desenvolvidas com base nesses princípios (RIBANI; MARENGONI, 2019). Um exemplo é a EfficientNet, que otimiza a eficiência computacional e o desempenho através de técnicas de escalonamento uniforme da profundidade, largura e resolução da rede (TAN; LE, 2019). Essas inovações permitiram criar modelos que desempenham positivamente, ampliando a aplicação das CNNs em diversas áreas da inteligência artificial.

2.1.5 *Overfitting*

O *overfitting*, ou sobreajuste, ocorre quando um modelo se ajusta excessivamente aos dados de treinamento, capturando não apenas os padrões reais, mas também o ruído e as variações aleatórias presentes no conjunto de dados (CANDIDO, 2023), assim o modelo se adapta “perfeitamente” ao conjunto de treinamento, como visto na Figura 2.4. Embora isso possa resultar em um desempenho excelente durante o treinamento, o modelo geralmente apresenta uma capacidade de generalização ruim quando exposto a novos dados, ou seja, ele falha em fazer previsões precisas em amostras não vistas (CANDIDO, 2023).

Figura 2.4 – Exemplo de como o modelo pode se adaptar aos dados de teste.



Fonte: (CANDIDO, 2023).

Esse fenômeno é comum quando o modelo é excessivamente complexo, com muitas camadas ou neurônios, permitindo que ele memorize o conjunto de treinamento ao invés de identificar padrões generalizáveis (CANDIDO, 2023). Como resultado, o desempenho nos dados de validação ou de teste tende a ser significativamente inferior ao obtido no treinamento, o que é um claro sinal de *overfitting*.

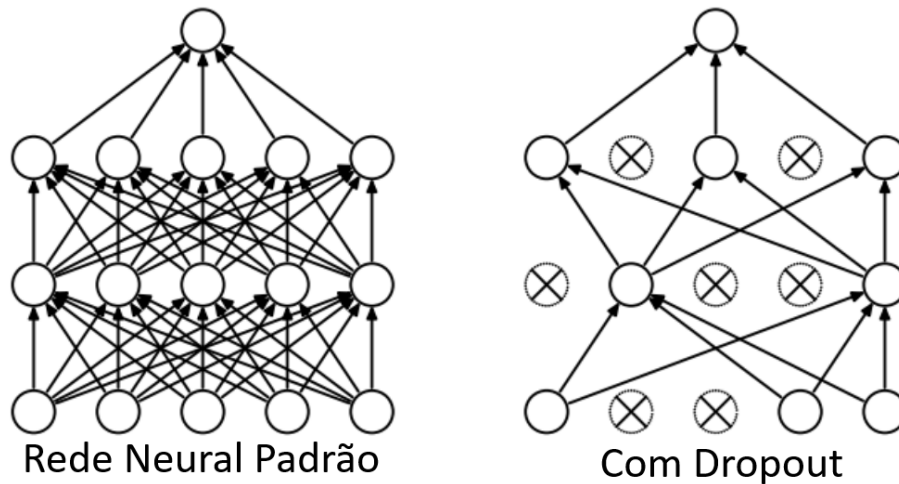
Para mitigar o *overfitting*, diversas técnicas podem ser aplicadas, como a redução da complexidade do modelo, o uso de mais dados de treinamento ou a aplicação de métodos de regularização, como o *Dropout*. Esses métodos forçam o modelo a aprender padrões mais robustos e generalistas, ao invés de memorizar os exemplos específicos do treinamento (BUDHIRAJA, 2016). Além disso, técnicas de validação cruzada podem ser usadas para avaliar a capacidade do modelo de generalizar antes mesmo de ser testado em um conjunto de dados separado.

2.1.6 *Dropout*

Para combater o risco de *overfitting* em modelos de aprendizado profundo, uma técnica de regularização conhecida como *Dropout* é uma possibilidade a ser aplicada (BUDHIRAJA, 2016).

Esta técnica consiste em desativar aleatoriamente uma proporção específica de neurônios em cada etapa do treinamento, como pode ser visto na [Figura 2.5](#). Ao fazer isso, a técnica de *Dropout* impede que o modelo dependa excessivamente de qualquer neurônio individual ou configuração específica de neurônios, promovendo assim uma aprendizagem mais robusta ([BUDHIRAJA, 2016](#)).

Figura 2.5 – Processo de funcionamento da técnica de *Dropout*.



Fonte: adaptado de ([BUDHIRAJA, 2016](#)).

Normalmente, o uso da técnica de *Dropout* aumenta a capacidade de generalização da rede para novos dados, não apenas replicando padrões observados no conjunto de treinamento. Esta abordagem foi destacada em estudos como o de [Budhiraja \(2016\)](#), que ressaltam a eficácia da técnica de *Dropout* na melhoria do desempenho dos modelos em cenários de aplicação real, onde os dados podem apresentar certas variações em relação ao conjunto de dados de treinamento.

2.1.7 *Data Augmentation*

A técnica de *Data Augmentation* consiste em gerar sinteticamente imagens à partir das imagens originais da base de dados, de tal forma a aumentar o tamanho e a diversidade do conjunto de dados de treinamento ([SHORTEN; KHOSHGOFTAAR, 2019](#)). Entre as operações mais comuns estão rotações, espelhamentos, alterações de brilho, recortes, adições de ruído, entre outras, alguns exemplos podem ser vistos na [Figura 2.6](#).

O principal objetivo da técnica de *Data Augmentation* é aumentar a diversidade de dados buscando melhorar a capacidade de generalização dos modelos ([SHORTEN; KHOSHGOFTAAR, 2019](#)). Ela também pode ser utilizada para evitar *overfitting*, já que em muitas aplicações há uma limitação na quantidade de dados disponíveis para treinamento, o que eventualmente pode facilitar o sobre-ajuste do modelo a estes dados. Ao introduzir variações no conjunto de dados, a técnica ajuda a simular uma maior variedade de possibilidades que podem ser encontradas no mundo real, promovendo um aprendizado mais eficiente em alguns casos ([SHORTEN; KHOSHGOFTAAR, 2019](#)).

Figura 2.6 – Exemplo do resultado da aplicação da técnica de *Data Augmentation*. Em (a) está a imagem original, em (b) foi aplicada uma rotação de 180°, em (c) a conversão para escala de cinza, em (d) um zoom-in na imagem e em (e) a pixelização combinada com um espelhamento horizontal.



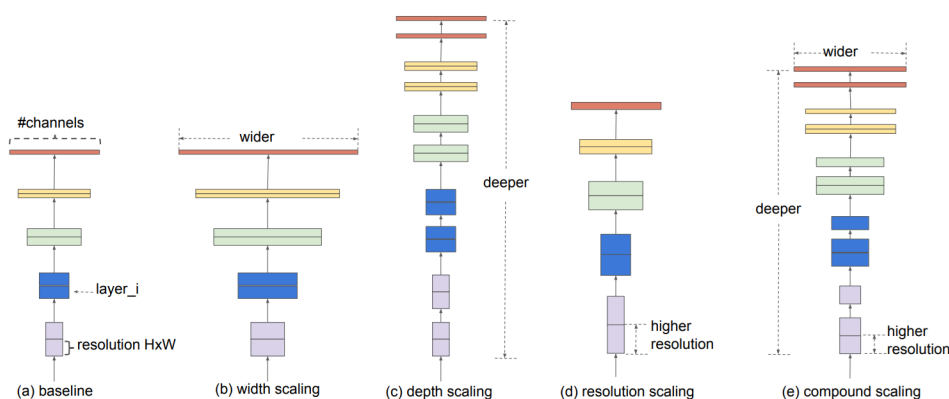
Fonte: (datacamp, 2022).

Portanto, a aplicação da técnica de *Data Augmentation* tende melhorar o desempenho de modelos, como no caso do estudo de [Minarno et al. \(2023\)](#), onde a utilização da técnica impactou positivamente para os resultados obtidos, todavia, não é possível afirmar que esse ganho sempre ocorrerá. Nesse trabalho, também será aplicado a técnica com o intuito de tentar aprimorar os resultados obtido pelo modelo proposto.

2.1.8 Família das arquiteturas EfficientNet

Conforme descrito por [Tan e Le \(2019\)](#), EfficientNet representa uma família de arquiteturas de redes neurais convolucionais. A grande inovação por trás do EfficientNet é a busca por um equilíbrio ótimo entre a complexidade do modelo e sua eficácia. Ao contrário das abordagens anteriores, que tendiam a aumentar a profundidade ou a largura da rede para melhorar a precisão, o EfficientNet introduziu o conceito de escala composta. Essa estratégia envolve o ajuste simultâneo de três dimensões-chave: largura, profundidade e resolução de entrada, buscando encontrar um conjunto de parâmetros que otimize essas dimensões e resulte em um modelo eficiente e preciso, como demonstrado na [Figura 2.7](#).

Figura 2.7 – Estratégias de Escalonamento na Arquitetura EfficientNet.



Fonte: (TAN; LE, 2019).

O escalonamento de largura (*width scaling*) implica em ajustar o número de canais em

cada camada da rede, enquanto o escalonamento de profundidade (*depth scaling*) está relacionado ao número total de camadas. Por fim, o escalonamento de resolução (*resolution scaling*) refere-se ao tamanho da entrada da imagem. O método de escala composta, por sua vez, adota uma técnica conhecida como redes escaláveis de profundidade e largura (*compound scaling*), que proporcionalmente aumenta a largura e a profundidade do modelo, mantendo uma taxa de crescimento constante. Além disso, a resolução de entrada é ajustada para se adequar ao aumento da complexidade (SARKAR, 2021).

Adicionalmente, o EfficientNet se beneficia de técnicas como regularização e ajuste de hiper-parâmetros, que auxiliam na prevenção de *overfitting* e na melhoria do desempenho geral do modelo (TAN; LE, 2019). A arquitetura EfficientNet-B0 se mostrou eficaz para tarefas de classificação e detecção de imagens (TAN; LE, 2019). Considerando que este trabalho está relacionado a essa área, o uso dessa arquitetura é especialmente relevante para tentar alcançar resultados satisfatórios na classificação de malária em células sanguíneas. Além disso, a escolha do EfficientNet-B0 é reforçada pela escassez de recursos computacionais, já que ele oferece um equilíbrio entre precisão e custo computacional (TAN; LE, 2019), permitindo obter bom desempenho mesmo em ambientes com limitações de hardware.

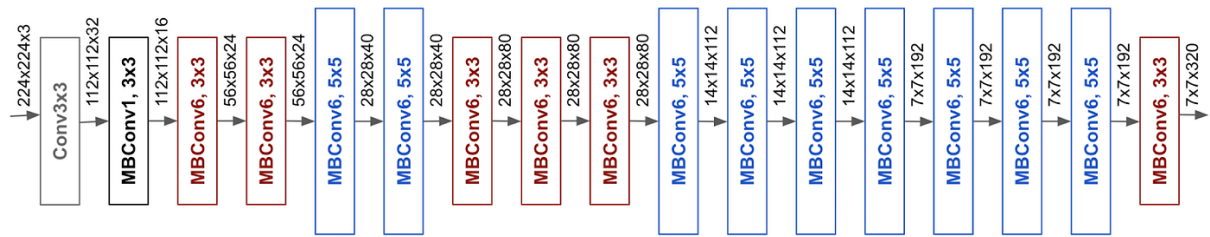
EfficientNet-B0

A EfficientNet-B0 é parte de uma família de CNNs, as quais foram criadas por meio do método de escalonamento composto com o objetivo de melhorar a precisão dos modelos (TAN; LE, 2019). Esse método de escalonamento composto realiza uma ampliação uniforme em todas as dimensões de largura, profundidade e resolução, utilizando um coeficiente composto (TAN; LE, 2019). Isso permite que a rede atinja um equilíbrio entre precisão e eficiência computacional, o que é especialmente importante para a implantação em dispositivos com pouco poder computacional e outras aplicações com recursos limitados (TAN; LE, 2019).

A arquitetura da EfficientNet-B0 é derivada dos componentes presentes na MobileNetV2, com destaque para o *Mobile Inverted Bottleneck Conv* (MBCONV) (SANDLER et al., 2018) e a adição do bloco de *Squeeze and Excitation* (SE) (HU; SHEN; SUN, 2018). Esses componentes são importantes para o desempenho da rede, pois permitem manter uma boa precisão com um número mínimo de parâmetros, o que contribui para uma rede leve e eficaz (TAN; LE, 2019). Além disso, a combinação do MBCONV com o bloco SE otimiza o uso de recursos computacionais (HU; SHEN; SUN, 2018) (SANDLER et al., 2018).

A Figura 2.8 ilustra a disposição das camadas na EfficientNet-B0. Dado a capacidade da EfficientNet-B0 em alcançar níveis promissores de precisão mantendo eficiência computacional (TAN; LE, 2019), é natural explorar a aplicação de TL utilizando essa arquitetura. O uso de pesos pré-treinados pode acelerar o processo de treinamento e melhorar ainda mais o desempenho em tarefas específicas, como a classificação de malária em imagens de esfregaço de sangue.

Figura 2.8 – Arquitetura base da rede EfficientNet-B0.

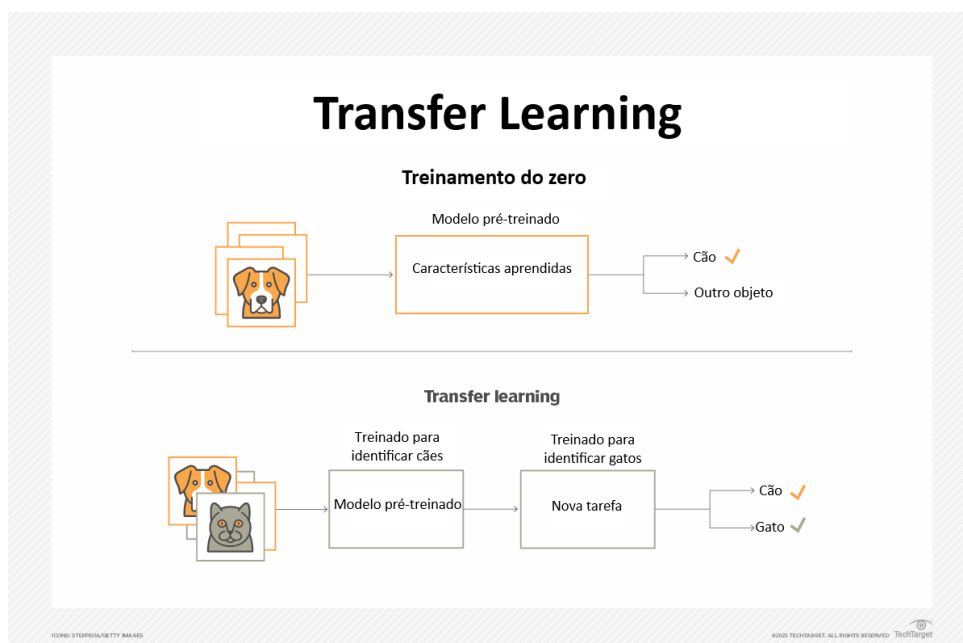


Fonte: (MONTALBO; D.ENG, 2021).

2.1.9 Transfer Learning

Transfer Learning é uma técnica fundamental e amplamente aplicada no campo do aprendizado de máquina que consiste em aproveitar os conhecimentos adquiridos por modelos pré-treinados em grandes conjuntos de dados e aplicá-los a novas tarefas (RIBANI; MARENGONI, 2019). Isso é possível ajustando os pesos de uma rede neural pré-treinada, para que se adequem às peculiaridades da nova tarefa. Durante o treinamento, os pesos das camadas iniciais são geralmente congelados, enquanto as camadas finais são ajustadas para se adaptarem aos novos dados.

Figura 2.9 – Exemplo do funcionamento de *Transfer Learning*.



Fonte: adaptado de (GILLIS, 2024).

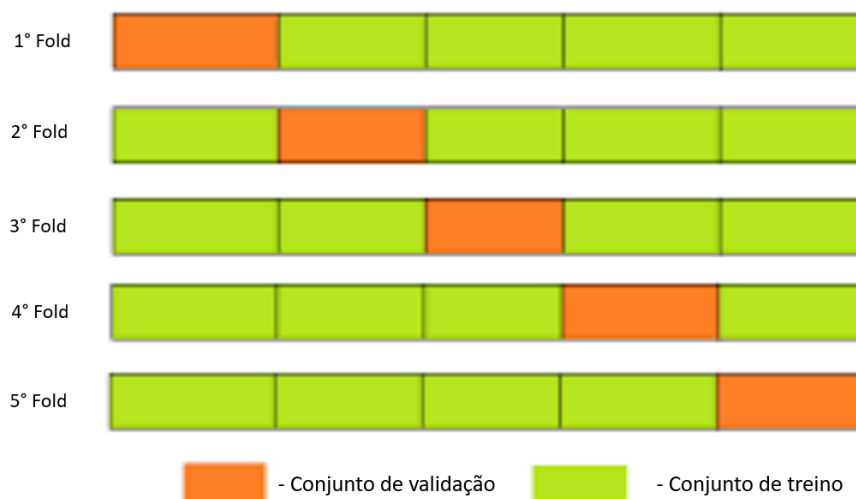
Essa estratégia é especialmente útil pois permite que o modelo se beneficie do conhecimento já adquirido em conjuntos de dados maiores. Essa técnica não só acelera o treinamento, como também tende a melhorar a generalização do modelo para novos dados, tornando-o mais robusto e preciso (RIBANI; MARENGONI, 2019). Algumas das arquiteturas comumente utilizadas em *Transfer Learning* incluem, ResNet (*Residual Network*) (REDDY; JULIET, 2019), Inception (GoogLeNet) Minarno et al. (2023) e EfficientNet (TAN; LE, 2019). Cada uma dessas

arquitecturas possui características específicas que as tornam adequadas para diferentes tipos de tarefas, conjuntos de dados e dos recursos computacionais disponíveis (NARAYANAN; ALI; HARDIE, 2019).

2.1.10 *StratifiedKFold*

No processo do *K-fold*, primeiro, o conjunto de dados é dividido em k *folds* de tamanhos aproximadamente iguais. Após isso, O modelo é treinado k vezes, cada vez utilizando $k-1$ *folds* para treinamento e o *fold* restante para teste. Em cada iteração, um *fold* diferente é usado como conjunto de teste como visto na Figura 2.10. Por fim, os resultados de todas as iterações são combinados para fornecer uma estimativa final do desempenho do modelo (PRUSTY; PATNAIK; DASH, 2022).

Figura 2.10 – Figura que ilustra o processo de validação cruzada *K-fold*.



Fonte: adaptado de (Velog, 2023).

O *StratifiedKFold* é uma variação da validação cruzada *k-fold* que estratifica os dados antes de dividi-los em k -*folds* (subconjuntos) (PRUSTY; PATNAIK; DASH, 2022). Isso significa que cada *fold* terá aproximadamente a mesma proporção de classes que o conjunto de dados original. Este método é particularmente útil em cenários onde as classes estão desbalanceadas, ou seja, quando uma ou mais classes são muito mais frequentes que outras (PRUSTY; PATNAIK; DASH, 2022).

Assim, *StratifiedKFold* oferece várias vantagens significativas na validação cruzada de modelos de aprendizado de máquina. Primeiramente, garante que cada *fold* mantenha a mesma distribuição das classes do conjunto de dados original, o que é crucial em cenários com desbalanceamento. Além disso, reduz a variabilidade dos resultados, proporcionando uma avaliação mais estável e confiável do desempenho do modelo. Isso assegura que as métricas

de performance obtidas sejam consistentes e menos sujeitas a flutuações aleatórias (PRUSTY; PATNAIK; DASH, 2022).

2.1.11 Cosine Decay

A técnica de *Cosine Decay* é utilizada em aprendizado profundo para ajustar a taxa de aprendizado ao longo do tempo de treinamento de um modelo (CORREA, 2019). Essa técnica permite um controle dinâmico da taxa de aprendizado, partindo de um valor inicial maior que possibilita ajustes maiores nos pesos do modelo, e gradualmente diminuindo até um valor mínimo, conforme o treinamento se aproxima do fim (CORREA, 2019). Isso é útil para equilibrar a exploração do espaço de soluções e a convergência, evitando que o modelo fique preso em mínimos locais e, ao mesmo tempo, ajudando a reduzir o overfitting.

O decaimento segue uma curva cossenoide definida pela seguinte equação:

$$\eta(t) = \eta_0 \times \frac{1}{2} \left(1 + \cos \left(\frac{t}{T} \pi \right) \right). \quad (2.4)$$

Na Equação (2.4), η_0 representa a taxa de aprendizado inicial, t é o passo de treinamento atual, e T define o número total de passos de decaimento. Opcionalmente, um parâmetro α pode ser utilizado para ajustar o valor final da taxa de aprendizado, garantindo que ela diminua até um limite mínimo estabelecido:

$$\eta_{final} = \alpha \times \eta_0. \quad (2.5)$$

Esse mecanismo permite que o treinamento inicie com atualizações maiores nos pesos, seguidas de ajustes progressivamente menores, favorecendo tanto a exploração quanto a estabilidade na convergência final do modelo (CORREA, 2019).

O *pipeline* aplicado nesse estudo implementa a técnica de *Cosine Decay* conforme o pseudo-código apresentado no [algoritmo 2.1](#).

Algoritmo 2.1: Definição da Taxa de Aprendizado com *Cosine Decay*.

Entrada : Taxa de aprendizado inicial ($lr_{inicial}$);
 Taxa de aprendizado final (lr_{final});
 Número de épocas (*epochs*);
 Tamanho do lote (*batch_size*);
 Número de exemplos (n).

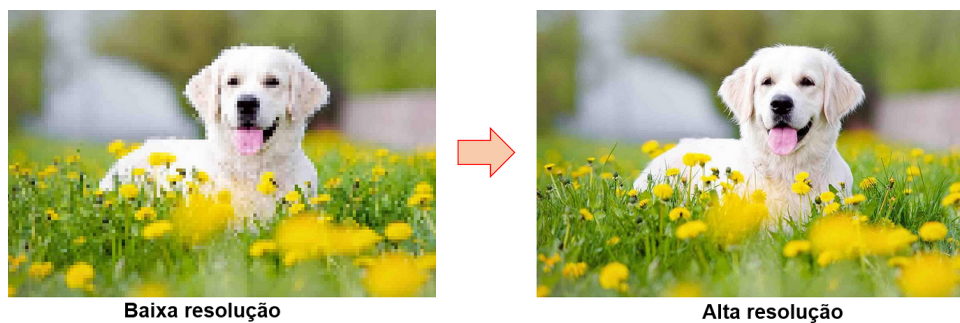
Saída : Agenda de taxa de aprendizado para treinamento

- 1 **Calcular** o fator de decaimento $\alpha = \frac{lr_{final}}{lr_{inicial}}$
 - 2 **Calcular** o número total de passos de decaimento $decay_steps = epochs \times \frac{n}{batch_size}$
 - 3 **Definir** uma função de decaimento de taxa de aprendizado usando a **CosineDecay**
 - 4 **Inicializar** a função de decaimento com os parâmetros: $lr_{inicial}$, $decay_steps$, α
 - 5 **return** Agenda da taxa de aprendizado (*learning rate schedule*) gerada
-

2.1.12 Super Resolution

A técnica de **SR** refere-se ao processo de aumentar a resolução de uma imagem, aprimorando sua qualidade e nível de detalhe, como pode ser observado na Figura 2.11 (WANG et al., 2018). Esse aumento de resolução é útil em diversas aplicações, como reconhecimento de padrões, análise médica, vigilância por vídeo e aprimoramento de imagens de baixa qualidade em *deep learning* (WANG et al., 2018).

Figura 2.11 – Ilustração do processo de *Super Resolution*.



Fonte: adaptado de (KUNDU, 2022).

O principal objetivo da técnica de **SR** é transformar imagens de baixa resolução em versões aprimoradas que se aproximem de imagens de alta resolução. Essa técnica é especialmente valiosa em cenários onde a captura de imagens de alta resolução é inviável ou onde a melhoria da resolução é essencial para análises mais precisas (WANG et al., 2018).

Entre as abordagens mais comuns para **SR**, destacam-se:

- **Métodos Baseados em Interpolação:** Técnicas tradicionais, como interpolação bilinear e bicúbica, utilizam a média dos *pixels* vizinhos para estimar novos valores e aumentar a resolução da imagem. Embora sejam métodos simples e rápidos, muitas vezes resultam em perda de nitidez e introdução de artefatos, não capturando detalhes finos da imagem (WANG et al., 2018).
- **Métodos Baseados em Aprendizado Profundo:**
 - **CNN:** Modelos como o *Super-Resolution Convolutional Neural Network* (SRCNN) utilizam redes neurais que aprendem a mapear diretamente imagens de baixa resolução para versões de alta resolução. Durante o treinamento, a rede identifica padrões e características que ajudam a reconstruir detalhes perdidos, na tentativa de proporcionar melhorias na qualidade da imagem (DONG et al., 2015).
 - **Generative Adversarial Networks (GAN):** Modelos como o *Super-Resolution Generative Adversarial Network* (SRGAN) envolvem duas redes neurais competindo entre si: uma geradora, que cria imagens de alta resolução a partir de entradas de baixa resolução, e uma discriminadora, que avalia a autenticidade das imagens geradas.

Esse processo adversarial resulta em imagens de alta resolução com maior qualidade perceptual e detalhes mais realistas (LEDIG et al., 2017).

- **Transformers**: Modelos mais recentes, como o SwinIR, utilizam a arquitetura de *Transformers*, originalmente desenvolvida para processamento de linguagem natural, adaptada para imagens. Esses modelos capturam relações de longo alcance e contextos amplos dentro da imagem, permitindo a reconstrução de detalhes finos e texturas complexas (LIANG et al., 2021).

Embora a técnica de SR tenha avançado significativamente, desafios persistem, como a preservação de detalhes mais finos. A busca por modelos mais eficientes e capazes de generalizar melhor na reconstrução de diferentes tipos de imagens continua sendo uma área ativa de pesquisa (WANG et al., 2018).

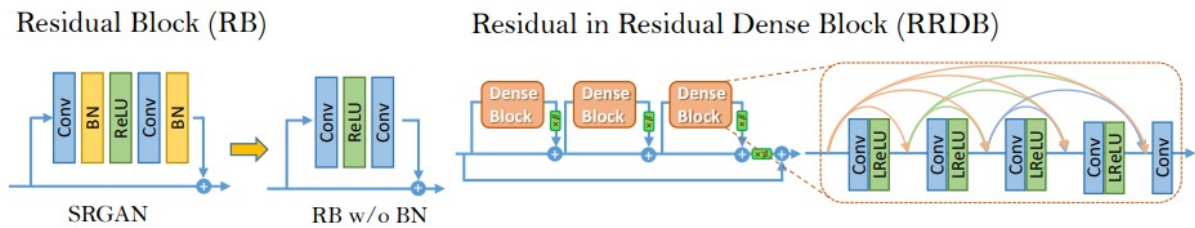
Nos últimos anos, um dos modelos mais eficazes para SR tem sido o *Enhanced Super-Resolution Generative Adversarial Network* (ESRGAN), que aprimora a abordagem do SRGAN ao fornecer imagens de alta qualidade com mais detalhes e texturas realistas (WANG et al., 2018). A próxima seção abordará o ESRGAN em mais detalhes, explorando seu funcionamento e aplicação no presente trabalho.

2.1.13 *Enhanced Super-Resolution Generative Adversarial Network (ESRGAN)*

O *Enhanced Super-Resolution Generative Adversarial Network* (ESRGAN) é uma evolução do modelo SRGAN, originalmente proposto por Ledig et al. (2017), e foi desenvolvido com o objetivo de aprimorar a qualidade das imagens reconstruídas. A principal motivação para essa evolução surgiu da limitação do SRGAN em preservar detalhes finos, assim produzindo artefatos visuais indesejados (LEDIG et al., 2017; WANG et al., 2018).

Conforme evidenciado por Wang et al. (2018), o ESRGAN introduz melhorias na arquitetura da rede geradora, substituindo os blocos residuais convencionais por uma estrutura denominada *Residual-in-Residual Dense Block* (RRDB). Essa modificação elimina as camadas de normalização em lote, como visto na Figura 2.12, uma vez que, segundo Wang et al. (2018), podem introduzir artefatos e limitar o desempenho da rede. O bloco RRDB permite que o modelo integre múltiplos níveis de aprendizado residual com conexões densas, favorecendo a propagação de gradientes em redes profundas e contribuindo para uma maior estabilidade do treinamento (WANG et al., 2018).

Além disso, o ESRGAN adota estratégias complementares como o *residual scaling* e a inicialização com valores reduzidos, com o intuito de evitar instabilidades durante o treinamento, sobretudo em redes com grande profundidade (WANG et al., 2018). Outro diferencial importante está na função de perda utilizada, que incorpora uma perda perceptiva modificada, baseada em

Figura 2.12 – Estrutura interna do bloco *Residual-in-Residual Dense Block*.

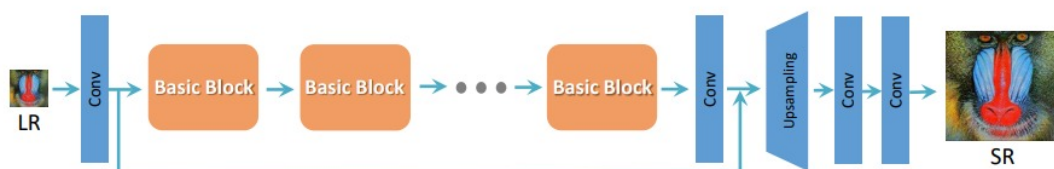
Fonte: (WANG et al., 2018).

características extraídas da rede VGG-19 antes da ativação, proporcionando uma supervisão mais sensível à preservação de brilho, texturas e detalhes mais finos (WANG et al., 2018).

O modelo também inclui uma variante da perda perceptiva baseada na rede MINC-VGG, treinada com o conjunto de dados *Materials in Context Database*, com foco em características texturais em vez de estruturais (BELL et al., 2015). Essas adaptações contribuem para melhorar a fidelidade visual das imagens geradas, especialmente em aplicações que demandam detalhes mais refinados (WANG et al., 2018). Outro ponto é a substituição do discriminador tradicional por um discriminador relativo médio *Relativistic average Discriminator* (RaGAN), proposto por Jolicoeur-Martineau (2018). Essa abordagem compara a probabilidade de uma imagem real ser mais realista que uma imagem sintética, permitindo que o gerador receba gradientes mais informativos e produza resultados visualmente superiores (WANG et al., 2018; JOLICOEUR-MARTINEAU, 2018).

A Figura 2.13 ilustra a arquitetura geral do ESRGAN, que recebe como entrada uma imagem em baixa resolução e a reconstrói em alta resolução por meio de um pipeline composto por camadas convolucionais, onde blocos básicos são módulos de *upsampling*, ou seja, responsáveis por realizar o aumento da resolução espacial da imagem, ampliando suas dimensões para gerar a saída em alta resolução. Essa etapa possibilita ao modelo converter as representações extraídas nos blocos anteriores em uma imagem final mais detalhada (WANG et al., 2018). Cada bloco básico corresponde ao componente RRDB, cuja estrutura detalhada é mostrada na Figura 2.12.

Figura 2.13 – Arquitetura do modelo ESRGAN.



Fonte: (WANG et al., 2018).

Dado o desempenho expressivo alcançado pelo ESRGAN, o modelo apresenta-se na literatura como uma abordagem eficaz para tarefas de SR, especialmente em cenários que exigem

elevada fidelidade visual (WANG et al., 2018). Considerando esses aspectos, o ESRGAN será adotado neste trabalho como modelo base para a realização de SR no conjunto de dados utilizado.

2.1.14 Vision Transformer (ViT)

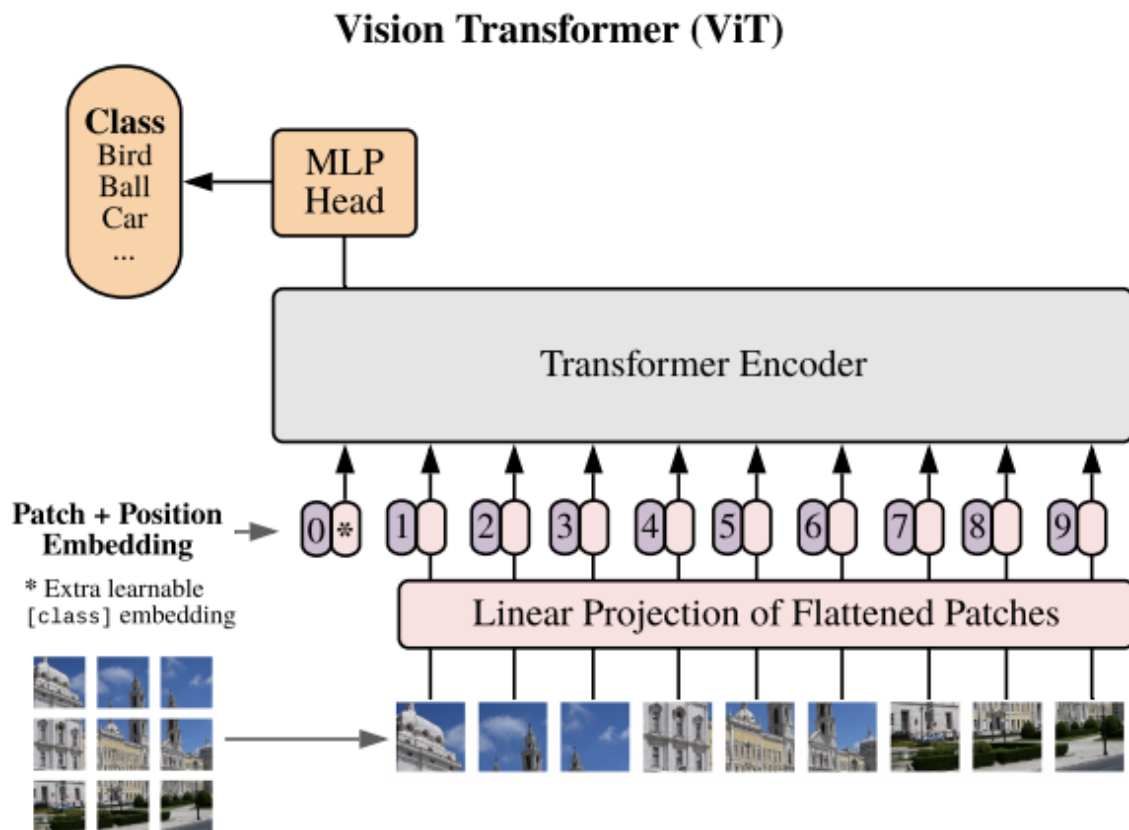
Os *Transformers* recentemente vêm sendo utilizados em diversas áreas da IA, principalmente em processamento de linguagem natural (VASWANI et al., 2017). No entanto, com a introdução do ViT, essa arquitetura foi adaptada para visão computacional (DOSOVITSKIY et al., 2020). O ViT demonstrou desempenho comparável e, em alguns casos, superior às CNNs em tarefas como classificação de imagens e detecção de objetos, especialmente em grandes conjuntos de dados (DOSOVITSKIY et al., 2020) (TOUVRON et al., 2021).

Diferentemente das CNNs, que utilizam filtros convolucionais para extrair características locais das imagens, o ViT divide a imagem em *patches* de tamanho fixo e os trata como *tokens* de uma sequência, semelhante ao processamento de palavras em modelos de linguagem natural (DOSOVITSKIY et al., 2020). Cada *patch* é transformado em um vetor de *embeddings* e combinado com uma codificação posicional antes de ser processado por camadas de autoatenção, um exemplo pode ser observado na Figura 2.14. Esse mecanismo permite que o modelo capture relações na imagem desde as primeiras camadas, diferentemente das CNNs, que aprendem relações progressivamente (RAGHU et al., 2021).

Um aspecto importante do ViT é o mecanismo de autoatenção, que calcula a importância de cada *patch* em relação aos outros. Esse processo é realizado por meio das matrizes de atenção que aprendem a destacar regiões importantes da imagem de forma adaptativa (KHAN et al., 2022). Essa abordagem reduz a necessidade de operações locais repetitivas, comuns nas CNNs, e permite que o modelo seja escalável (TOUVRON et al., 2021). Além das camadas de autoatenção, o ViT também inclui mecanismos como camadas de normalização e projeções lineares para estabilizar o treinamento e melhorar a eficiência computacional (XIE et al., 2021). O *token* de classificação é um elemento adicional inserido na sequência de *patches* e representa a saída final do modelo, sendo utilizado para tarefas de classificação de imagens (DOSOVITSKIY et al., 2020).

Com o crescimento do ViT na área de visão computacional, muitas ideias e modelos foram propostos para melhorar sua eficiência e adaptabilidade. Um exemplo é o modelo ViT-Base-Patch16-224-In21k, disponibilizado pelo Google na plataforma *Hugging Face*, que foi treinado em um grande conjunto de dados e apresenta desempenho positivo em tarefas de visão computacional (Hugging Face, 2020). A arquitetura ViT-Base-Patch16-224-In21k mantém os princípios fundamentais do ViT original, utilizando *patches* de 16×16 *pixels* e um *encoder Transformer* para processar a imagem como uma sequência de *tokens*, garantindo flexibilidade e eficiência no aprendizado de representações visuais. Devido a isso, o ViT será utilizado para os testes desse trabalho, onde o conjunto de imagens aplicados passarão anteriormente por um processo de aumento de resolução com técnicas de SR.

Figura 2.14 – Estrutura do ViT, que divide imagens em *patches* e utiliza um codificador baseado em *Transformer* para extrair características globais.



Fonte: (DOSOVITSKIY et al., 2020).

2.2 Trabalhos Relacionados

O primeiro estudo abordado apresenta uma análise para a classificação de malária utilizando TL, a estratégia descrita por Reddy e Juliet (2019) faz uso da rede pré-treinada no banco de dados da ImageNet, a ResNet50. Essa é uma arquitetura residual amplamente reconhecida para tarefas de classificação de imagens. O autor propõe uma extensão da ResNet50, adicionando ao final uma camada densa e totalmente conectada com a função de ativação *sigmoid*. A base de dados utilizada no estudo contém 27.558 imagens de células infectadas e não infectadas por malária (REDDY; JULIET, 2019). Os resultados finais evidenciados pelo modelo são promissores na tarefa de classificação, alcançando uma acurácia de 95,91% no treinamento e 95,4% no teste. O autor conclui que o uso de *transfer learning* para a classificação de imagens de malária trouxe bons resultados, mesmo sem a utilização de hardwares modernos, como GPUs ou TPUs. No entanto, ele sugere que o uso desses hardwares modernos pode aumentar a precisão e reduzir o tempo de execução. Além do modelo ResNet50 utilizado no estudo, o autor recomenda a experimentação com outros modelos, como o Google Inception ou o VGG de Oxford, no conjunto de dados de imagens de células de malária. Ele acredita que esses modelos podem potencialmente proporcionar uma melhor acurácia na classificação das imagens (REDDY; JULIET, 2019).

Já o estudo conduzido por [Minarno et al. \(2023\)](#), embora também empregue a técnica de TL para a classificação de malária, e utilize da mesma base de dados, apresenta suas particularidades. Inicialmente, observa-se a escolha da rede Inception V3. Além disso autor optou por aplicar *data augmentation* no conjunto de dados, técnica essa que consiste em aplicar leves alterações no conjunto de dados já existente, como rotações e mudança de cor, com o intuito de evitar *overfitting* e garantir que o modelo generalize de maneira eficaz ([RAY, 2021](#)). Outro aspecto interessante é a divisão do estudo em três cenários distintos, nos quais diferentes otimizadores foram aplicados para fins comparativos. No primeiro cenário, foi adotado o SGD (*Stochastic Gradient Descent*) com uma taxa de aprendizado de 0.001, enquanto no segundo e terceiro cenários com uma taxa de aprendizado de 0.001 e 0.0001 foram utilizados o Adam e o RMSprop (*Root Mean Square Propagation*), respectivamente. O autor conclui dizendo que com base no cenário de teste que modelou a arquitetura Inception-V3, a aplicação do otimizador RMSprop demonstrou gerar um desempenho superior, com uma acurácia de 97%, quando comparado aos cenários de teste anteriores que utilizavam o otimizador Adam e o otimizador SGD, como visto na [Tabela 2.1](#).

Tabela 2.1 – Resultados obtidos nos três cenários propostos.

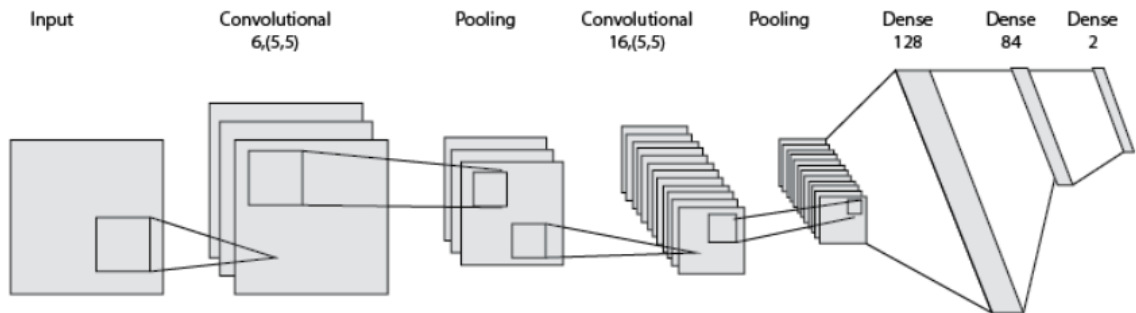
Cenário	Precisão	Recall	F1-Score	Acurácia
Cenário 1	0,95	0,97	0,96	0,96
Cenário 2	0,96	0,98	0,97	0,97
Cenário 3	0,97	0,98	0,97	0,97

Fonte: ([MINARNO et al., 2023](#)).

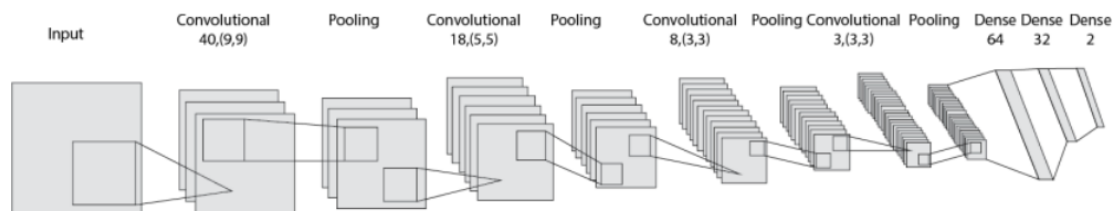
[Harahap et al. \(2021\)](#) investigaram duas arquiteturas de CNN em conjunto com técnicas de *data augmentation* para a tarefa de classificação da malária em células sanguíneas, utilizando a mesma base de dados dos artigos anteriores. A primeira arquitetura utilizada é a LeNet-5, que conta com duas camadas de convolução e duas camadas densas ocultas, como ilustrada pela [Figura 2.15 \(a\)](#). Já a segunda rede escolhida pelo autor é a DRNet, que, diferente da anterior, possui quatro camadas de convolução e duas camadas densas ocultas, conforme a [Figura 2.15 \(b\)](#). Os autores concluem dizendo que o método de CNN com os modelos Lenet-5 e DRnet mostrou-se eficaz na classificação de células sanguíneas, já que avaliando o desempenho obtido no estudo para ambas as redes, é possível destacar a LeNet-5, que obteve uma acurácia de 95,7%, enquanto a rede DRNet obteve 95%.

Outro trabalho que pode ser considerado relevante para o estudo que está sendo proposto foi escrito por [Çinar e Yildirim \(2020\)](#). Ele testa uma base de dados distinta da que será trabalhada nesse estudo, em um total de seis arquiteturas de CNN, algumas já foram até mesmo citadas anteriormente, como a Inception V3 e ResNet50. O ponto destaque desse artigo é o pré-processamento dos dados, o autor opta por testar a base padrão e depois aplicar dois tipos de filtros espaciais. O primeiro é o filtro Gaussiano, que é utilizado no processamento de imagens para suavizar uma imagem, ou seja, reduzir ruído da imagem como um todo ([JESUS; JR, 2015](#)). O segundo filtro é

Figura 2.15 – Arquiteturas avaliadas por Harahap et al. (2021).



(a) Arquitetura LeNet-5.



(b) Arquitetura DRNet.

Fonte: (HARAHAP et al., 2021).

o da mediana, ele também é utilizado com o intuito de reduzir o ruído da imagem, mas também conserva os detalhes, como por exemplo as bordas (BATISTA; NASCIMENTO; FILHO, 1998). O artigo é finalizado com uma análise dos resultados, onde após verificar valores de acurácia combinando diferentes redes com os tipos de dados (Padrão, Filtro Gaussiano, Filtro da Mediana), foi observado que o o dataset com o filtro Gaussiano aplicado desempenhou melhor, em especial com a rede DenseNet201, obtendo uma acurácia de 97,83%.

O estudo de Shekar, Revathy e Goud (2020) aborda o desenvolvimento e avaliação de três modelos de CNN para a classificação de malária: o modelo CNN Básico, o modelo CNN Congelado e o modelo CNN Ajustado. O modelo CNN Básico, construído do zero e treinado com imagens de células sanguíneas infectadas e saudáveis, alcançou uma precisão de 94%. Em seguida, o modelo CNN Congelado, que consiste na repetição dos passos de treinamento e teste após o congelamento de camadas do modelo básico, obteve uma precisão de 92%. Por fim, o modelo CNN Ajustado, que envolve o ajuste fino do modelo congelado, demonstrou a maior precisão, atingindo 96%, com a correta identificação de 4.004 células saudáveis e 3.933 células infectadas. Segundo o autor, os resultados indicam que o modelo CNN Ajustado supera os demais em termos de precisão na predição de células infectadas pela malária, destacando a eficácia dos algoritmos de *deep learning* na classificação da doença. Além disso Shekar, Revathy e Goud (2020), apontam para futuras aplicações dessas técnicas na classificação de outras doenças, como pneumonia, câncer de mama e COVID-19, promovendo avanços na área de diagnósticos médicos e contribuindo para melhores desfechos de saúde.

Os estudos abordados demonstram o sucesso de diferentes arquiteturas de *Convolutional Neural Network* e técnicas de aprendizado de máquina na classificação da malária em imagens de células sanguíneas. O trabalho de Reddy e Juliet (2019) destacou a eficácia da ResNet50 com uma camada adicional e a utilização de *transfer learning*, enquanto Minarno et al. (2023) explorou a Inception V3 com *data augmentation* e diferentes otimizadores, revelando que o RMSprop apresentou o melhor desempenho. Harahap et al. (2021) investigaram a LeNet-5 e a DRNet, com a LeNet-5 obtendo uma ligeira vantagem em termos de acurácia. Além disso, Çinar e Yildirim (2020) enfatizaram a importância do pré-processamento de dados, especialmente com o uso de filtros espaciais, onde a DenseNet201 se destacou após a aplicação do filtro Gaussiano. Por fim, Shekar, Revathy e Goud (2020) comparou três modelos de CNN, concluindo que o modelo CNN ajustado foi o mais preciso.

O estudo conduzido por Sabir et al. (2023) apresenta uma abordagem para a classificação de fibrose pulmonar a partir de imagens de tomografia computadorizada do tórax, utilizando um *framework* baseado em ViT denominado FibroVit. A arquitetura FibroVit foi inicializada utilizando especificamente o modelo ViT-base-patch16-224 (Hugging Face, 2020), que passou por um pré-treinamento no conjunto de dados ImageNet-21k (SABIR et al., 2023). Os autores utilizaram um *dataset* de 13.486 amostras selecionadas de um conjunto maior disponível publicamente no Kaggle. Para preparar os dados, as imagens em escala de cinza foram convertidas para RGB, e aplicou-se uma estratégia de *data augmentation* que incluiu rotações, *flips*, *cropping*, redimensionamento e ajustes randômicos de brilho, contraste, saturação e tonalidade, entre outras transformações (SABIR et al., 2023). Os modelos foram treinados e validados com uma divisão de dados de 80% para treino, 10% para validação e 10% para teste, e otimizados utilizando o otimizador AdamW com uma taxa de aprendizado fixa de 0.0001 durante uma única época de treinamento. Os resultados indicaram que o modelo obteve um desempenho positivo, alcançando uma acurácia de teste de 100% e uma acurácia de validação de 99.85%, além de apresentar as menores perdas de treinamento e validação (SABIR et al., 2023). Os autores concluem que o modelo ViT otimizado funciona como uma ferramenta de diagnóstico confiável para a categorização automatizada de indivíduos com fibrose pulmonar usando exames de tomografia do tórax, destacando seu potencial para melhorar a precisão diagnóstica e otimizar os recursos de saúde (SABIR et al., 2023).

A maioria dos estudos mencionados, incluindo Reddy e Juliet (2019), Minarno et al. (2023), Harahap et al. (2021), e Shekar, Revathy e Goud (2020), utilizam o NIH Malaria *Dataset*, que também é a base de dados usada no presente estudo. Este conjunto de dados é composto por 27.558 imagens de células sanguíneas, categorizadas entre infectadas e não infectadas por parasitas da malária. Em consonância com esses trabalhos, o presente estudo explora a arquitetura EfficientNet-B0 para a classificação de malária, visando sua eficiência e precisão já demonstradas em tarefas de classificação de imagens (TAN; LE, 2019). Além disso, motivado pelos avanços recentes em ViTs, também se investiga o ViT-base-patch16-224-in21k (Hugging Face, 2020), modelo que se destacou na detecção de fibrose pulmonar (SABIR et al., 2023).

A adoção dessa abordagem visa avaliar o potencial dos *Transformers* em relação os métodos convencionais baseados em CNNs, ampliando o escopo da pesquisa e assim possibilitando uma análise comparativa entre ambas as arquiteturas para a tarefa de classificação da malária.

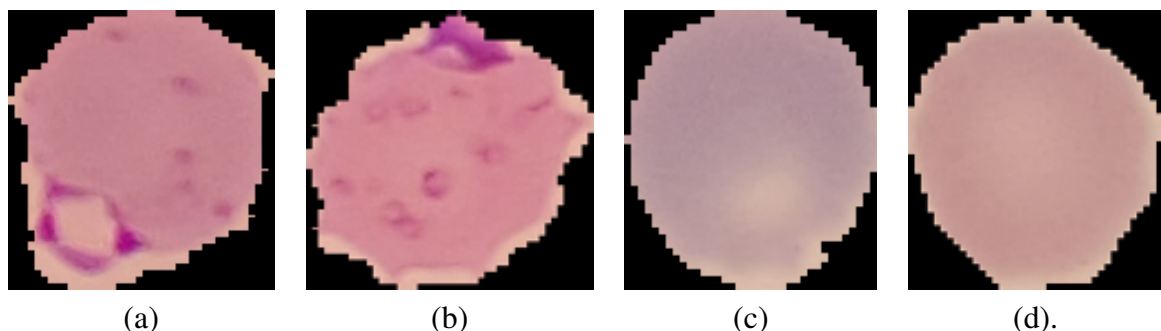
3 Base de dados e Metodologia Proposta

Nos capítulos anteriores, algumas abordagens para a classificação da malária em células sanguíneas utilizando *deep learning* foram exploradas. Neste capítulo, o objetivo é alcançar resultados comparáveis a esses trabalhos, utilizando a arquitetura de CNN EfficientNet-B0, conhecida por sua eficiência em termos de custo computacional e desempenho em classificação de imagens (TAN; LE, 2019), a fim de verificar o desempenho do modelo e viabilizar novas opções. Será também apresentada a base de dados utilizada neste estudo, suas particularidades e como foi processada para ser aplicada ao objetivo almejado (Seção 3.1). Além disso, as configurações do modelo proposto serão apresentadas, bem como as métricas utilizadas para avaliar o desempenho do mesmo (Seção 3.2).

3.1 Base de dados

O presente estudo faz uso de um conjunto de dados composto por 27.558 imagens de RBC, divididas em duas classes: 13.775 imagens de células parasitadas Figura 3.1(a) e 13.783 de células não infectadas Figura 3.1(b). Este conjunto de dados foi originalmente disponibilizado por Rajaraman et al. (2018) e está acessível no site oficial da *National Library of Medicine*. As imagens foram capturadas a partir de esfregaços sanguíneos, utilizando microscópios acoplados a *smartphones* Android. As imagens capturadas foram segmentadas ao ponto de que a base é constituída apenas pelas RBCs.

Figura 3.1 – Exemplos de imagens contidas na base de dados. As imagens (a) e (b) representam células infectadas e (c) e (d), células não infectadas.



Fonte: (RAJARAMAN et al., 2018).

Para segmentar as RBCs, Rajaraman et al. (2018) optou um algoritmo baseado no método *level-set*, que é uma técnica popular em segmentação de imagens. O processo começou com a aplicação de um filtro conhecido como *Laplacian of Gaussian* (LoG). Este filtro combina duas etapas: primeiro, ele suaviza a imagem, diminuindo o ruído e tornando os detalhes mais claros; em seguida, ele destaca as bordas das células, que são as regiões onde a cor ou a intensidade da

imagem muda de maneira mais brusca. Dessa forma, o filtro LoG ajuda a identificar o centro das células, fornecendo pontos iniciais para a segmentação. O filtro LoG é obtido através da combinação do operador Laplaciano com a função Gaussiana. A função Gaussiana em duas dimensões é dada por:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad (3.1)$$

onde x e y são as coordenadas da imagem, e σ é o desvio padrão da função Gaussiana que controla o grau de suavização. Após a aplicação da função Gaussiana para suavizar a imagem (LINI, 2023), aplica-se o operador Laplaciano para detectar as bordas, definido como:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}, \quad (3.2)$$

onde I é a imagem suavizada. A combinação do operador Laplaciano com a suavização Gaussiana resulta no filtro Laplaciano do Gaussiano (LoG) (LINI, 2023), que pode ser expresso como:

$$LoG(x, y) = -\frac{1}{\pi\sigma^4} \left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}. \quad (3.3)$$

Este filtro suaviza a imagem e detecta as bordas das RBCs, fornecendo uma base para a segmentação inicial (RAJARAMAN et al., 2018).

Após a identificação dos centroides das células, o método *level-set* foi empregado para delinear o contorno exato de cada célula. O *level-set* é uma técnica que começa com uma forma inicial simples, como um círculo, e vai moldando essa forma até que ela coincida com as bordas reais das células. Isso é feito de maneira iterativa, como se a forma inicial fosse “empurrada” e “puxada” pelas forças ao longo das bordas das células, até que se ajuste perfeitamente a elas (RAJARAMAN et al., 2018).

Matematicamente, o método *level-set* pode ser descrito pela equação de evolução de curvas:

$$\frac{\partial \phi}{\partial t} = F|\nabla \phi|, \quad (3.4)$$

onde ϕ é a função de nível (*level-set function*), F é a força de movimentação da curva e $\nabla \phi$ é o gradiente da função de nível (DIAZ, 2015). Essa equação descreve como a curva se move ao longo do tempo, ajustando-se às bordas das células (DIAZ, 2015). O valor de F é determinado pelas características da imagem, como a intensidade e as bordas detectadas pelo filtro LoG, fazendo com que a curva seja atraída para as bordas das RBCs.

Após a segmentação inicial, técnicas de pós-processamento, como a abertura morfológica, foram aplicadas para remover pequenos fragmentos indesejados e falsos positivos, garantindo que apenas as células reais fossem mantidas (RAJARAMAN et al., 2018). Este mesmo conjunto de dados está disponível no Kaggle¹ e no TensorFlow², oferecendo acesso aberto e incentivando sua utilização em pesquisas adicionais.

¹ Disponível em: <<https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria>>. Acesso em setembro de 2024.

² Disponível em: <<https://www.tensorflow.org/datasets/catalog/malaria?hl=pt-br>> Acesso em setembro de 2024.

3.1.1 Pré-processamento da base

O objetivo principal dessa etapa é garantir que as imagens estejam em um formato adequado para o processamento pela arquitetura proposta (MINARNO et al., 2023). O conjunto de dados utilizado foi carregado a partir do *TensorFlow Datasets*, especificamente o *dataset* de malária, que contém imagens de células sanguíneas infectadas e não infectadas (RAJARAMAN et al., 2018). Cada imagem foi redimensionada para 32×32 pixels, que é o tamanho de imagem esperado pela camada de entrada que será utilizada.

Além do redimensionamento, as imagens foram normalizadas dividindo-se por 255, o que converte os valores de inteiros (0-255) para uma escala de ponto flutuante entre 0 e 1 (REIS-SILVA, 2022). Esta normalização é importante para garantir que os dados de entrada estejam na mesma escala, o que acelera o treinamento do modelo e tende a melhorar a estabilidade numérica, evitando problemas com gradientes muito pequenos ou muito grandes (REIS-SILVA, 2022).

Os rótulos das imagens, que inicialmente estavam em formato categórico (0 - infectado, 1 - não infectado), foram mantidos no formato binário. Esse formato permite que o modelo classifique corretamente as amostras durante o treinamento, sem a necessidade de transformações adicionais. Ao utilizar rótulos binários, cada classe é representada por um único valor escalar, o que simplifica o processo de treinamento e evita a necessidade de remover qualquer noção de ordem ou hierarquia entre as classes, já que elas são mutuamente exclusivas e não ordenadas.

3.1.2 Divisão dos dados

Nos experimentos conduzidos neste trabalho, a divisão dos dados do *dataset* foi realizada utilizando duas abordagens distintas, de forma a permitir uma análise mais comparativa e abrangente. A primeira abordagem segue a metodologia adotada por artigos de estado da arte no tema, por exemplo (MINARNO et al., 2023) no qual o *dataset* é dividido em 70% para treino, 15% para validação e 15% para teste. Essa configuração foi escolhida para manter a consistência com a literatura existente, viabilizando a comparação direta dos resultados obtidos com outros estudos.

No contexto de *deep learning*, o conjunto de dados de treinamento é utilizado para que a CNN aprenda, ajustando seus pesos e parâmetros com base nas imagens e classes fornecidas. Como essa etapa é de extrema importância para o aprendizado do modelo, a maior parte dos dados é destinada a essa fase. Durante o treinamento, o conjunto de validação é empregado para monitorar o desempenho do modelo e realizar ajustes nos hiper-parâmetros quando necessário, garantindo que a rede seja capaz de generalizar para dados não vistos. Esse conjunto avalia a precisão e a performance do modelo em exemplos que não participaram do treinamento, prevenindo o *overfitting*. Finalmente, o conjunto de teste, composto por dados diferentes dos utilizados durante o treino e validação, é usado para avaliar a performance final da CNN, oferecendo uma estimativa realista da sua capacidade de generalização em novos exemplos.

A segunda abordagem envolve o uso de validação cruzada estratificada, que, neste caso, utiliza cinco *folds* ($k = 5$) para dividir o *dataset*. Essa técnica é utilizada para garantir que o modelo seja testado em diferentes subconjuntos de dados, promovendo uma avaliação mais ampla de seu desempenho. A aplicação do *stratified k-fold* também permite explorar a variabilidade dos resultados em diferentes partições dos dados (PRUSTY; PATNAIK; DASH, 2022), minimizando o risco de *overfitting* e proporcionando um resultado mais preciso e abrangente da capacidade do modelo em generalizar.

As abordagens oferecem vantagens distintas: a primeira, por permitir comparações diretas com estudos anteriores, e a segunda, por fornecer uma avaliação mais abrangente em diferentes cenários, viabilizando assim diferentes cenários de comparação, para de fato avaliar o desempenho do modelo proposto.

3.2 Modelos Propostos

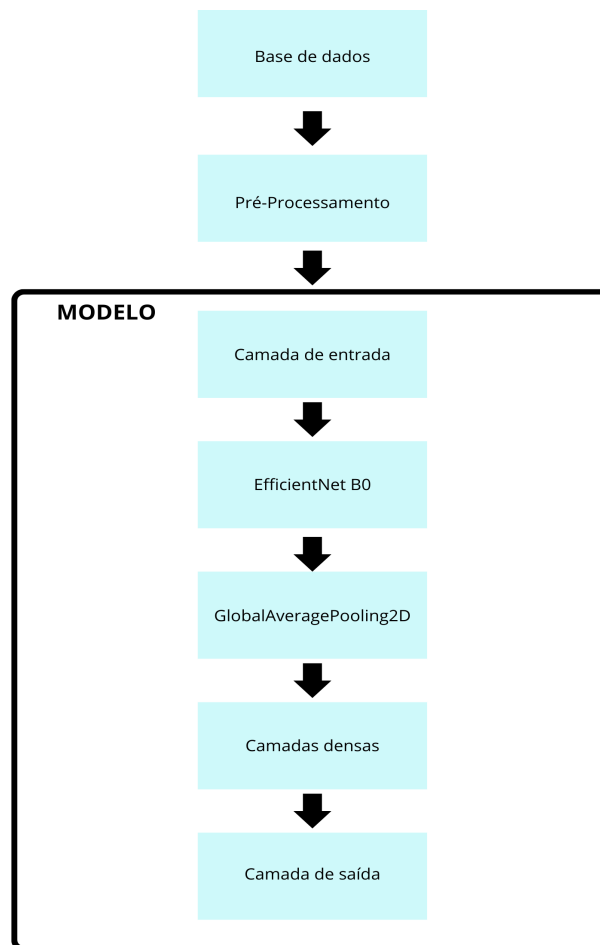
Esta seção é dedicada a explicar os modelos propostos neste trabalho. Foram escolhidas a EfficientNet-B0 e o ViT como bases das arquiteturas, ambas redes pré-treinadas na base *ImageNet*. A EfficientNet-B0 foi selecionada devido ao seu equilíbrio entre precisão e eficiência computacional, alcançando um desempenho competitivo com um número relativamente menor de parâmetros em comparação a outras arquiteturas (TAN; LE, 2019). Já a ViT-base-patch16-224-in21k (Hugging Face, 2020), um modelo de ViT, foi incluída para avaliar a eficiência dos *transformers* em tarefas de visão computacional, uma vez que este modelo tem demonstrado resultados promissores ao capturar relações de longo alcance em imagens de forma mais eficaz do que as CNN tradicionais (DOSOVITSKIY et al., 2020). Além disso, também serão abordadas as estratégias utilizadas para o treinamento dos modelos em cada arquitetura, como *Data Augmentation* (aumento de dados) e SR.

3.2.1 EfficientNet-B0

A primeira camada do modelo é a de entrada, onde se espera imagens redimensionadas pelo processamento para um tamanho de 32×32 *pixels*, com três canais de cor. Essa escolha foi feita para reduzir a complexidade computacional, uma vez que imagens menores demandam menos recursos para serem processadas, mantendo, contudo, informações suficientes para a tarefa de classificação. O fluxo do modelo pode ser visto na Figura 3.2

A base do modelo é composta pela EfficientNet-B0, que foi configurada para não incluir a camada final de classificação original, já que o objetivo é realizar um ajuste fino para uma tarefa específica de classificação binária da malária. A saída da EfficientNet-B0 é conectada a uma camada de *GlobalAveragePooling2D*, que é utilizada em vez de *Flatten* para reduzir o número de parâmetros e ajudar na generalização do modelo. O *Global Average Pooling* resume as informações espaciais, produzindo uma única saída por canal de convolução, o que diminui a

Figura 3.2 – Figura que ilustra o fluxo do modelo.



Fonte: próprio autor.

dimensionalidade do problema e reduz a tendência do modelo ao *overfitting* (NGUYEN *et al.*, 2019).

Após o *GlobalAveragePooling2D*, foram adicionadas três camadas densas de 64, 32 e 32 neurônios, respectivamente, todas utilizando a função de ativação ReLU (*Rectified Linear Unit*). Essas camadas foram incluídas para refinar as características extraídas pela EfficientNet-B0 e aumentar a capacidade do modelo de capturar padrões não lineares complexos. Cada uma dessas camadas densas é seguida por uma camada de *BatchNormalization*, que normaliza as ativações da camada anterior, acelerando o treinamento e promovendo maior estabilidade ao modelo.

Para mitigar o risco de *overfitting*, foram incorporadas duas técnicas de regularização nas camadas densas citadas anteriormente: *Dropout* e regularização L2. O *Dropout*, aplicado com uma taxa de 30%, desativa aleatoriamente uma fração dos neurônios durante o treinamento, forçando a rede a aprender características mais generalistas e a depender menos de neurônios específicos, o que melhora o desempenho em dados não vistos (BUDHIRAJA, 2016). Já a regularização L2 adiciona uma penalidade ao custo total do modelo, proporcional ao quadrado dos pesos das conexões (CORTES; MOHRI; ROSTAMIZADEH, 2012). Essa penalização incentiva a rede a manter os pesos menores, o que ajuda a controlar a complexidade do modelo e evita que

ele se ajuste excessivamente aos dados de treinamento (CORTES; MOHRI; ROSTAMIZADEH, 2012). Ao penalizar pesos grandes, a regularização L2 torna o modelo menos propenso a memorizar os dados de treinamento, favorecendo uma melhor generalização (CORTES; MOHRI; ROSTAMIZADEH, 2012).

Por fim, a camada de saída do modelo implementado utiliza a função de ativação *sigmoid* (Figura 3.3), uma vez que o objetivo deste trabalho é distinguir entre duas classes: células infectadas e não infectadas. A função *sigmoid* é definida pela Equação 3.5 :

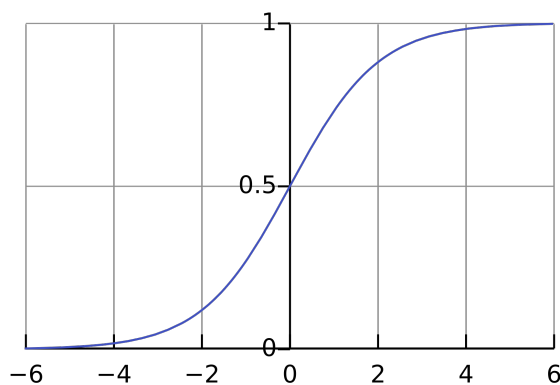
$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (3.5)$$

onde:

- $\sigma(x)$ é a saída da função sigmoide,
- x é a entrada da função,
- e é a constante de Euler.

A função *sigmoid* (Figura 3.3) transforma a saída da camada densa final em uma probabilidade entre 0 e 1, onde valores próximos a 0 indicam células infectadas e valores próximos a 1 indicam células não infectadas (HAN; MORAGA, 1995). Esse comportamento é ideal para a classificação binária, pois garante que a saída do modelo seja uma probabilidade, permitindo que cada entrada seja mapeada para uma das duas classes de forma clara e eficiente (ZAIDI, 2022). Além disso, a função *sigmoid* tem a propriedade de suavizar a transição entre as classes, o que pode auxiliar na estabilidade do treinamento do modelo e na convergência durante o processo de otimização (HAN; MORAGA, 1995).

Figura 3.3 – Função Sigmoid.



Fonte: (PyTorch, 2024).

A otimização do modelo é realizada com o algoritmo Adam, que é uma extensão do método de descida do gradiente estocástico. O Adam é bastante utilizado em problemas de *deep*

learning, especialmente em grandes conjuntos de dados e modelos complexos (MINARNO et al., 2023).

Uma característica importante da configuração do otimizador é o uso de uma programação de decaimento da taxa de aprendizado baseada em cosseno, implementada com o *Cosine Decay*. A função de decaimento cosseno modula a taxa de aprendizado ao longo do tempo, começando de uma taxa inicial relativamente alta e reduzindo gradualmente para uma taxa final mais baixa (CORREA, 2019). Esse mecanismo foi aplicado para evitar o *overfitting*, que a rede estava apresentando durante o treinamento. Ao utilizar uma taxa de aprendizado mais alta no início, o modelo explora melhor o espaço de parâmetros, e o decaimento progressivo ajuda a refinar os ajustes na fase final, minimizando o erro de generalização (CORREA, 2019).

3.2.2 ViT

Para a implementação deste modelo, foi utilizada a versão pré-treinada e seguindo a configuração *default* do *google/vit-base-patch16-224-in21k*. Esta versão do ViT opera com imagens de entrada no tamanho de 224×224 pixels, divididas em *patches* de 16×16 pixels, os quais são linearmente projetados em *embeddings* e processados por camadas sequenciais de atenção (Hugging Face, 2020).

O pipeline de treinamento seguiu uma estrutura semelhante à utilizada na EfficientNet-B0. Inicialmente, as imagens foram carregadas e redimensionadas para 224×224 pixels. A extração de características foi realizada com o *ViTFeatureExtractor*, responsável por adequar as imagens ao formato esperado pelo modelo e um conjunto personalizado com os dados foi construído.

A divisão do conjunto de dados seguiu a mesma proporção aplicada na etapa anterior, sendo 70% para treino, 15% para validação e 15% para teste. Além disso, todos os treinos do modelo foram realizados utilizando as imagens com o SR aplicado tanto sobre as imagens do conjunto de treinamento quanto sobre o conjunto de teste. Após o treinamento, o modelo foi avaliado quantitativamente por meio das métricas de acurácia, precisão, *recall* e *F1-score*.

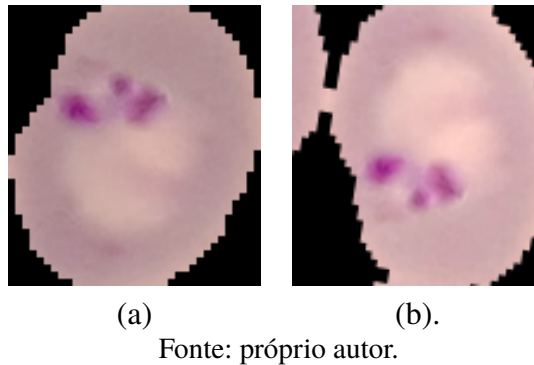
3.2.3 Estratégia com *Data Augmentation*

Nesta estratégia, será aplicada a técnica de *Data Augmentation* para verificar seu impacto na performance do modelo EfficientNet-B0 e do ViT. Embora o conjunto de dados possua uma quantidade relativamente alta de imagens, a aplicação de técnicas de aumento de dados pode contribuir para melhorar a generalização do modelo ao adicionar imagens com ruídos diferentes e reduzir o *overfitting* (MINARNO et al., 2023).

A abordagem segue a metodologia proposta por Minarno et al. (2023), que demonstrou resultados promissores em tarefas similares. As transformações implementadas incluem rotações, espelhamentos horizontais e verticais, deslocamentos e ajustes de escala, criando variações artificiais dos dados originais sem alterar a informação semântica das imagens, como pode ser

observado na Figura 3.4. Vale ressaltar que essa técnica somente foi aplicada aos dados de treinamento e que a divisão proposta por Minarno et al. (2023) foi mantida (70% para treino, 15% para validação e 15% para teste).

Figura 3.4 – Exemplo de imagem original (a) e sua versão aumentada por data augmentation (b).



O objetivo principal do uso desta estratégia é avaliar se a diversificação artificial do conjunto de treinamento através do *Data Augmentation* proporciona uma melhoria significativa na capacidade de generalização do modelo EfficientNet-B0 e do modelo ViT quando comparado aos resultados obtidos sem esta técnica. Os parâmetros específicos das transformações aplicadas serão detalhados na seção de experimentos.

3.2.4 Estratégia com *Super Resolution*

Nesta estratégia, será aplicada a técnica de *Super Resolution* com o objetivo de avaliar seu impacto na capacidade de classificação do modelo. Como o conjunto de dados contém imagens de RBC, que apresentam estruturas celulares pequenas e detalhes sutis, a hipótese é que a aplicação de SR possa compensar possíveis limitações na resolução original, tornando mais evidentes as diferenças entre células saudáveis e infectadas.

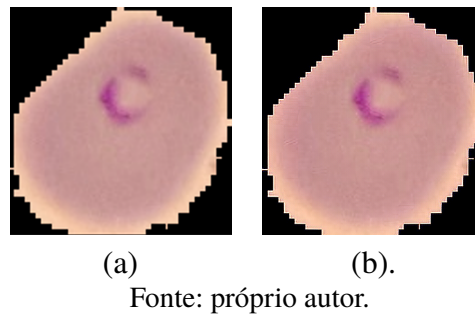
Para implementação desta técnica, será utilizado o modelo pré-treinado ESRGAN. O ESRGAN é conhecido por sua capacidade de recuperar detalhes sutis em imagens de baixa resolução (WANG et al., 2018), o que potencialmente poderia melhorar a capacidade de generalização do classificador ao evidenciar características discriminativas importantes nas células.

É possível observar na Figura 3.5 que o processo de SR conseguiu destacar alguns detalhes bem finos na imagem processada. Como, por exemplo, bordas da imagem e também em partes onde se encontram o protozoário da malária.

As imagens processadas pelo ESRGAN serão redimensionadas antes de serem utilizadas para treinar tanto o modelo EfficientNet-B0 quanto o ViT. Para uma avaliação mais abrangente, no modelo EfficientNet-B0 serão testados três otimizadores distintos: Adam, RMSprop e SGD, buscando identificar possíveis variações no desempenho do modelo com a aplicação do SR.

O principal objetivo é determinar se a aplicação de (KUNDU, 2022) resulta em uma

Figura 3.5 – Exemplo de imagem original (a) e sua versão após o processo de SR (b).



Fonte: próprio autor.

melhoria significativa no desempenho dos modelos, especialmente na distinção de características sutis que podem ser críticas para a classificação correta das células (WANG et al., 2018).

3.2.5 Métricas

No contexto deste trabalho, é essencial avaliar o desempenho do modelo utilizando uma combinação de métricas. Embora a acurácia seja uma métrica comum, ela pode não ser suficiente por se tratar de um conjunto de dados médicos (MINARNO et al., 2023). Por essa razão, além da acurácia, serão utilizadas as métricas de precisão, *recall* e *F1-score*. Todas elas podem ser definidas em termos de:

- **True Positives (TP):** São os verdadeiros positivos, ou seja, as instâncias que foram corretamente classificadas como positivas.
- **True Negatives (TN):** São os verdadeiros negativos, ou seja, as instâncias que foram corretamente classificadas como negativas.
- **False Positives (FP):** São os falsos positivos, as instâncias que foram incorretamente classificadas como positivas.
- **False Negatives (FN):** São os falsos negativos, as instâncias que foram incorretamente classificadas como negativas.

Elas são definidas da seguinte forma:

A acurácia é uma das métricas mais simples para avaliar o desempenho de um modelo de classificação. Ela mede a proporção de previsões corretas sobre o total de previsões feitas pelo modelo (YACOUBY; AXMAN, 2020). Formalmente, a acurácia é dada pela fórmula:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN}. \quad (3.6)$$

Os valores da acurácia variam de 0 até 1, sendo que um valor de 1 indica que o modelo fez todas as previsões corretamente, enquanto um valor de 0 significa que todas as previsões

foram incorretas. Valores intermediários indicam o percentual de acertos do modelo em relação ao total de previsões feitas. A acurácia é considerada uma boa métrica quando as classes estão balanceadas, mas pode não ser uma boa escolha em casos onde há desbalanceamento da base de dados (YACOUBY; AXMAN, 2020).

A precisão mede a proporção de instâncias corretamente classificadas como positivas em relação ao total de instâncias que o modelo previu como positivas. Ela é útil em situações onde o custo de um falso positivo é alto, ou seja, onde é importante minimizar as previsões incorretas de positivo (YACOUBY; AXMAN, 2020). A fórmula da precisão é:

$$\text{Precisão} = \frac{TP}{TP + FP}. \quad (3.7)$$

Os valores da precisão variam de 0 até 1. Um valor de 1 indica que todas as instâncias classificadas como positivas pelo modelo são, de fato, positivas, ou seja, não houve falsos positivos. Já um valor de 0 significa que todas as previsões positivas foram incorretas, representando um modelo que classifica de forma errada todas as instâncias previstas como positivas. Valores intermediários indicam a proporção de previsões positivas corretas em relação ao total de previsões positivas feitas. Uma alta precisão significa que, entre as instâncias classificadas como positivas, a maioria realmente pertence à classe positiva (YACOUBY; AXMAN, 2020).

Considerando o contexto do trabalho atual, a precisão é uma métrica essencial para avaliar a qualidade das predições positivas feitas pelo modelo. No escopo do diagnóstico de malária, uma alta precisão indica que a maioria das amostras classificadas como positivas realmente corresponde a casos verdadeiros de malária. Isso é importante, pois reduz a ocorrência de falsos positivos, evitando que pacientes saudáveis sejam submetidos a tratamentos desnecessários.

O *recall* mede a capacidade do modelo de identificar corretamente todas as instâncias positivas. Ou seja, ele calcula a proporção de verdadeiros positivos em relação ao total de instâncias que são de fato positivas (YACOUBY; AXMAN, 2020). A fórmula do *recall* é:

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (3.8)$$

Os valores do *recall* também variam de 0 a 1. Um valor de 1 indica que o modelo conseguiu identificar todas as instâncias positivas, ou seja, não houve falsos negativos. Por outro lado, um valor de 0 significa que o modelo falhou em identificar qualquer instância positiva. Valores intermediários indicam o percentual de instâncias positivas corretamente identificadas pelo modelo. O *recall* é importante em situações onde o custo de falsos negativos é alto (YACOUBY; AXMAN, 2020), como por exemplo em diagnósticos médicos, onde é fundamental identificar todas as instâncias de uma determinada doença. Portanto, o *recall* pode ser considerado uma das mais importantes métricas para garantir que o modelo seja eficaz na classificação dos casos de malária.

O *F1-score* é a métrica que combina a precisão e o *recall* em uma única métrica de desempenho, sendo a média harmônica dessas duas. Ele é particularmente útil quando há um desbalanceamento entre as classes, já que leva em conta tanto os falsos positivos quanto os falsos negativos (YACOUBY; AXMAN, 2020). A fórmula do *F1-score* é:

$$F1 = 2 \times \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}} \quad (3.9)$$

O F1-score atinge seu valor máximo (1) quando tanto a precisão quanto o *recall* são perfeitos, e é 0 quando o modelo falha completamente em uma das métricas (YACOUBY; AXMAN, 2020).

4 Experimentos e Resultados

Neste capítulo, serão apresentados os experimentos realizados e seus respectivos resultados, com o objetivo de alcançar resultados comparáveis aos melhores encontrados na literatura, em especial ao artigo de [Minarno et al. \(2023\)](#), permitindo uma análise comparativa direta.

4.1 Setup dos experimentos

A implementação do classificador proposto tem como base a arquitetura EfficientNet-B0, como visto anteriormente. Os experimentos foram conduzidos em dois ambientes distintos. No primeiro, o *Google Colab*, foram avaliadas técnicas como a validação cruzada estratificada e a comparação entre os resultados obtidos com e sem a utilização de pesos pré-treinados da *ImageNet*. O segundo ambiente de testes foi uma máquina local equipada com um processador *AMD Ryzen Threadripper 3960X*, contendo 24 cores físicos (48 *threads*) de 3.70GHz e 128GB de RAM DDR4, além de uma GPU RTX 3090, com 24GB de RAM GDDR6X e mais de 10 mil CUDA cores. Nesse ambiente, além do treino do modelo utilizando ViT, foram exploradas técnicas como *data augmentation* e SR, sendo esta última testada com diferentes otimizadores. O modelo foi programado em código *Python* utilizando as bibliotecas *TensorFlow* e *Keras*.

Após a apresentação dos ambientes de experimentos, detalha-se a configuração do otimizador e as estratégias aplicadas no treinamento do modelo EfficientNet-B0. Entre essas estratégias, destaca-se o uso do *cosine decay* para atualização dinâmica da taxa de aprendizado, implementado com a API do *Keras*, com o objetivo de evitar mínimos locais, diminuindo o *overfitting*.

Para esta configuração, a taxa de aprendizado inicial foi definida como 0,01, permitindo que o modelo faça grandes ajustes nos pesos durante as primeiras iterações. À medida que o treinamento avança, a taxa de aprendizado decai suavemente até 0,0001, de forma a reduzir o impacto das atualizações dos pesos conforme o modelo converge.

A razão entre a taxa final e a inicial (α) foi calculada pela [Equação 4.1](#):

$$\alpha = \frac{\text{final_learning_rate}}{\text{initial_learning_rate}} = \frac{0.0001}{0.01}. \quad (4.1)$$

O número de passos de decaimento foi determinado com base no número de épocas e no tamanho do lote (*batch size*), conforme a [Equação 4.2](#):

$$\text{decay_steps} = \text{epochs} \times \left(\frac{\text{len(features)}}{\text{batch_size}} \right). \quad (4.2)$$

Essa estratégia foi implementada utilizando a seguinte função da biblioteca *Keras*: `tf.keras.optimizers.schedules.CosineDecay`. O pseudo-código mostrando a configuração do algoritmo pode ser visto no [algoritmo 2.1](#)

Essa configuração foi feita para permitir que o modelo explore amplamente o espaço de soluções no início do treinamento e, posteriormente, ajuste os parâmetros de maneira mais suave nas etapas finais, promovendo uma convergência mais estável e reduzindo o risco de *overfitting*.

Além disso, a função de perda escolhida é a *Binary Crossentropy*. Ela mede a diferença entre as distribuições de probabilidade previstas pelo modelo e os rótulos verdadeiros binários (SAXENA, 2024). Essa função é adequada para tarefas de classificação binária, como a deste trabalho, onde há apenas duas classes (células infectadas e não infectadas). A *Binary Crossentropy* busca minimizar a diferença entre as probabilidades previstas para cada classe e os rótulos corretos, garantindo que o modelo aprenda a distinguir entre as duas categorias (SAXENA, 2024).

Matematicamente, a *Binary Crossentropy* é dada pela Equação 4.3:

$$L = -\frac{1}{N} \sum_{i=1}^N [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)], \quad (4.3)$$

onde N é o número de amostras, y_i é o rótulo verdadeiro da i -ésima amostra, e p_i é a probabilidade prevista pelo modelo para a classe 1.

Agora sobre o *setup* do ViT, foi utilizada a versão pré-treinada e seguindo a configuração default do google/vit-base-patch16-224-in21k, disponível na biblioteca Transformers da Hugging Face (2020). O modelo foi configurado com dois neurônios na camada de saída, representando as classes “infectado” e “não infectado”. O treinamento foi realizado utilizando o *Trainer* da Hugging Face, com os seguintes hiperparâmetros: taxa de aprendizado de 0.0001 e 0.001, *batch size* de 32, e 20 épocas de treinamento. A estratégia de avaliação foi definida para ocorrer ao final de cada época, com armazenamento do melhor modelo ao término do processo. Além disso, foi utilizada a aceleração via FP16 (*mixed precision*) para otimizar o desempenho computacional (Hugging Face, 2020).

4.2 Avaliação com validação cruzada estratificada

Nesta seção os experimentos serão realizados utilizando o modelo descrito anteriormente, tanto com a técnica de TL quanto sem ela, aplicando a validação cruzada estratificada (*Stratified K-Fold*) com $k = 5$. Essa abordagem permite uma avaliação mais completa do desempenho do modelo, garantindo que todas as classes estejam representadas de maneira proporcional em cada subdivisão. A utilização do *Stratified K-Fold* busca reduzir variações de proporção das classes entre diferentes conjuntos de treino e teste, proporcionando uma avaliação mais confiável da capacidade do modelo em generalizar. Os resultados obtidos serão analisados e comparados com os experimentos anteriores, a fim de verificar o quão impactante é essa técnica na performance do modelo, principalmente em relação às métricas como acurácia, precisão e *recall*, e assim avaliar se o uso do TL mantém seu benefício em um cenário de validação mais rigoroso.

Ademais, serão realizados experimentos adicionais para analisar o desempenho do modelo com o uso de TL e sem o uso do *cosine decay*, comparando esses resultados com os experimentos que utilizaram essa técnica de ajuste da taxa de aprendizado. A remoção do *cosine decay* será analisada para avaliar o impacto dessa escolha sobre a estabilidade do treinamento e o comportamento de *overfitting*. Espera-se que, sem o uso dessa técnica, o modelo apresente maior inconstância e possíveis sinais de *overfitting*, uma vez que o decaimento da taxa de aprendizado ao longo das épocas tem o papel de suavizar os ajustes no final do treinamento, evitando que o modelo se ajuste demais aos dados de treino. Esses novos resultados serão contrastados com os anteriores, a fim de identificar se o TL, combinado com o *cosine decay*, continua a fornecer uma vantagem significativa no treinamento do modelo e em seu desempenho geral.

Para evidenciar a utilidade do *cosine decay*, nessa primeira parte dos experimentos, a validação cruzada será aplicada com TL em dois cenários: o primeiro com um *learning rate* fixo de 0,01, e o segundo com o *learning rate* dinâmico, utilizando a configuração do *cosine decay*, os resultados obtidos podem ser vistos na Tabela 4.1, e os valores de acurácia para cada *fold* foram de 88%, 83%, 91%, 89% e 72% respectivamente. A média geral da acurácia foi de 84%, isso evidencia que o modelo não conseguiu alcançar resultados próximos aos já reportados na literatura em quase nenhum *fold*.

Tabela 4.1 – Resultados obtidos com *Transfer Learning* (TL) para 5 *Folds* para os dois cenários de configuração de *learning rate*.

Fold	Classe	<i>Learning rate fixo</i>			<i>Cosine decay</i>		
		<i>Recall</i>	<i>Precisão</i>	<i>F1-score</i>	<i>Recall</i>	<i>Precisão</i>	<i>F1-score</i>
Fold 1	Infetado	0,89	0,88	0,88	0,96	0,96	0,96
	Não infectado	0,88	0,89	0,88	0,96	0,96	0,96
	Média	0,88	0,88	0,88	0,96	0,96	0,96
Fold 2	Infetado	0,66	1,00	0,79	0,94	0,97	0,96
	Não infectado	1,00	0,74	0,85	0,97	0,94	0,96
	Média	0,83	0,87	0,82	0,96	0,96	0,96
Fold 3	Infetado	0,92	0,90	0,91	0,98	0,86	0,92
	Não infectado	0,90	0,91	0,91	0,85	0,98	0,91
	Média	0,91	0,91	0,91	0,91	0,92	0,91
Fold 4	Infetado	0,78	0,99	0,88	0,92	0,99	0,95
	Não infectado	0,99	0,82	0,90	0,99	0,93	0,96
	Média	0,89	0,91	0,89	0,96	0,96	0,96
Fold 5	Infetado	1,00	0,65	0,78	0,91	0,97	0,94
	Não infectado	0,45	0,99	0,62	0,97	0,92	0,94
	Média	0,72	0,82	0,70	0,94	0,94	0,94
Média geral	Infetado	0,85	0,88	0,84	0,94	0,94	0,94
	Não infectado	0,84	0,87	0,83	0,94	0,94	0,94
	Média	0,84	0,87	0,84	0,94	0,94	0,94

Fonte: próprio autor.

Já os resultados obtidos com a aplicação do *cosine decay* e TL foram promissores. A

acurácia obtida em cada *fold* foi de 96%, 95%, 91%, 96%, 94% respectivamente, e a média de todos os *folds* foi de 94%. Os resultados das outras métricas podem ser observados na Tabela 4.1. O *recall* para a classe infectada variou entre 0.91 no *Fold 5* e 0.98 no *Fold 3*, enquanto para a classe não infectada, os valores variaram um pouco mais, atingindo 0.85 no *Fold 3* e 0.99 no *Fold 4*. Esses valores indicam que o modelo conseguiu identificar a maior parte das amostras, especialmente na classe não infectada.

No entanto, é importante destacar que no *Fold 3* o modelo apresentou uma queda no desempenho, com um *recall* de 0.85 para a classe não infectada e uma precisão de 0.86 para a classe infectada, o que indica uma maior dificuldade em classificar corretamente todas as amostras desse subconjunto. Esse comportamento pode ser um indicativo de que, em certos cenários, o modelo apresenta maior sensibilidade a variações nos dados de entrada, especialmente em subconjuntos mais desafiadores, indicando que o modelo ainda pode ser aprimorado. Ainda assim, o impacto desse resultado foi mitigado pelos desempenhos mais consistentes obtidos nos outros *folds*.

Com base nos resultados apresentados na Tabela 4.1 e na comparação descrita no texto, é possível afirmar que o uso do *cosine decay* demonstrou uma vantagem em relação à utilização de um *learning rate* fixo. Nos experimentos com o *learning rate* fixo, a acurácia variou de forma mais inconsistente entre os *folds*, atingindo valores baixos quanto 72% em um dos casos, o que sugere que o modelo enfrentou dificuldades em manter um desempenho consistente. Em contraste, ao aplicar o *cosine decay*, observou-se uma maior estabilidade nos resultados, com acurácias significativamente mais altas e uma média geral de 94%, indicando uma melhoria notável na capacidade do modelo em generalizar e aprender os padrões de forma mais eficiente.

Além disso, as métricas como *recall*, precisão e F1-score também demonstraram melhor resultado com o *cosine decay*, especialmente para a classe “infectado”, onde os valores ficaram altos e consistentes em quase todos os *folds*. Isso sugere que o uso de uma taxa de aprendizado decrescente, suavizada ao longo do treinamento, ajudou a mitigar o *overfitting*, consequentemente fazendo o modelo convergir de maneira mais eficiente.

Já na segunda parte desse experimento, será avaliado o desempenho do modelo sem o uso de TL. Entretanto, será mantido o uso da validação cruzada e o *cosine decay*, possibilitando uma avaliação mais completa de como a técnica de TL impacta o modelo, pois o desempenho será comparado em termos de acurácia, precisão, *recall* e *F1-Score*. Ao eliminar o TL, espera-se observar uma possível piora nas métricas de desempenho, já que o modelo não terá o benefício de partir de uma base de conhecimento pré-treinada, sendo necessário aprender os padrões diretamente a partir dos dados do conjunto de treino. Essa comparação fornecerá uma visão sobre a importância do TL em um cenário de validação mais rigoroso, permitindo verificar se a técnica contribui para a melhoria do desempenho geral ou se o modelo é capaz de obter resultados similares sem o uso de pesos pré-treinados. Os resultados são apresentados na Tabela 4.2.

Os valores de acurácia obtidos em cada *fold* foram de 59%, 96%, 92%, 89%, 70%,

Tabela 4.2 – Resultados obtidos sem e com *Transfer Learning* (TL) utilizando o modelo EfficientNet-B0 para 5 *Folds*.

Fold	Classe	Sem TL			Com TL		
		Recall	Precisão	F1-score	Recall	Precisão	F1-score
Fold 1	Infectado	0,19	1,00	0,31	0,96	0,96	0,96
	Não infectado	1,00	0,55	0,71	0,96	0,96	0,96
	Média	0,59	0,77	0,51	0,96	0,96	0,96
Fold 2	Infectado	0,95	0,97	0,96	0,94	0,97	0,96
	Não infectado	0,97	0,95	0,96	0,97	0,94	0,96
	Média	0,96	0,96	0,96	0,96	0,96	0,96
Fold 3	Infectado	0,96	0,89	0,93	0,98	0,86	0,92
	Não infectado	0,88	0,96	0,92	0,85	0,98	0,91
	Média	0,92	0,93	0,92	0,91	0,92	0,91
Fold 4	Infectado	0,78	1,00	0,88	0,92	0,99	0,95
	Não infectado	1,00	0,82	0,90	0,99	0,93	0,96
	Média	0,89	0,91	0,89	0,96	0,96	0,96
Fold 5	Infectado	0,39	1,00	0,57	0,91	0,97	0,94
	Não infectado	1,00	0,62	0,77	0,97	0,92	0,94
	Média	0,70	0,81	0,67	0,94	0,94	0,94
Média geral	Infectado	0,65	0,97	0,73	0,94	0,94	0,94
	Não infectado	0,97	0,78	0,85	0,94	0,94	0,94
	Média	0,81	0,87	0,79	0,94	0,94	0,94

Fonte: próprio autor.

respectivamente, com uma média geral entre os *folds* de 81%, representando uma queda em relação aos 94% observados no experimento anterior. Os resultados das outras métricas podem ser observados na Tabela 4.2. É possível observar que no *Fold* 1, o *Recall* para a classe infectada foi de apenas 0.19, indicando uma dificuldade do modelo em identificar corretamente amostras infectadas nesse cenário. A precisão, contudo, atingiu 1.00 para a classe infectada, sugerindo que as amostras identificadas como positivas eram todas verdadeiramente infectadas, mas muitas amostras não foram corretamente identificadas. Os *folds* seguintes, especialmente os *folds* 2 e 3, apresentaram um desempenho mais equilibrado, com valores elevados tanto de *Recall* quanto de Precisão para ambas as classes, o que apresenta uma maior capacidade de generalização do modelo nesses casos. Ainda assim, a média geral de *Recall* para a classe infectada foi de 0.65, o que indica que, em alguns cenários, o modelo teve dificuldades em detectar corretamente todas as amostras infectadas, impactando o *F1-Score* médio, que foi de 0.73. Para a classe não infectada, o desempenho foi mais estável, com um *Recall* médio de 0.97, mas a precisão foi um pouco inferior, com uma média de 0.78, evidenciando que, embora o modelo identificasse a maioria das amostras não infectadas, algumas delas foram classificadas erroneamente.

Diante dos resultados apresentados, é possível concluir que o uso de TL foi importante na melhoria do desempenho do modelo, especialmente na capacidade de generalizar e classificar amostras de ambas as classes de maneira mais consistente. A ausência de TL resultou em um modelo com menor acurácia e grandes variações em métricas como *recall* e *F1-Score*. Assim,

os resultados sugerem que, para o cenário de classificação de malária, a utilização de pesos pré-treinados melhora seu desempenho em um ambiente de validação mais rigoroso.

4.3 Avaliação do *Transfer Learning* (TL)

Nesta seção serão apresentados dois tipos de experimentos: um sem a utilização da técnica de TL e outro utilizando essa técnica. No primeiro experimento, o modelo será treinado apenas com a arquitetura da EfficientNet-B0, sem os pesos previamente ajustados da ImageNet. No segundo experimento, a mesma arquitetura será utilizada, porém com pesos pré-treinados na ImageNet. Em ambos os casos, a base de dados será dividida de forma estratificada em 70% para treino, 15% para validação e 15% para teste, seguindo a mesma divisão utilizada no estudo de Minarno et al. (2023). Os resultados obtidos serão comparados entre si e também com o trabalho de Minarno et al. (2023), que atingiu uma acurácia de 97% no melhor cenário ao classificar as células de RBC contendo o vírus da malária.

No primeiro experimento, onde TL não foi aplicado, a acurácia obtida foi de 83%. Já o valor de *recall*, que reflete a capacidade do modelo de identificar corretamente os casos positivos (infectados, neste caso), foi obtido um valor de 70%. Isso indica que, de todos os casos que realmente deveriam ser classificados como infectados, apenas 70% foram identificados corretamente. Isso sugere que o modelo falhou em detectar 30% dos casos infectados, classificando-os erroneamente como não infectados, o que compromete a sua eficácia.

Em um contexto médico, essa falha pode ter consequências graves, já que falsos negativos (ou seja, casos infectados classificados como não infectados) significam que pessoas doentes podem não receber o tratamento necessário. Portanto, apesar de uma precisão alta (99%), que indica que quase todos os casos que o modelo classificou como infectados realmente estavam infectados, o baixo *recall* significa que o modelo está deixando de detectar uma parte significativa dos casos que precisariam de intervenção.

No segundo experimento, utilizando TL com a mesma arquitetura EfficientNet-B0, porém com pesos pré-treinados na ImageNet, os resultados obtidos foram significativamente diferentes do primeiro experimento.

Neste experimento com TL, a acurácia do modelo aumentou para 96%. O *recall*, que representa a capacidade do modelo de identificar corretamente os casos positivos (infectados), também alcançou 96%. Isso indica uma melhoria em comparação com o experimento anterior, onde a acurácia era de 83% e o *recall* de apenas 70%. Em um contexto médico, essa melhoria é crucial. Enquanto no primeiro experimento o modelo deixava de detectar 30% dos casos infectados, agora com TL, a taxa de detecção aumentou consideravelmente. Isso significa que o modelo agora classifica corretamente quase todos os casos, reduzindo o risco de falsos negativos.

4.4 Avaliação do *Data Augmentation*

Nesta seção será aplicada a técnica de *Data Augmentation* seguindo a aumentação realizada pelo trabalho de [Minarno et al. \(2023\)](#). A divisão da base de dados será feita de forma estratificada em 70% para treino, 15% para validação e 15% para teste, seguindo a mesma divisão utilizada no estudo de [Minarno et al. \(2023\)](#). O intuito é avaliar o impacto da técnica no modelo proposto, na tentativa de melhorar os resultados obtidos. Vale ressaltar que o melhor modelo obtido na seção anterior foi aplicado para os experimentos desta seção.

A [Tabela 4.3](#) apresenta os parâmetros utilizados no processo de aumento de dados. Essas configurações incluem:

- ***Horizontal Flip e Vertical Flip***: Habilitados para permitir que as imagens sejam invertidas horizontal e verticalmente, respectivamente, gerando variações de orientação.
- ***Rotation Range***: Definido como 90 graus, permitindo que as imagens sejam rotacionadas em um intervalo de até 90° em qualquer direção.
- ***Height Shift Range e Width Shift Range***: Configurados em 0,2, permitindo deslocamentos verticais e horizontais equivalentes a 20% das dimensões da imagem.
- ***Zoom Range***: Também ajustado para 0,2, introduzindo variações de zoom de até 20% para aumentar a diversidade das escalas nas imagens.

Tabela 4.3 – Parâmetros usados para realizar o *data augmentation*.

Parâmetro	Valor
<i>Horizontal Flip</i>	<i>True</i>
<i>Vertical Flip</i>	<i>True</i>
<i>Rotation Range</i>	90
<i>Height Shift Range</i>	0,2
<i>Width Shift Range</i>	0,2
<i>Zoom Range</i>	0,2

Fonte: próprio autor.

Os resultados apresentados na [Tabela 4.4](#) demonstram que a aplicação da técnica de aumento de dados não resultou em um impacto significativo nas principais métricas de desempenho. Sem *data augmentation*, o modelo alcançou uma acurácia de 96%, enquanto, com a aplicação da técnica, a acurácia permaneceu no intervalo de 94% a 96%, indicando uma diferença mínima. De forma semelhante, as métricas de precisão, *recall* e F1-score mostraram valores praticamente equivalentes. Isso sugere que o conjunto de dados utilizado já continha variações suficientes para treinar o modelo de forma eficiente, limitando os benefícios adicionais do *data augmentation* para este caso específico.

Tabela 4.4 – Resultados do modelo proposto com e sem *data augmentation*.

Configuração	Recall	Precisão	F1-score	Acurácia
Sem <i>data augmentation</i>	0,96	0,96	0,96	0,96
Com <i>data augmentation</i>	0,95	0,95	0,95	0,95

Fonte: próprio autor.

4.5 Avaliação do *Super Resolution* (SR)

Nesta seção, será aplicada a técnica de SR utilizando a implementação do ESRGAN disponibilizada pelo *TensorFlow*, que já possui pesos pré-treinados. O objetivo da aplicação dessa técnica no contexto deste trabalho foi tentar compensar a resolução das imagens do conjunto de dados, que contém amostras de RBC. Como são imagens de células, a expectativa era que o ESRGAN pudesse recuperar detalhes sutis (WANG et al., 2018), tornando a distinção entre células saudáveis e infectadas mais evidente, e assim potencialmente melhorando a capacidade de generalização do classificador.

Para avaliar o desempenho do SR nesse cenário, as imagens foram processadas pelo ESRGAN, redimensionadas para um tamanho de 100×100 pixels, e utilizadas para treinar a rede EfficientNet-B0. Além disso, foram testados três otimizadores distintos: *Adam*, *RMSprop* e *SGD*, buscando identificar possíveis variações no desempenho do modelo. A Tabela 4.5 apresenta os resultados obtidos.

Tabela 4.5 – Resultados do modelo proposto com diferentes otimizadores para SR.

Otimizador	Recall	Precisão	F1-score	Acurácia
<i>Adam</i>	0,96	0,96	0,96	0,96
<i>RMSprop</i>	0,96	0,96	0,96	0,96
<i>SGD</i>	0,96	0,96	0,96	0,96

Fonte: próprio autor.

Os resultados indicam que, apesar da aplicação do SR, não houve melhorias no desempenho do modelo. Foi possível observar uma menor variabilidade dos resultados, já que, enquanto nas outras técnicas empregadas a acurácia se mantinha na faixa de 94% a 96%, com o SR todas as métricas permaneceram constantes em 96% para *recall*, *precisão*, *F1-score* e *acurácia*, independentemente do otimizador utilizado. Esse comportamento sugere que a resolução original das imagens já era suficiente para que o modelo conseguisse extrair as características relevantes para a classificação, limitando os benefícios do ESRGAN neste estudo.

4.6 Avaliação do *Vision Transformer* (ViT)

Nesta seção, serão apresentados os resultados obtidos com o treinamento do modelo ViT. Nesse experimento, as imagens utilizadas passaram por um pré-processamento utilizando a

técnica de SR. Essa abordagem tem como objetivo potencializar o desempenho do ViT, permitindo uma melhor extração de características relevantes para a classificação (WANG et al., 2018).

Os experimentos foram conduzidos utilizando o otimizador *Adam* e variando o *learning rate*, sendo avaliadas duas configurações principais: 10^{-3} e 10^{-4} . Os resultados obtidos em termos de *recall*, precisão, *F1-score* e acurácia são apresentados na Tabela 4.6.

Tabela 4.6 – Resultados obtidos com o treinamento do ViT.

Learning Rate	Recall	Precisão	F1-score	Acurácia
0,001	0,97	0,95	0,96	0,96
0,0001	0,98	0,95	0,97	0,97

Fonte: próprio autor.

Em ambas as configurações testadas, o ViT obteve um desempenho interessante em comparação com as arquiteturas e técnicas previamente analisadas. Para a taxa de aprendizado de 0,001, o modelo alcançou uma acurácia de 96%, com *recall* de 97%. Já para a taxa de aprendizado de 0,0001, observou-se um leve incremento no desempenho, atingindo uma acurácia de 97% e um *recall* de 98%. Esses resultados indicam que, além de uma melhora na classificação geral, o modelo demonstrou uma capacidade elevada de identificar corretamente os casos positivos (infectados), reduzindo a ocorrência de falsos negativos.

Essa melhora no *recall* é um aspecto crucial no contexto médico, pois implica que menos casos de pacientes infectados estão sendo erroneamente classificados como não infectados. Dessa forma, o uso do ViT combinado com o pré-processamento de SR demonstrou ser a abordagem mais eficaz dentre todas as testadas.

4.7 Comparação com a literatura

A Tabela 4.7 apresenta a comparação entre os modelos testados e aqueles encontrados na literatura. Observa-se que o modelo baseado em ViT alcançou um desempenho equivalente ao estudo de Minarno et al. (2023) em todas as métricas analisadas. Isso reforça a robustez dessa abordagem para a classificação de malária, uma vez que os resultados obtidos estão alinhados com um dos principais trabalhos de referência na área, além de que o valor de *recall* foi superior a todos os outros modelos comparados.

Tabela 4.7 – Comparação entre modelos da literatura com os modelos testados.

Trabalho/Modelo	Recall	Precisão	F1-score	Acurácia
Minarno et al. (2023)	0,97	0,97	0,97	0,97
Reddy e Juliet (2019)	-	-	-	0,95
EfficientNet-B0	0,96	0,96	0,96	0,96
ViT	0,98	0,95	0,97	0,97

Fonte: próprio autor.

Já o modelo baseado em EfficientNet-B0 demonstrou um desempenho competitivo, com métricas muito próximas às do ViT e do estudo de Minarno et al. (2023), mas com a vantagem de ser uma alternativa mais eficiente em termos computacionais. Essa característica faz com que ele seja uma solução interessante para aplicações que exigem um equilíbrio entre precisão e custo computacional.

Em relação ao estudo de Reddy e Juliet (2019), que reportou uma acurácia de 95%, todos os modelos avaliados apresentaram resultados superiores. No entanto, como esse estudo não disponibiliza outras métricas de desempenho, uma comparação mais detalhada é limitada.

Dessa forma, os resultados indicam que tanto o ViT quanto o EfficientNet-B0 são alternativas viáveis para a classificação de malária, oferecendo precisão competitiva em relação à literatura. Ainda que existam oportunidades para melhorias, como a utilização de *data augmentation* para o treinamento do ViT - como feito no trabalho de Sabir et al. (2023) -, além da validação do SR considerando seu impacto no tempo de processamento e a análise do desempenho do ViT sem essa técnica, os modelos testados demonstram grande potencial como ferramentas eficientes para essa tarefa.

5 Considerações Finais

Neste capítulo serão apresentadas as conclusões alcançadas com o desenvolvimento desse trabalho, levando em consideração os objetivos que foram almeçados no [Capítulo 1](#). Além disso, serão apresentadas propostas para a continuação do trabalho, a partir do que já foi desenvolvido.

5.1 Conclusão

Neste trabalho, foi proposto um modelo para a classificação de malária em [RBC](#), utilizando como base a [CNN EfficientNet-B0](#). O principal objetivo deste trabalho era avaliar se a rede era capaz de desempenhar bem quando comparada a outros trabalhos, como o de [Minarno et al. \(2023\)](#), por exemplo. Um dos objetivos que foi almeçado era a escolha de uma base de dados adequada para o problema, cuja a escolhida foi a de [Rajaraman et al. \(2018\)](#), contendo uma quantidade suficiente e equilibrada de dados. Além disso, o desenvolvimento e os testes de desempenho do modelo na tarefa que se deseja realizar e a avaliação e comparação dos resultados obtidos.

A aplicação do *cosine decay* foi um fator que colaborou para o desenvolvimento do trabalho e os resultados obtidos. O *learning rate* fixo estava apresentando sinais de *overfitting*, fazendo com que o modelo ficasse preso em mínimos locais e perdesse a capacidade de generalizar. O *learning rate* dinâmico se mostrou eficiente e isso foi evidenciado nos testes realizados.

Também foi possível observar que o uso da técnica de [TL](#) foi importante para o desempenho do modelo. Com valores de acurácia de 94% e 96%, respectivamente com o uso da validação cruzada estratificada e sem ela, os resultados se mostraram superiores e mais equilibrados quando aplicado o [TL](#). Além disso, métricas importantes como o *recall* também foram superiores com o uso de [TL](#), o que é importante quando colocado no contexto do trabalho, onde é interessante reduzir o número de falsos negativos.

Em relação às técnicas de *Data Augmentation* e *Super Resolution (SR)*, os resultados apresentaram comportamentos semelhantes. A aplicação de *Data Augmentation*, seguindo a metodologia de [Minarno et al. \(2023\)](#), não resultou em melhorias significativas no desempenho do modelo. Isso sugere que o conjunto de dados utilizado já continha variações suficientes para treinar o modelo.

Por outro lado, a aplicação de [SR](#) utilizando o [ESRGAN](#), apesar de não resultar em melhorias significativas no desempenho do modelo, foi observada uma menor variabilidade nos resultados. Apesar da expectativa de que esta técnica pudesse recuperar detalhes sutis nas imagens de [RBC](#) e tornar mais evidente a distinção entre células saudáveis e infectadas, os resultados permaneceram constantes em 96% para *recall*, precisão, *F1-score* e acurácia, independentemente

do otimizador utilizado. Este comportamento sugere que a resolução original das imagens já era suficiente para que o modelo extraísse as características relevantes para a classificação, limitando os benefícios do *ESRGAN* neste estudo específico.

Os resultados obtidos com o *Vision Transformer* (ViT) foram positivos, confirmando a expectativa de um bom desempenho na tarefa de classificação de imagens, conforme indicado por estudos anteriores (DOSOVITSKIY et al., 2020). Devido à sua capacidade de modelar relações globais dentro da imagem, esperava-se que essa arquitetura também fosse eficaz na classificação de malária. O desempenho alcançado, com uma acurácia de 97%, demonstra a eficiência do modelo, uma vez que os resultados foram equiparáveis aos obtidos por Minarno et al. (2023), que representam o estado da arte na literatura. Esses achados reforçam o potencial do ViT como uma alternativa para a classificação de glóbulos vermelhos infectados, ainda que seu maior custo computacional deva ser considerado na escolha da melhor abordagem para aplicações práticas.

Além disso, quando comparados os resultados obtidos com os trabalhos de Minarno et al. (2023) e Reddy e Juliet (2019), foi possível observar um desempenho positivo dos modelos propostos. Os resultados obtidos estão equiparáveis ao estado-da-arte mostrando que os modelos propostos são capazes de desempenhar bem na tarefa de classificação de malária.

5.2 Trabalhos Futuros

Os resultados obtidos neste estudo abrem margem para possíveis estudos futuros. Uma ideia possível é o treinamento de um modelo específico de *Super Resolution* voltado para células sanguíneas, com o objetivo de aprimorar ainda mais a qualidade das imagens antes da etapa de classificação. Essa abordagem permitiria avaliar se um modelo ajustado para esse contexto específico pode impactar positivamente no desempenho da classificação (WANG et al., 2018).

Outra possibilidade é a avaliação do impacto de técnicas de *Data Augmentation* no desempenho do *Vision Transformer*. Considerando que essa arquitetura se baseia na modelagem de relações globais dentro da imagem (DOSOVITSKIY et al., 2020), é interessante investigar se o aumento da diversidade da base pode contribuir para uma melhor generalização do modelo.

Além disso, explorar arquiteturas mais complexas para a tarefa de classificação de malária, como variantes do ViT mais otimizadas para eficiência computacional (TOLSTIKHIN et al., 2021) ou híbridos entre redes convolucionais e *Transformers*, que têm demonstrado desempenho promissor na área de visão computacional (KHAN et al., 2022).

Por fim, outra linha de investigação relevante seria a análise do impacto da aplicação dessas técnicas em cenários do mundo real, como a implementação dos modelos em dispositivos de baixo custo para uso em clínicas de países endêmicos. Testes em ambientes com restrições computacionais poderiam fornecer informações valiosas sobre a viabilidade dessas soluções e sua aplicação prática na classificação de malária.

Referências

- BATISTA, G. T.; NASCIMENTO, P. S. R.; FILHO, R. A. Efeito de pré-processamento (filtro mediana) no desempenho da segmentação e classificação de imagens landsat-tm. In: Simpósio Latino-americano de Percepção Remota. [S.l.: s.n.], 1998. v. 8, p. 569–570.
- BELL, S.; UPCHURCH, P.; SNAVELY, N.; BALA, K. Material recognition in the wild with the materials in context database. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2015. p. 3479–3487.
- BOOK, D. L. Capítulo 10 – As 10 Principais Arquiteturas de Redes Neurais. 2024. <<https://www.deeplearningbook.com.br/as-10-principais-arquiteturas-de-redes-neurais/>>. Acesso em julho de 2024.
- BUDHIRAJA, A. Dropout in (Deep) Machine learning. 2016. <<https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5>>. Acesso em agosto de 2024.
- CANDIDO, G. Overfitting vs Underfitting. 2023. <<https://medium.com/@lg0702/overfitting-e-underfitting-6cba339aebfb>>. Acesso em setembro de 2024.
- CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. A discussion of semi-supervised learning and transduction. In: Semi-supervised learning. [S.l.]: MIT Press, 2006. p. 473–478.
- ÇINAR, A.; YILDIRIM, M. Classification of malaria cell images with deep learning architectures. Ingénierie des Systèmes d’Information, International Information and Engineering Technology Association (IIETA), v. 25, n. 1, p. 35, 2020.
- CORREA, S. Cosine Learning rate decay. 2019. <<https://scorrea92.medium.com/cosine-learning-rate-decay-e8b50aa455b>>. Acesso em setembro de 2024.
- CORTES, C.; MOHRI, M.; ROSTAMIZADEH, A. L2 regularization for learning kernels. arXiv preprint arXiv:1205.2653, 2012.
- datacamp. A Complete Guide to Data Augmentation. 2022. Acesso em dezembro 2024. Disponível em: <<https://www.datacamp.com/tutorial/complete-guide-data-augmentation>>.
- DIAZ, I. B. Atlas to patient registration with brain tumor based on a new mesh-free method. 2015.
- DONG, C.; LOY, C. C.; HE, K.; TANG, X. Image super-resolution using deep convolutional networks. In: IEEE. IEEE transactions on pattern analysis and machine intelligence. [S.l.], 2015.
- DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S. et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.
- Eduardo Marinho. Convidado CONTESTA ARGUMENTO do Igor. 2022. Acesso em setembro 2024 via YouTube. Disponível em: <<https://www.youtube.com/watch?v=V7qwwCe5oB0>>.

- ENGELEN, J. E. V.; HOOS, H. H. A survey on semi-supervised learning. Machine learning, Springer, v. 109, n. 2, p. 373–440, 2020.
- GeeksforGeeks. Backpropagation in Neural Network. 2024. <<https://www.geeksforgeeks.org/backpropagation-in-neural-network/>>. Acesso em julho de 2024.
- GILLIS, A. S. transfer learning. 2024. <<https://www.techtarget.com/searchcio/definition/transfer-learning>>. Acesso em julho de 2024.
- HAN, J.; MORAGA, C. The influence of the sigmoid function parameters on the speed of backpropagation learning. In: SPRINGER. International workshop on artificial neural networks. [S.l.], 1995. p. 195–201.
- HARAHAP, M.; JEFFERSON, J.; BARTI, S.; SAMOSIR, S.; TURNIP, C. A. Implementation of convolutional neural network in the classification of red blood cells have affected of malaria. Sinkron: jurnal dan penelitian teknik informatika, v. 5, n. 2, p. 199–207, 2021.
- HU, J.; SHEN, L.; SUN, G. Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2018. p. 7132–7141.
- Hugging Face. vit-base-patch16-224-in21k. 2020. Acesso em março de 2025. Disponível em: <<https://huggingface.co/google/vit-base-patch16-224-in21k>>.
- IBM. O que é uma rede neural? 2024. <<https://www.ibm.com/br-pt/topics/neural-networks>>. Acesso em junho de 2024.
- JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. Electronic Markets, Springer, v. 31, n. 3, p. 685–695, 2021.
- JESUS, E. O.; JR, R. C. A utilização de filtros gaussianos na análise de imagens digitais. Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 3, n. 1, 2015.
- JOLICOEUR-MARTINEAU, A. The relativistic discriminator: a key element missing from standard gan. arXiv preprint arXiv:1807.00734, 2018.
- KANNA, R. K.; PANIGRAHI, B. S.; SAHOO, S. K.; REDDY, A. R.; MANCHALA, Y.; SWAIN, N. K. Cnn based face emotion recognition system for healthcare application. EAI Endorsed Transactions on Pervasive Health and Technology, v. 10, 2024. Acesso em julho de 2024.
- KHAN, S.; NASEER, M.; HAYAT, M.; ZAMIR, S. W.; KHAN, F. S.; SHAH, M. Transformers in vision: A survey. ACM computing surveys (CSUR), ACM New York, NY, v. 54, n. 10s, p. 1–41, 2022.
- KLINGLER, N. EfficientNet: Optimizing Deep Learning Efficiency. 2024. <<https://viso.ai/deep-learning/efficientnet/>>. Acesso em setembro de 2024.
- KRICHEN, M. Convolutional neural networks: A survey. Computers, MDPI, v. 12, n. 8, p. 151, 2023.
- KUNDU, R. Super-Resolution. 2022. Acesso em fevereiro de 2025. Disponível em: <<https://heartbeat.comet.ml/super-resolution-580424fb7f7c>>.
- LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. nature, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015.

- LEDIG, C.; THEIS, L.; HUSZÁR, F.; CABALLERO, J.; CUNNINGHAM, A.; ACOSTA, A.; AITKEN, A.; TEJANI, A.; TOTZ, J.; WANG, Z. et al. Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2017. p. 4681–4690.
- LIANG, J.; CAO, J.; SUN, G.; ZHANG, K.; GOOL, L. V.; TIMOFTE, R. Swinir: Image restoration using swin transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. [S.l.: s.n.], 2021. p. 1833–1844.
- LINI, R. Laplacian of Gaussian Filter (LoG) for Image Processing. 2023. <<https://medium.com/@rajilini/laplacian-of-gaussian-filter-log-for-image-processing-c2d1659d5d2>>. Acesso em setembro de 2024.
- MINARNO, A. E.; ARIPA, L.; AZHAR, Y.; MUNARKO, Y. Classification of malaria cell image using inception-v3 architecture. JOIV: International Journal on Informatics Visualization, v. 7, n. 2, p. 273–278, 2023.
- MONTALBO, F. J.; D.ENG, A. S. A. Empirical analysis of a fine-tuned deep convolutional model in classifying and detecting malaria parasites from blood smears. KSII Transactions on Internet and Information Systems, v. 15, p. 147–165, 01 2021.
- NARAYANAN, B. N.; ALI, R.; HARDIE, R. C. Performance analysis of machine learning and deep learning architectures for malaria detection on cell images. In: SPIE. Applications of Machine Learning. [S.l.], 2019. v. 11139, p. 240–247.
- NGUYEN, Q. H.; NGUYEN, B. P.; DAO, S. D.; UNNIKISHNAN, B.; DHINGRA, R.; RAVICHANDRAN, S. R.; SATPATHY, S.; RAJA, P. N.; CHUA, M. C. Deep learning models for tuberculosis detection from chest x-ray images. In: IEEE. 2019 26th international conference on telecommunications (ICT). [S.l.], 2019. p. 381–385.
- O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458, 2015.
- PANNU, A. Artificial intelligence and its application in different areas. Artificial Intelligence, v. 4, n. 10, p. 79–84, 2015.
- PEREIRA, R. O que é inteligência artificial? 2023. <<https://medium.com/@rodolfopereira.ai/o-que-é-inteligência-artificial-dcdcac396e8a>>. Acesso em maio de 2024.
- POLONI, K. Redes neurais convolucionais. 2022. <<https://medium.com/itau-data/redes-neurais-convolucionais-2206a089c715>>. Acesso em julho de 2024.
- PRUSTY, S.; PATNAIK, S.; DASH, S. K. Skcv: Stratified k-fold cross-validation on ml classifiers for predicting cervical cancer. Frontiers in Nanotechnology, Frontiers Media SA, v. 4, p. 972421, 2022.
- PyTorch. Sigmoid. 2024. Acesso em agosto de 2024. Disponível em: <<https://pytorch.org/docs/stable/generated/torch.nn.Sigmoid.html#torch.nn.Sigmoid>>.
- RAGHU, M.; UNTERTHINER, T.; KORNBLITH, S.; ZHANG, C.; DOSOVITSKIY, A. Do vision transformers see like convolutional neural networks? Advances in neural information processing systems, v. 34, p. 12116–12128, 2021.

- RAJARAMAN, S.; ANTANI, S. K.; POOSTCHI, M.; SILAMUT, K.; HOSSAIN, M. A.; MAUDE, R. J.; JAEGER, S.; THOMA, G. R. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. PeerJ, PeerJ Inc., v. 6, p. e4568, 2018.
- RAY, S. What Is Data Augmentation? 2021. <<https://medium.com/lansaar/what-is-data-augmentation-3da1373e3fa1>>. Acesso em maio de 2024.
- REDDY, A. S. B.; JULIET, D. S. Transfer learning with resnet-50 for malaria cell-image classification. In: IEEE. 2019 International conference on communication and signal processing (ICCSP). [S.l.], 2019. p. 0945–0949.
- REIS-SILVA, V. F. Dcnv-19: A deep convolutional neural network for covid-19 detection in chest computed tomographies. arXiv preprint arXiv:2208.09349, 2022.
- RIBANI, R.; MARENGONI, M. A survey of transfer learning for convolutional neural networks. In: IEEE. 2019 32nd SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T). [S.l.], 2019. p. 47–57.
- ROJAS, R.; ROJAS, R. The backpropagation algorithm. Neural networks: a systematic introduction, Springer, p. 149–182, 1996.
- SABIR, M. W.; FARHAN, M.; ALMALKI, N. S.; ALNFIAI, M. M.; SAMPEDRO, G. A. Fibrovit—vision transformer-based framework for detection and classification of pulmonary fibrosis from chest ct images. Frontiers in medicine, Frontiers Media SA, v. 10, p. 1282200, 2023.
- SANDLER, M.; HOWARD, A.; ZHU, M.; ZHMOGINOV, A.; CHEN, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2018. p. 4510–4520.
- SARKAR, A. Understanding EfficientNet — The most powerful CNN architecture. 2021. <<https://arjun-sarkar786.medium.com/understanding-efficientnet-the-most-powerful-cnn-architecture-eaeb40386fad>>. Acesso em junho de 2024.
- SAXENA shipra. Binary Cross Entropy/Log Loss for Binary Classification. 2024. <<https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/>>. Acesso em setembro de 2024.
- Secretaria de Estado da Saúde do Pará. O que é a malária? 2023. <<http://www.saude.pa.gov.br/a-secretaria/diretorias/dvs/malaria/o-que-e-malaria/>>. Acesso em agosto de 2024.
- SHAH, H. A.; SAEED, F.; YUN, S.; PARK, J.-H.; PAUL, A.; KANG, J.-M. A robust approach for brain tumor detection in magnetic resonance images using finetuned efficientnet. Ieee Access, IEEE, v. 10, p. 65426–65438, 2022.
- SHEHZADI, T.; STRICKER, D.; AFZAL, M. Z. et al. Semi-supervised object detection: A survey on progress from cnn to transformer. arXiv preprint arXiv:2407.08460, 2024.
- SHEKAR, G.; REVATHY, S.; GOUD, E. K. Malaria detection using deep learning. In: IEEE. 2020 4th international conference on trends in electronics and informatics (ICOEI)(48184). [S.l.], 2020. p. 746–750.

- SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. Journal of big data, Springer, v. 6, n. 1, p. 1–48, 2019.
- TAN, M.; LE, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In: PMLR. International conference on machine learning. [S.l.], 2019. p. 6105–6114.
- TOLSTIKHIN, I. O.; HOULSBY, N.; KOLESNIKOV, A.; BEYER, L.; ZHAI, X.; UNTERTHINER, T.; YUNG, J.; STEINER, A.; KEYSERS, D.; USZKOREIT, J. et al. Mlp-mixer: An all-mlp architecture for vision. Advances in neural information processing systems, v. 34, p. 24261–24272, 2021.
- TOUVRON, H.; CORD, M.; DOUZE, M.; MASSA, F.; SABLAYROLLES, A.; JÉGOU, H. Training data-efficient image transformers & distillation through attention. In: PMLR. International conference on machine learning. [S.l.], 2021. p. 10347–10357.
- VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. Advances in neural information processing systems, v. 30, 2017.
- Velog. KFold and StratifiedKFold. 2023. Acesso em agosto de 2024. Disponível em: <<https://velog.io/@sae0912/KFold-êµŘří-êšÄìçĴKFold-StratifiedKFold>>.
- WANG, Q.; MA, Y.; ZHAO, K.; TIAN, Y. A comprehensive survey of loss functions in machine learning. Annals of Data Science, Springer, p. 1–26, 2020.
- WANG, X.; YU, K.; WU, S.; GU, J.; LIU, Y.; DONG, C.; QIAO, Y.; LOY, C. C. Esrgan: Enhanced super-resolution generative adversarial networks. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops. [S.l.: s.n.], 2018. p. 0–0.
- XIE, E.; WANG, W.; YU, Z.; ANANDKUMAR, A.; ALVAREZ, J. M.; LUO, P. Segformer: Simple and efficient design for semantic segmentation with transformers. Advances in neural information processing systems, v. 34, p. 12077–12090, 2021.
- YACOUBY, R.; AXMAN, D. Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In: Proceedings of the first workshop on evaluation and comparison of NLP systems. [S.l.: s.n.], 2020. p. 79–91.
- YAO, W.; BAI, J.; LIAO, W.; CHEN, Y.; LIU, M.; XIE, Y. From cnn to transformer: A review of medical image segmentation models. Journal of Imaging Informatics in Medicine, Springer, p. 1–19, 2024.
- ZAIDI, A. Mathematical justification on the origin of the sigmoid in logistic regression. Central European Management Journal, v. 30, n. 4, p. 1327–1337, 2022.