UNIVERSIDADE FEDERAL DO CEARÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

ANDRESSA GOMES MOREIRA

ANÁLISE DO DESEMPENHO DE REDES ADVERSÁRIAS GENERATIVAS COMO UM MÉTODO DE AUMENTO DE DADOS DE IMAGENS DA MUCOSA OCULAR DE OVINOS

SOBRAL

2022

ANDRESSA GOMES MOREIRA

ANÁLISE DO DESEMPENHO DE REDES ADVERSÁRIAS GENERATIVAS COMO UM MÉTODO DE AUMENTO DE DADOS DE IMAGENS DA MUCOSA OCULAR DE OVINOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Orientador: Prof. Dr. Iális Cavalcante de Paula Júnior

SOBRAL

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

M836a Moreira, Andressa Gomes.
Análise do desempenho de Redes Adversárias Generativas como um método de aumento de dados de imagens da mucosa ocular de ovinos / Andressa Gomes Moreira. – 2022.
75 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2022.
Orientação: Prof. Dr. Iális Cavalcante de Paula Júnior.

1. Redes Adversárias Generativas. 2. Aumento de dados. 3. Classificação. 4. Redes Neurais Convolucionais. I. Título.

CDD 621.39

ANDRESSA GOMES MOREIRA

ANÁLISE DO DESEMPENHO DE REDES ADVERSÁRIAS GENERATIVAS COMO UM MÉTODO DE AUMENTO DE DADOS DE IMAGENS DA MUCOSA OCULAR DE OVINOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Aprovada em: 14 de Julho de 2022

BANCA EXAMINADORA

Prof. Dr. Iális Cavalcante de Paula Júnior (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Marcelo Marques Simões de Souza
Universidade Federal do Ceará (UFC)

Prof. Dr. Reuber Régis de Melo
Universidade Federal do Ceará (UFC)

Dedico este trabalho aos meus pais por acreditarem em mim. Mãe, o seu incentivo, os seus cuidados e a sua fé me deram a força necessária para concluir esta etapa. Pai, seu apoio me ajudou a seguir em frente.

AGRADECIMENTOS

Agradeço a Deus pelo dom da vida e por restaurar minhas forças todas as vezes que pensei em desistir.

Aos meus pais por cuidarem de mim e acreditarem que seria possível.

Ao Prof. Dr. Iális Cavalcante de Paula Júnior por me orientar durante toda a graduação. Agradeço pelos ensinamentos, pela paciência, e principalmente, por ser um modelo de profissional que quero seguir.

Ao meu namorado Carlos Augusto pela paciência e pelos cuidados.

À minha amiga Stefane, que me ajudou no desenvolvimento deste trabalho e compartilhou comigo às aflições durante a preparação deste projeto.

Aos amigos Rony, Fabrício, Michel, Pedrosa e Israel, que alegraram meus dias durante a graduação.

Ao colega Antonio Marcio pela disponibilização das imagens utilizadas neste projeto.

Aos colegas do grupo de pesquisa MINSKY pelos conhecimentos compartilhados.

Ao Programa Educacional de Educação Tutorial (PET) por me propocionar a oportunidade de iniciar este projeto de pesquisa.

Aos professores da Universidade Federal do Ceará (UFC) por todos os ensinamentos.

“O Senhor é a minha luz e a minha salvação; a quem temerei? O Senhor é o meu forte refúgio; de quem me recearei? "

(Salmos 27:1)

RESUMO

Tarefas de classificação de imagens, realizadas por redes neurais profundas, exigem um grande volume de dados de treinamento para alcançar o aprendizado efetivo do modelo. Entretanto, fatores como a escassez de informações afetam o desempenho do classificador, visto que os dados na maioria das vezes são formados por poucas imagens ou não estão disponíveis publicamente para uso. Nesse contexto, para expandir de forma significativa a base de dados e mitigar a falta de dados rotulados, são utilizados os métodos de Aumento de Dados (DA). A priori, a aplicação de Transformações de Imagens no conjunto de treinamento é uma técnica amplamente utilizada. Ademais, uma nova abordagem é a utilização de *Generative Adversarial Network* (*GAN*), capaz de gerar dados sintéticos plausíveis a partir dos dados originais. Portanto, este trabalho propõe a análise da eficiência de uma *GAN* como um método de DA. Dessa forma, foi realizado o aumento do número de instâncias da base dados de imagens da mucosa ocular de ovinos, utilizadas no desenvolvimento de um sistema computacional para a detecção da anemia nesses animais. Dessarte, os modelos de classificação EfficienteNetB0V2, InceptionV3 e ResNet101V2 foram treinados com base no conjunto de dados original, em seguida, pelo conjunto de treinamento expandido por técnicas de aumento de dados tradicionais e, por fim, com os dados de treinamento aumentados com as imagens geradas pela *GAN*. Portanto, o desempenho de todos os modelos utilizados neste trabalho apresentou uma melhora significativa em valores de acurácia média de validação ao serem treinados com a base de dados expandida com imagens geradas pela *GAN*.

Palavras-chave: Redes Adversárias Generativas; Aumento de dados; Classificação; Redes Neurais Convolucionais.

ABSTRACT

Image classification tasks, performed by deep neural networks, require a large volume of training data to achieve effective model learning. However, factors such as the scarcity of information affect the performance of the classifier, since the data most of the times are formed by few images or are not publicly available for use. In this context, to significantly expand the database and mitigate the lack of labeled data, the Data Augmentation (DA) methods are used. In principle, the application of Image Transformations in the training set is a widely used technique. Furthermore, a new approach is the use of the e Generative Adversarial Network (GAN), capable of generating plausible synthetic data from the original data. Therefore, this work proposes the analysis of the efficiency of a *GAN* as a method of DA. In this way, the number of instances in the database of images of the ocular mucosa of sheep was increased, used in the development of a computer system for the detection of anemia in these animals. Thus, the classification models EfficienteNetB0V2, InceptionV3 and ResNet101V2 were trained based on the original dataset, then by the training set expanded by traditional data augmentation techniques and, finally, with the training data augmented with the images. generated by *GAN*. Therefore, the performance of all models used in this work showed a significant improvement in mean validation accuracy values when trained with the expanded database with images generated by *GAN*.

Keywords: Generative Adversarial Networks; Data augmentation; Classification; Convolutional Neural Networks.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DA	Aumento de Dados
GAN	*Generative Adversarial Network*
CNN	*Convolutional Neural Network*
ReLU	Unidade Linear Retificada
StyleGAN	*Style-Based Generator Architecture for Generative Adversarial Networks*
FID	*Frechet Inception Distance*
PPM	Pesquisa Pecuária Municipal
IBGE	Instituto Brasileiro de Geografia e Estatística
ML	*Machine Learning*
AM	Aprendizado de máquina
IA	Inteligência Artificial
RNA	Redes Neurais Artificiais
MLP	*Multilayer Perceptron*
DL	*Deep Learning*
FC	Camada Totalmente Conectada
FP	Falso Positivo
FN	Falso Negativo
VP	Verdadeiro Positivo
VN	Verdadeiro Negativo
EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
UFC	Universidade Federal do Ceará

LISTA DE SÍMBOLOS

x_j	Sinais de entrada de um modelo neural
w_j	Pesos sinápticos de um modelo neural
b	Bias
φ	Função de ativação
y	Sinal de saída de um neurônio
δ_l	Erro associado a um neurônio
L	Função de Perda
α	Taxa de aprendizado
D	Rede Discriminadora
G	Rede Geradora
V	Função Objetivo
p_x	Dados reais de um sistema *GAN*
p_z	Distribuição dos dados da rede geradora
$D(x)$	Valores definidos pelo discriminador D
$G(x)$	Valores definidos pelo gerador G
E_x	Ruído de entrada de um sistema *GAN*
ϕ	Transformações em imagens
S	Conjunto de treinamento original
T	Conjunto expandido de S
S'	Conjunto de treinamento aumentado

SUMÁRIO

1 INTRODUÇÃO

Em 2020, os dados econômicos referentes à pecuária brasileira foram influenciados, entre outros fatores, pela decorrência da pandemia da Covid-19. A Pesquisa Pecuária Municipal (PPM), realizada pelo Instituto Brasileiro de Geografia e Estatística (IBGE), que fornece informações importantes sobre a produção pecuária, afirma que o desdobrar da pandemia e suas medidas restritivas ocasionou o aumento dos preços dos insumos pecuários, em especial de rebanhos caprino e ovino. Dessa forma, o rebanho ovino no Brasil registrou uma taxa de crescimento de 3,3% em relação ao ano de 2019, totalizando 20,6 milhões de cabeças (IBGE, 2020).

A ovinocultura é uma atividade bastante explorada e tem sido uma alternativa de alimentação de diversos brasileiros, principalmente para os nordestinos. A região Nordeste é responsável por 70,6% do total de ovinos do país, o que valida a adaptabilidade das criações com as condições locais. Ademais, a prática dispõe de um elevado potencial para ampliar a produção de leite e derivados, da carne e do setor industrial. (EMBRAPA, 2020; ELOY *et al.*, 2007). Desse modo, tendo em vista o impacto econômico provocado pela criação e manejo de ovinos em âmbito regional e estadual, é de suma importância os cuidados com a saúde desses animais, uma vez que patologias, como a Hemoncose, podem inviabilizar cadeias produtivas e limitar a atividade pecuária, causando prejuízos econômicos em diversas regiões (BIRGEL, 2013).

A Hemoncose é uma verminose que acomete ovinos e caprinos. A enfermidade é provocada pelo parasita *Haemonchus contortus*, que se localiza no abomaso de ruminantes. A doença é disseminada pela ingestão do pasto contaminado com a larva em sua fase infectante. A patologia pode causar perda sanguínea, disfunções na absorção dos nutrientes, diarreia com sangue, quadros intensos de anemia e, em casos mais graves, o óbito do animal (SILVA *et al.*, 2019).

O método FAMACHA, desenvolvido na África do Sul, é um recurso para é identificar diferentes graus de anemia em ovinos, sem o auxílio de exames laboratoriais. Os pesquisadores estabeleceram uma correlação entre a coloração conjuntiva ocular de pequenos ruminantes e cinco intervalos de anemia indicados pelo exame de sangue, hematócrito, que mede a porcentagem de células vermelhas (WYK *et al.*, 1997). Isto posto, o método utiliza o cartão FAMACHA, ilustrado com os cinco graus de coloração, direcionando a vermifugação do animal. Logo, os graus 1 e 2 são de coloração bem vermelha e indicam que os traços de anemia são praticamente inexistentes. No grau 3, já é indicada a vermifugação. Nos graus 4 e 5, a vermifugação é totalmente presente, uma vez que a mucosa apresenta palidez intensa (CHAGAS *et al.*, 2007).

Portanto, o método FAMACHA auxilia na redução do número de tratamentos aplicados, visto que se trata de um método de tratamento seletivo, ou seja, recebem tratamento anti-helmíntico apenas os animais que apresentam anemia, o que permite retardar o aparecimento de resistência parasitária (SILVA *et al.*, 2019). Entretanto, pode apresentar falhas, atribuídas a fatores como luminosidade, estresse do animal, além de erro humano causado pela falta de treinamento técnico para comparação adequada do cartão à cor da mucosa do animal (DEMOLINER; ALVES, 2017).

Desse modo, o método FAMACHA por ser um processo manual está sujeito a falhas. Logo, estudos recentes fazem uso de técnicas de processamento de imagens e inteligência computacional para reconhecimento de padrões em imagens. Neste contexto, metodologias são desenvolvidas para classificar a coloração da mucosa ocular do animal, a fim de realizar o reconhecimento da anemia por meio de um sistema computacional (ALMEIDA, 2021).

Entretanto, apesar do êxito das técnicas de aprendizado de máquina na classificação de imagens, existem fatores que afetam o desempenho do classificador, como a escassez de informações, tendo em vista que os dados de treinamento são formados por poucas imagens ou não estão disponíveis publicamente para uso. Desse modo, para expandir de forma significativa a base de dados e mitigar a falta de dados rotulados, propõe-se a utilização de métodos de DA (TAYLOR; NITSCHKE, 2017).

Portanto, este trabalho propõe analisar a eficiência de uma *GAN*. como um método de DA, possibilitado o aumento do número de instâncias da base dados, que serão utilizadas no desenvolvimento de um sistema computacional para a detecção da anemia de ovinos. Em suma, o objetivo é gerar imagens sintéticas de qualidades para expandir o conjunto de dados provenientes da coleta de imagens da mucosa ocular de ovinos e verificar como a adição dos novos dados implica na eficiência do classificador. O algoritmo *CNN*, responsável pelo reconhecimento da anemia nos animais, associa a ideia do método FAMACHA com a precisão dos exames laboratoriais.

1.1 Justificativa

O método FAMACHA, amplamente utilizado para detectar anemia em ovinos, por ser um processo manual, está sujeito a falhas. Dessa forma, estudos recentes apresentam alternativas para o método FAMACHA. Sistemas computacionais capturam imagens da mucosa ocular de ovinos e, por meio destas realizam a análise da coloração da conjuntiva do animal.

Nesse contexto, é possível determinar o grau de anemia, de acordo com as cores presentes no cartão FAMACHA (DEMOLINER; ALVES, 2017). Ademais, as *CNN*s, estão sendo amplamente utilizadas em aplicações de classificação, detecção e reconhecimento em imagens, pois aplicam filtros detectores de padrões e preservam a noção espacial das imagens (VARGAS *et al.*, 2016).

Entretanto, as redes neurais artificiais profundas exigem uma grande quantidade de dados de treinamento, que na maioria das vezes não estão disponíveis publicamente, tornando a tarefa complexa e custosa. Dessa forma, realizar o treinamento de uma *CNN* com um pequeno conjunto de dados a torna propensa a *overfitting*, inibindo a capacidade da rede de generalizar para dados invariantes não vistos. Portanto, uma solução em potencial é fazer uso de métodos de DA para expandir a quantidade de dados rotulados em um conjunto de treinamento, e consequentemente, melhorar os resultados do classificador (TAYLOR; NITSCHKE, 2017).

Logo, diversas abordagens são propostas para expandir artificialmente os dados rotulados. A priori, o aumento de dados pode ser realizado por meio de transformações em imagens, que garantem a geração de novas amostras por meio de transformações como rotações, giros, efeitos de brilho, entre outros, no conjunto original. Outrossim, outra abordagem seria o uso de *GAN*, uma rede neural profunda capaz de gerar novas instâncias de dados semelhantes ao conjunto de dados original. Isto posto, estudos demonstram a eficiência da aplicação de *GAN* para melhorar o desempenho de classificadores em ambientes com poucos dados, apresentando um o modelo generativo capaz de aprender a generalizar um dado de um domínio de origem e gerar novos dados (BISSOTO *et al.*, 2021; WAHEED *et al.*, 2020; FRID-ADAR *et al.*, 2018).

Diante das informações apresentadas, é plausível analisar a eficiência da utilização de *GAN* como método DA, a fim de expandir a base de dados das imagens da mucosa ocular de ovinos. Logo, pode-se perceber a relevância em utilizar as instâncias geradas pela *GAN* para realizar o treinamento de uma *CNN*, responsável por detectar o grau de anemia em ruminantes, visto que, redes neurais profundas podem apresentar resultados ineficientes ao serem treinadas com uma carência de dados.

1.2 Trabalhos Relacionados

O trabalho de Demoliner e Alves (2017) propõe informatizar o método FAMACHA, por meio de um aplicativo. A metodologia desenvolvida baseia-se no cartão de cores, utilizado para determinar a anemia em ovinos, causada pelo parasita *Haemonchus contortus*. Entretanto, o método FAMACHA está sujeita a erros humanos na classificação, por trata-se de um processo

manual. Logo, o estudo apresenta app móvel, como uma alternativa para tornar o processo mais preciso e rápido. Dessa forma, para o desenvolvimento do aplicativo foram utilizadas as bibliotecas *OpenCV* e *WEKA*. A coleta das imagens se deu através do aplicativo, totalizando em 414 imagens da conjuntiva do animal, divididas em 66% em imagens de treinamento e 34% para a fase de teste. Ademais, o método estatístico, *Naive Bayes*, permitiu o processo de inferência da classe anêmica, baseando-se nas cores do cartão FAMACHA. Como resultado, foi alcançado uma taxa de acerto de 50,6% para classificar os cinco graus do FAMACHA. A taxa de acerto aumentou para 66,3% ao configurar a classificação em dois grupos: Não tratar (graus 1 e 2) e tratar (Graus 3 a 5).

Os autores Magdalena *et al.* (2022) desenvolveram um estudo para detectar a anemia conjuntiva baseada em imagem usando um dos métodos de aprendizado profundo, a *CNN*. O objetivo principal consiste em obter características mais específicas na distinção de condições normais e anêmicas com base na imagem da conjuntiva palpebral. Para o desenvolvimento do método foram utilizadas 2.000 imagens da conjuntiva palpebral que continham anemia e condições normais, divididas em 1.440 imagens para treinamento, 160 imagens para validação e 400 imagens para teste de modelo. O modelo de *CNN* proposto pelos autores consiste em cinco camadas ocultas, a camada totalmente conectada e a função de ativação sigmóide, para classificar condições normais e anêmicas. Portanto, o trabalho obteve a melhor acurácia de 94%, com o valor médio de precisão de 0.935, um *recall* de 0.94 e *F1-score* igual a 0.935. Os resultados do estudo indicam que o sistema é capaz de classificar condições normais e anêmicas com erros mínimos.

Os autores Waheed *et al.* (2020) apresentaram um método para gerar imagens sintéticas de radiografia de tórax (CXR) envolvendo um modelo baseado em Rede Adversarial Geradora de Classificador Auxiliar (ACGAN) chamado CovidGAN. O trabalho baseia-se em demonstrar que as imagens sintéticas produzidas a partir do CovidGAN podem ser utilizadas para melhorar o desempenho da *CNN* na detecção de COVID-19. Dessa forma, redes de aprendizado profundo, precisam de uma quantidade substancial de dados de treinamento, entretanto, por se tratar de um surto recente, é difícil reunir um número significativo de imagens radiográficas. Logo, o presente trabalho apresentou uma precisão de 85% na classificação usando apenas o conjunto de dados original e ao adicionar imagens sintéticas produzidas pelo CovidGAN, a precisão aumentou para 95%.

O trabalho de Frid-Adar *et al.* (2018) propõe um método de aumento de dados

que gera imagens médicas sintéticas usando *GAN*s. O estudo apresenta um conjunto de dados limitado de imagens de tomografia computadorizada (TC) de 182 lesões hepáticas (53 cistos, 64 metástases e 65 hemangiomas). A priori, os autores realizaram o treinamento com aumento de dados clássico para ampliar o conjunto de dados e, em seguida, para realizar o aumento do conjunto de dados de treinamento com amostras sintéticas aplicou-se *GAN*. Logo, o desempenho da classificação com o aumento de dados clássico foi de 78,6% de sensibilidade e 88,4% de especificidade. Ademais, ao adicionar os dados sintéticos gerados pela *GAN*, os resultados aumentaram significativamente para 85,7% de sensibilidade e 92,4% de especificidade.

1.3 Objetivos

1.3.1 Objetivo geral

Este trabalho possui como objetivo analisar como a adição dos dados sintéticos, gerados por uma *GAN*, implica na eficiência de um classificador, responsável por detectar o grau de anemia nos animais, por meio de imagens da mucosa ocular de ovinos.

1.3.2 Objetivos Específicos

- Analisar trabalhos que utilizem o conceito de geração de imagens com *GAN*;
- Utilizar a arquitetura *StyleGAN* para gerar imagens artificiais da mucosa ocular de ovinos;
- Calcular a pontuação *FID* para avaliar a qualidade das imagens geradas;
- Analisar diferentes arquiteturas *CNN* com a função de detectar a anemia em animais através das imagens da região da mucosa do olho de ovinos;
- Verificar como a proposta de aumento de dados impacta no desempenho do classificador final.

1.4 Organização do Documento

A seção 2 apresenta os principais conceitos teóricos abordados no desenvolvimento deste projeto. Em seguida, a seção 3 descreve as etapas metodológicas realizadas na elaboração deste trabalho. A seção 4 apresenta os resultados obtidos para os experimentos realizados. Por fim, a seção 5 apresenta as considerações finais deste projeto e determina os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção será apresentado um embasamento teórico sobre os principais conceitos abordados no desenvolvimento deste projeto.

2.1 Método FAMACHA

O método FAMACHA, desenvolvido na região da África do Sul (BATH *et al.*, 1996), após um longo período de pesquisa, estabeleceu uma correlação entre a coloração da mucosa ocular de pequenos ruminantes e cinco intervalos de anemia indicados pelo exame de sangue (Jan A. Van Wyk; Gareth F. Bath, 2002). O exame laboratorial, hematócrito (Ht), indica a porcentagem de eritrócitos, também conhecido como glóbulos vermelhos, hemácias ou células vermelhas, no volume total do sangue. É um parâmetro muito utilizado para indicar a saúde do animal e possibilita a identificação e o diagnóstico de determinadas situações, como o estado anêmico do animal, relacionado a diminuição do hematócrito (LACVET, 2022).

Dessa forma, o método FAMACHA é um importante recurso para identificar clinicamente diferentes graus de anemia em animais, causados pelo parasita *Haemonchus contortus*, localizado no abomaso de ruminantes. A Hemoncose é a doença parasitária, disseminada pela ingestão do pasto contaminado com a larva em sua fase infectante e pode causar perda sanguínea, disfunções na absorção dos nutrientes, quadros intensos de anemia e, em casos mais graves, levar o animal a óbito (VIEIRA, 2007).

Dentre as vantagens do método FAMACHA, a redução do número de tratamentos aplicados é de ampla relevância. Isso se deve ao fato de se tratar de um método de tratamento seletivo, ou seja, recebem tratamento anti-helmíntico apenas os animais que apresentam anemia, o que permite retardar o aparecimento de resistência parasitária e minimizar a presença de resíduos nos produtos de origem animal e no ambiente (SILVA *et al.*, 2019; MALAN *et al.*, 2001).

O método consiste na utilização do cartão FAMACHA, ilustrado com os cinco graus de coloração para a mucosa do olho de ovinos, direcionando a vermifugação do animal. A técnica exige expor a conjuntiva, pressionando a pálpebra superior com um dedo polegar e abaixar levemente a pálpebra inferior, para realizar a comparação da coloração conjuntiva do animal com as cores do cartão, como ilustrado na Figura 1 (EMBRAPA, 2017).

Desse modo, os graus 1 e 2, que são de coloração bem vermelha, indicam que os

Figura 1 – Utilização do cartão FAMACHA para avaliação do grau de anemia em caprinos e ovinos.

Fonte: (EMBRAPA, 2017).

traços de anemia são praticamente inexistentes. No grau 3, já é indicada a vermifugação. Nos graus 4 e 5, a vermifugação é imprescindível, uma vez que a mucosa apresenta palidez intensa. Portanto, em (CHAGAS *et al.*, 2007) foi fundamentado a relação do grau FAMACHA com a coloração da mucosa ocular e o valor do hematócrito, orientando a necessidade ou não da realização do tratamento, como definido na Tabela 1.

Tabela 1 – Relação do grau Famacha com a coloração da conjuntiva ocular e o hematócrito, orientando ou não o tratamento.

Grau FAMACHA	Coloração	Hematócrito (%)	Atitude clínica
1	vermelho robusto	> 27	Não tratar
2	vermelho rosado	23 a 27	Não tratar
3	rosa	18 a 22	tratar
4	rosa pálido	13 a 17	tratar
5	branco	< 13	tratar

Fonte: Adaptado de (CHAGAS *et al.*, 2007).

2.2 *Machine Learning*

O *Machine Learning* (*ML*) ou Aprendizado de máquina (AM) é uma área da Inteligência Artificial (IA) que implica no desenvolvimento de técnicas computacionais sobre o aprendizado e em sistemas capazes de adquirir conhecimento de forma automática (MONARD; BARANAUSKAS, 2003).

Samuel (1959) definiu o *ML* como um campo de estudo que fornece capacidade de

aprendizado para computadores sem serem explicitamente programados. Em suma, o conceito refere-se à capacidade de um sistema de adquirir, integrar e estender o conhecimento por meio de observações em larga escala (ALZUBI *et al.*, 2018; WOOLF, 2009).

O estudo de técnicas de *ML* é um campo multidisciplinar com uma ampla gama de domínios de pesquisa que tornou-se amplamente difundido. Atualmente, o *ML* pode ser aplicado em diversas áreas como mineração de texto, detecção de spam, recomendação de vídeo, classificação de imagens, detecção de objetos, recomendação de produtos e diagnóstico médico (ALZUBI *et al.*, 2018; ALZUBAIDI *et al.*, 2021).

No *ML* a inferência indutiva é uma das principais formas para obter conhecimento e predizer eventos futuros. O aprendizado indutivo é realizado por meio de inferência lógica sobre exemplos fornecidos em um processo externo ao sistema de aprendizado. Dessa forma, o aprendizado indutivo pode ser dividido em supervisionado e não supervisionado (MONARD; BARANAUSKAS, 2003).

2.2.1 Aprendizado Supervisionado

O aprendizado supervisionado é responsável por uma gama de atividades de pesquisa e aplicações em *ML*. A principal característica desse tipo de aprendizado é a disponibilidade de treinamento anotados (CUNNINGHAM *et al.*, 2008). Isto posto, é disponibilizado ao algoritmo de aprendizado um conjunto de exemplos ou módulos de treinamento associados aos rótulos de classe. Em suma, os exemplos de entrada são definidos por um vetor de características (atributos) e pelo rótulo da classe associada e, com base no conjunto de treinamento, o algoritmo é treinado para responder com precisão o valor da saída de novos dados (MONARD; BARANAUSKAS, 2003).

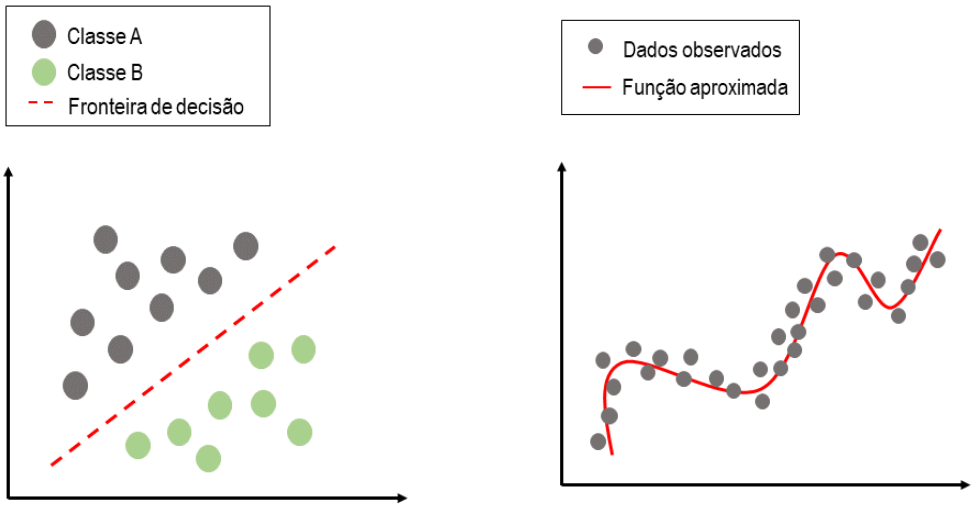
O objetivo do aprendizado supervisionado consiste em definir um algoritmo capaz de determinar corretamente os rótulos de classe de novos exemplos ainda não rotulados. Dessa forma, as atividades de aprendizado supervisionado podem ser definidas como tarefas de classificação ou tarefas de regressão, com base nos rótulos de classe. Portanto, problemas de classificação possuem rótulos de saída discretos, enquanto problemas de regressão consistem em rótulos de saída contínuos (ALZUBI *et al.*, 2018).

Desse modo, a classificação está relacionada com problemas em que a saída pode ser rotulada com base nas classes conhecidas, podendo ser um problema de classificação binária ou multiclasse, de acordo com o número de classes. Portanto, se o rótulo de classes assume um

domínio de valores discretos tem-se um problema de classificação (ALZUBI *et al.*, 2018). A Figura 2(a) ilustra um problema de classificação binária, cujo objetivo principal é determinar uma fronteira de decisão para separar os exemplos de acordo com as classes. Assim sendo, por se tratar de um contexto simples com exemplos linearmente separáveis, a fronteira de decisão pode ser uma reta. Todavia, em um contexto com exemplos não linearmente separáveis seria necessária uma combinação de retas. Se os exemplos apresentassem mais de dois atributos de entrada, em vez de retas, seriam utilizados hiperplanos de separação (FACELI *et al.*, 2011).

Já os algoritmos de regressão são usados para determinar uma saída a partir de um conjunto de valores contínuos (ALZUBI *et al.*, 2018). A Figura 2(b) apresenta um cenário simples de regressão, onde o modelo determina uma aproximação para os pontos com valores contínuos (FACELI *et al.*, 2011).

Figura 2 – Gráfico ilustrativo das tarefas de Aprendizado Supervisionado.



(a) Classificação (b) Regressão.

Fonte: Adaptado de (FACELI *et al.*, 2011).

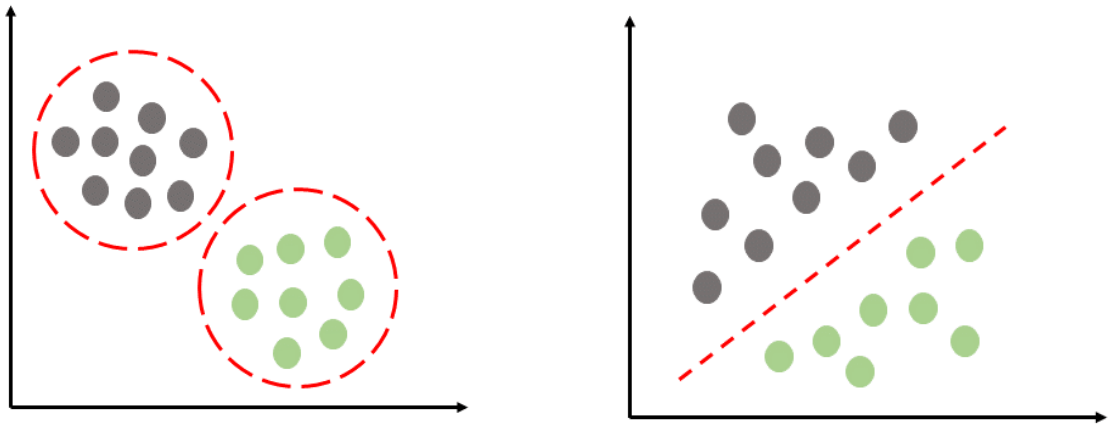
2.2.2 Aprendizado Não Supervisionado

O aprendizado não supervisionado concentra-se em treinar um modelo para definir padrões em um conjunto de dados não rotulados. Desse modo, o algoritmo de aprendizagem realiza um processo de agrupamento, que consiste particionar os dados em grupos, chamados de *clusters*, com base em características similares determinadas por critérios pré-estabelecidos (ALZUBI *et al.*, 2018; MONARD; BARANAUSKAS, 2003).

Portanto, a Figura 3 apresenta exemplos de aplicações de aprendizado não super-

visionado e aprendizado supervisionado. A Figura 3(a) ilustra um problema de aprendizado não supervisionado, em que dados foram agrupados em dois grupos (*clusters*) com base na similaridade entre os objetos. Já a Figura 3(b) representa um problema de classificação do aprendizado supervisionado, onde os objetos são separados em duas classes distintas.

Figura 3 – Gráfico ilustrativo de tarefas de Aprendizado Não Supervisionado e Aprendizado Supervisionado.



(a) Agrupamento. (b) Classificação.

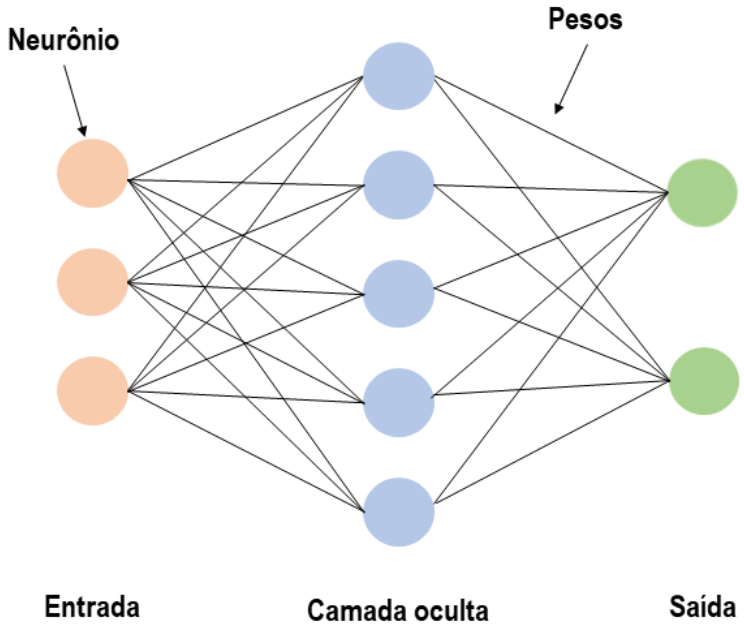
Fonte: Adaptado de (FACELI *et al.*, 2011).

2.2.3 Redes Neurais Artificiais

O estudo de Redes Neurais Artificiais (RNA), chamadas de "Redes Neurais", foi motivado, a priori, pelo reconhecimento de que o cérebro humano computa de forma diferente comparado ao computador digital convencional, devido a capacidade de organizar seus constituintes estruturais, conhecidos como neurônios. Haykin (2009) afirma que uma rede neural é uma máquina projetada para modelar a forma que o cérebro realiza uma determinada tarefa de interesse e é implementada por meio de componentes eletrônicos ou é simulada em *software*.

Desse modo, as redes neurais podem ser aplicadas em diversos campos como análise de séries temporais, reconhecimento de padrões, processamento de sinais, entre outros, devido a sua capacidade de aprender a partir de dados de entrada com ou sem um professor (CSÁJI *et al.*, 2001). Em suma, uma RNA consiste em uma camada de entrada de neurônios (conhecidos como nós ou unidades), seguida por uma ou duas camadas ocultas de neurônios e uma camada final de neurônios de saída. A Figura 4 ilustra uma arquitetura de uma rede neural, com as linhas realizando a conexão dos neurônios (WANG, 2003).

Figura 4 – Representação da arquitetura de uma rede neural com uma camada de entrada, uma camada oculta e uma camada final de neurônios de saída.



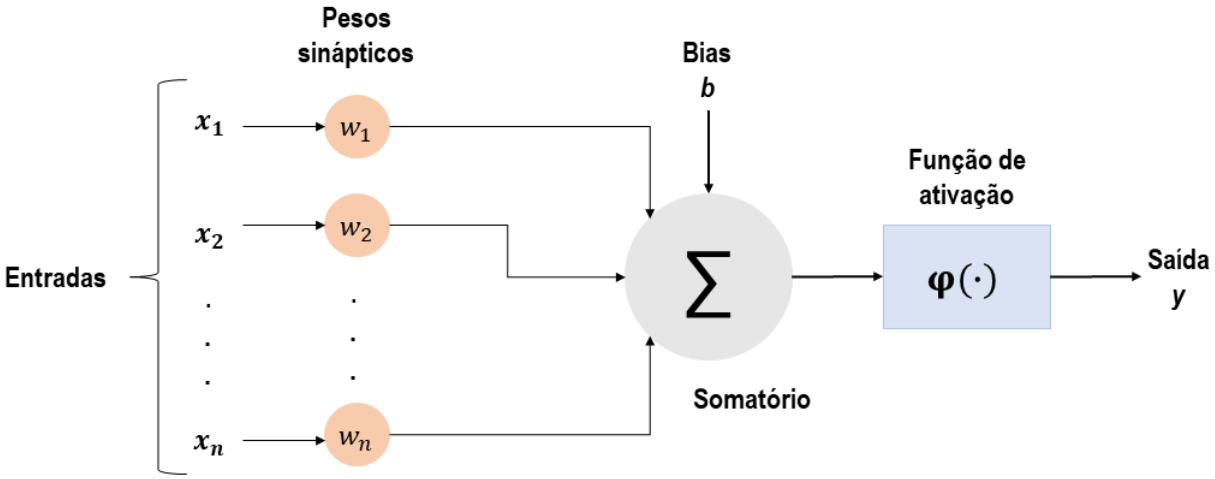
Fonte: Adaptado de (WANG, 2003).

2.2.3.1 Neurônio artificial

Um neurônio artificial é a unidade de informação fundamental para o funcionamento de uma rede neural e são unidades primitivas comparado aos neurônios biológicos. Isto posto, um neurônio artificial possui diversas entradas e apenas uma saída (WANG, 2003).

Dessa forma, existem três elementos essenciais que compõe o fundamento de um modelo neural: um conjunto de sinapses, um somador e uma função de ativação. A priori, o conjunto de sinapses (ou elos de conexão), são caracterizados por um peso, na qual existe um produto do sinal x_j, na entrada da sinapse j conectada a um neurônio, pelo peso sináptico w_j. Em seguida, o somador é responsável por somar os sinais de entrada. Por fim, a função de ativação realiza a limitação da amplitude da saída de um neurônio, normalizada como um intervalo unitário fechado $[0,1]$ ou $[-1,1]$. Ademais, o modelo de um neurônio também inclui um viés externo, que realizam o efeito de aumentar ou diminuir a entrada da função de ativação. A Figura 5 mostra o modelo de um neurônio, que forma a base para projetar as RNA. Portanto, em termos matemáticos, é possível descrever um neurônio k escrevendo a equação (HAYKIN, 2009; WANG, 2003):

Figura 5 – Modelo não linear de um Neurônio Artificial.



Fonte: Adaptado de (HAYKIN, 2009).

$$y = \varphi(\sum_{j=1}^{n} w_j x_j + b) \tag{2.1}$$

Onde, $x_1, ..., x_n$ são os sinais de entrada e $w1, ..., w_n$ são os pesos sinápticos do neurônio k. Ademais, b é o bias, φ é a função de ativação e y é o sinal de saída do neurônio.

2.2.3.2 Perceptron

O modelo Perceptron, desenvolvido por Rosenblatt (1958), consiste na arquitetura mais simples de uma RNA, sendo um modelo que possibilita uma compreensão sobre o funcionamento de uma rede neural em termos matemáticos. Isto posto, o Perceptron é um classificador linear (binário), usado no aprendizado supervisionado para classificar os dados de entrada fornecidos (ACADEMY, 2022).

Dessa forma, o Perceptron segue o modelo *feed-forward*, ou seja, as entradas denotadas por $x_1, x_2, ..., x_n$ são enviadas para o neurônio, processadas e resultam em uma saída y. Os pesos sinápticos do Perceptron são denotados por $w_1, w_2, ..., w_m$, a função de ativação por φ e o viés (*bias*) aplicado externamente é denotado por b (HAYKIN, 2009).

Ademais, o principal objetivo do modelo é classificar corretamente o conjunto de estímulos de entrada $x_1, x_2, ..., x_n$ em duas classes distintas. Portanto, o processo de treinamento de um modelo Perceptron consiste em apresentar ao modelo os valores de entrada e as possíveis

saídas e fazer com que o modelo aprenda os valores ideais de pesos e *bias*. Por conseguinte, com o modelo treinado, é possível prever a saída para novos dados de entrada. Em termos matemáticos, o processamento interno realizado pelo modelo pode ser descrito como (ACADEMY, 2022; HAYKIN, 2009):

$$v = \sum_{i=1}^{m} w_i x_i + b \tag{2.2}$$

Onde, $x_1, ..., x_n$ são os sinais de entrada e $w1, ..., w_n$ são os pesos sinápticos do perceptron. O viés (*bias*) aplicado externamente é denotado por b.

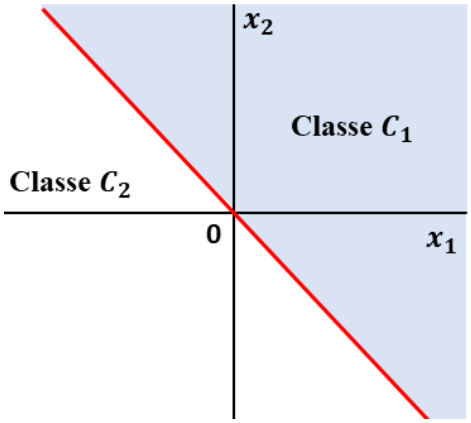
Na forma mais simples do Perceptron, é possível desenvolver um entendimento sobre o comportamento de um classificador de padrões por meio de um mapa das regiões de decisão. Dessa forma, existem duas regiões de decisão separadas por um hiperplano definido como (HAYKIN, 2009):

$$\sum_{i=1}^{m} w_i x_i + b = 0 \tag{2.3}$$

Logo, como ilustrado na Figura 6, o limite de decisão assume a forma de uma linha reta para as entradas x_1 e x_2. Portanto, um ponto (x_1, x_2) que está acima da linha de limite é atribuído à classe C_1, enquanto um ponto definido abaixo da linha de limite é atribuído à classe C_2. Dessa forma, as regiões, separadas por uma única linha, são chamadas de regiões linearmente separáveis. Entretanto, em problemas reais geralmente encontram-se funções não linearmente separáveis, o que reduz a utilidade do Perceptron, visto que este não consegue gerar um hiperplano (HAYKIN, 2009).

Por fim, funções não linearmente separáveis não podem ser alcançados por um único Perceptron. No entanto, esse problema poderia ser superado usando mais de um Perceptron organizado em redes neurais *feedforward*. Dessa forma, é evidente a importância do Perceptron para iniciar o estudo e entendimento de redes neurais, mas existe a necessidade de definir arquiteturas de RNA mais avançadas, uma vez que existem diversos problemas complexos no mundo real (HAYKIN, 2009).

Figura 6 – Ilustração do hiperplano com limite de decisão para um sistema com duas classes.



Fonte: Adaptado de (HAYKIN, 2009).

2.2.3.3 Multilayer Perceptron

O *Multilayer Perceptron* (*MLP*) consiste em um sistema de neurônios interconectados por pesos e sinais de saída, definidos como uma somatório das entradas do neurônio ajustados por uma função de ativação. Dessa forma, a arquitetura *MLP* é formada por diversas camadas de neurônios: uma camada de entrada, que passa o vetor de entrada para a rede, uma ou mais camadas ocultas, composta por neurônios interconectados, e uma camada de saída com as saídas da rede neural ao ambiente externo (GARDNER; DORLING, 1998).

O processo de treinamento de um *MLP* consiste em um sinal de entrada, definido como um estímulo aplicado na entrada da rede, e em pesos que são ajustados até que ocorra o mapeamento desejado. A rede adquire conhecimento por meio do aprendizado supervisionado. Dessa forma, durante o treinamento é definido um sinal de erro, originado em um neurônio de saída e que se propaga para trás, que indica a diferença entre a saída prevista pela rede e a saída real. Isto posto, o sinal de erro é um parâmetro usado pela rede para determinar em que grau os pesos devem ser ajustados para minimizar o erro geral. Portanto, existem diversos algoritmos para treinar um *MLP*, entre eles, o algoritmo *backpropagation* é amplamente utilizado (GARDNER; DORLING, 1998; HAYKIN, 2009).

2.2.3.3.1 *Backpropagation*

O algoritmo *Backpropagation*, definido por Rumelhart *et al.* (1986), surgiu como uma proposta para sanar o obstáculo em utilizar redes multicamadas, que era a ausência de um algoritmo para o treinamento das redes. Dessa forma, o algoritmo é baseado em gradiente

descendente, capaz de tornar mais rápido e eficiente o treinamento de modelos profundos (FACELI *et al.*, 2011; ACADEMY, 2022).

O algoritmo *backpropagation* é definido em duas fases: *forward* e *backward*. Na primeira fase, definida como o passo para frente, cada objeto é apresentado à rede. O propósito da fase *forward* é propagar os dados de entrada através da rede até à camada de saída. Dessa forma, os dados de entrada são recebidos pelos neurônios da camada oculta, onde são ponderados pelo peso correspondente. Ademais, uma função de ativação é aplicada à entrada total de cada neurônio, produzindo uma saída, utilizada como valor de entrada pelos neurônios da camada seguinte. O processo persiste até atingir os neurônios da camada final, para que assim seja produzido um valor de saída que será comparado ao valor de saída desejado. A diferença entre os dois valores compõe o erro cometido pela rede (FACELI *et al.*, 2011).

Outrossim, já na fase de *backward*, conhecida também como o passo para trás, é calculado o gradiente da função de perda na camada final. O valor do erro de cada neurônio da camada de saída é utilizado para ajustar os pesos de entrada, prosseguindo da camada final até a primeira camada oculta. Portanto, o ajuste dos pesos de uma rede neural pelo algoritmo *backpropagation* é dado por (FACELI *et al.*, 2011):

$$w_{jl}(t+1) = w_{jl}(t) + \eta x^j \delta_l \tag{2.4}$$

Onde, w_{jl} representa o peso entre um neurônio l e a saída do $j-esimo$ neurônio da camada anterior. δ_l indica o erro associado ao neurônio correspondente, enquanto x^j indica a entrada recebida por esse neurônio.

2.2.3.3.2 Função de Perda

A função de perda, também conhecida como função de custo, serve para medir o desempenho do modelo de classificação. A função quantifica a compatibilidade entre o resultado produzido pelo modelo, por meio de propagação direta, e os rótulos verdadeiros do conjunto de treinamento. Portanto, as redes neurais são treinadas utilizando um processo de otimização e que requer uma função de perda para calcular o erro do modelo. A função de perda comumente usada para classificação binária é a entropia cruzada (YAMASHITA *et al.*, 2018; BROWNLEE, 2019).

2.2.3.3.3 Gradiente Descendente

A descida de gradiente é amplamente utilizada como algoritmo de otimização. O objetivo da otimização é encontrar os parâmetros, ou seja, os *kernels* e os pesos, que minimizam a função de perda. O gradiente indica a direção na qual a função de custo possui a maior taxa de aumento. Isto posto, os parâmetros são atualizados na direção negativa do gradiente com um tamanho de passo arbitrário determinado com base em uma taxa de aprendizagem. Matematicamente, o gradiente pode ser definido como a derivada parcial da perda em relação a cada parâmetro que pode ser aprendido (YAMASHITA *et al.*, 2018):

$$w := w - \alpha * \frac{\partial L}{\partial w} \tag{2.5}$$

Onde, w representa os parâmetros a serem aprendidos, L determina a função de perda e α indica a taxa de aprendizado.

Portanto, os métodos de otimização são primordiais para obter uma função de custo mínima e melhorar o desempenho das redes neurais. Logo, os algoritmos de otimização *SGD* (QIAN, 1999), *RMSprop* (HINTON *et al.*, 2012) e *Adam* (KINGMA; BA, 2014), são comumente utilizados para minimizar a taxa de erro, buscando evitar o sobreajuste excessivo dos dados (YAMASHITA *et al.*, 2018).

2.2.3.4 Função de ativação

Em redes neurais, as funções de ativação desempenham um papel importante no treinamento e no desempenho da rede, pois ajudam na aprendizagem e oferecem as propriedades não lineares necessárias para que o modelo consiga aprender as representações complexas (RASAMOELINA *et al.*, 2020).

A organização das redes neurais consiste em várias camadas, cada uma composta por nós interconectados associados às funções de ativação. As RNAs são responsáveis por realizarem transformações matemáticas nos dados em processamento. Assim sendo, em cada neurônio ocorre a multiplicação do valor de entrada pelo peso do neurônio correspondente e o resultado é somado ao valor do *bias* em questão, para que assim o resultado seja passado para a próxima camada. Entretanto, essas operações são lineares e possuem uma complexidade limitada, o que minimiza a capacidade da rede de aprender e reconhecer mapeamentos complexos a partir de

dados (SHARMA *et al.*, 2020).

Desse modo, a rede neural sem funções de ativação atua como um Modelo de Regressão Linear com desempenho limitado na maioria dos casos. Portanto, a principal razão para o uso das funções de ativação é conferir a capacidade não-linear ao processamento realizado pelas redes neurais. Desse modo, o uso de funções de ativação possibilita execução de tarefas mais complexas, como a modelagem de tipos de dados diversos, de alta dimensão e não lineares (SHARMA *et al.*, 2020; ACADEMY, 2020).

2.2.3.4.1 ReLU

Proposta por Nair e Hinton (2010), a função ReLU tem sido amplamente utilizada por pesquisadores em redes neurais devido ao desempenho de treinamento superior em relação a outras funções de ativação, como a sigmóide logística e a tangente hiperbólica. A função assume valor zero se o valor de entrada para a função de ativação (x) for negativo, caso contrário, retorna o valor de x. Matematicamente, a função ReLU é definida como (RASAMOELINA *et al.*, 2020; GUSTINELI, 2022):

$$\varphi(x) = max(0,x) \tag{2.6}$$

Dessa forma, por ser uma função linear, a ReLU possui baixo custo computacional, uma vez que transforma as entradas negativas passadas para a função em uma saída zero. Entretanto, os neurônios com pesos negativos não contribuem para o desempenho geral da rede. Ademais, a ReLU proporciona melhor desempenho e generalização quando comparada às funções sigmoid e tanh (GUSTINELI, 2022). O gráfico dessa função pode ser visualizado na Figura 7(a).

2.2.3.4.2 Sigmóide

Também conhecida como função sigmóide logística, é uma das funções mais comumente usadas em redes neurais, devido à simplicidade da derivada. A função consiste em traduzir a faixa de entrada igual a $(-\infty;+\infty)$ para valores do intervalo $[0,1]$. Isto posto, quanto maior o valor de entrado, mais próximo o valor de saída estará de $1,0$, em contrapartida, quanto menor for o valor de entrada mais próximo o valor de saída estará de $0,0$. A função sigmóide é

definida para todos os valores reais de entrada da seguinte forma (RASAMOELINA *et al.*, 2020; GUSTINELI, 2022):

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.7}$$

Ademais, a função sigmóide é uma função exponencial, o que resulta em um alto custo computacional, contudo, o problema pode ser resolvido utilizando o gradiente, que é definido como (GUSTINELI, 2022):

$$\sigma'(x) = \sigma(x)(1-\sigma(x)) \tag{2.8}$$

Por fim, a desvantagem está associada ao fato da função está limitada em um intervalo entre 0 e 1. Ademais, a função sempre produz valores positivos como saída. Logo, uma grande alteração nos valores de entrada gera apenas uma pequena mudança no valor de saída (RASAMOELINA *et al.*, 2020). O gráfico dessa função pode ser visualizado na Figura 7(b).

2.2.3.4.3 Tangente Hiperbólica

A função Tanh possui comportamento semelhante à função sigmóide. A função produz valores no intervalo $[-1,1]$, logo, quanto maior a entrada, mais próximo o valor da saída estará de $1,0$, em contrapartida quanto menor a entrada, mais próximo o valor da saída estará de $-1,0$. Em termos matemático, é definida como (GUSTINELI, 2022):

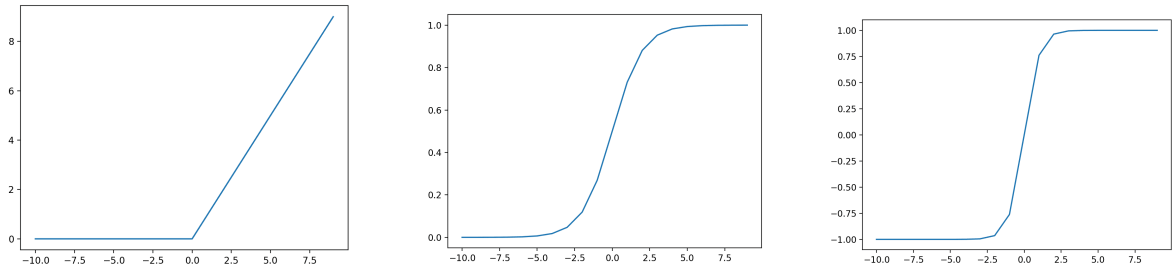
$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1 \tag{2.9}$$

Dessa forma, trata-se de uma função contínua e diferenciável. O gráfico da função Tanh é mostrado na Figura 7(c).

2.2.4 Deep Learning

O *Deep Learning* (*DL*), se estabeleceu como um subcampo do *ML*, remodelando o futuro da IA, devido as melhorias no poder computacional, ao armazenamento rápido de dados e da capacidade de gerar recursos de alto nível otimizados automaticamente. Os fundamentos

Figura 7 – Gráfico de Entradas x Saídas para as Funções de Ativação.



(a) ReLU. (b) Sigmóide. (c) tanh

Fonte: (BROWNLEE, 2021)

teóricos do *DL* baseiam-se em redes neurais, com o uso de diversos neurônios e camadas ocultas (RAVì *et al.*, 2017).

Em *DL*, modelos computacionais compostos por diversas camadas de processamento conseguem aprender representações de dados com vários nível de abstração. Dessa forma, o *DL* faz uso do algoritmo *backpropagation* para descobrir uma estrutura complexa em grandes conjuntos de dados, de tal modo que seja possível a rede alterar os parâmetros internos utilizados para calcular a representação em cada camada, por meio da representação da camada anterior (LECUN *et al.*, 2015).

Dessa forma, o *DL* tornou-se uma crescente área de interesse nos últimos anos, destacando estudos em subáreas como reconhecimento de faces (GUO; ZHANG, 2019), imagens médicas (ANAYA-ISAZA *et al.*, 2021) e bioinformática (MIN *et al.*, 2016). Portanto, entre as variantes metodológicas de aprendizado profundo, as *GAN*s e as *CNN*s apresentam grande visibilidade. Dessa forma, as *CNN*s foram responsáveis pelos avanços no processamento de imagens, vídeo, fala e áudio, enquanto, as *GAN*s indicam melhorias significativas na geração de vídeo e imagens, transformação e síntese de imagem e super-resolução de imagem (LECUN *et al.*, 2015; ABIRAMI *et al.*, 2021).

2.2.4.1 GAN

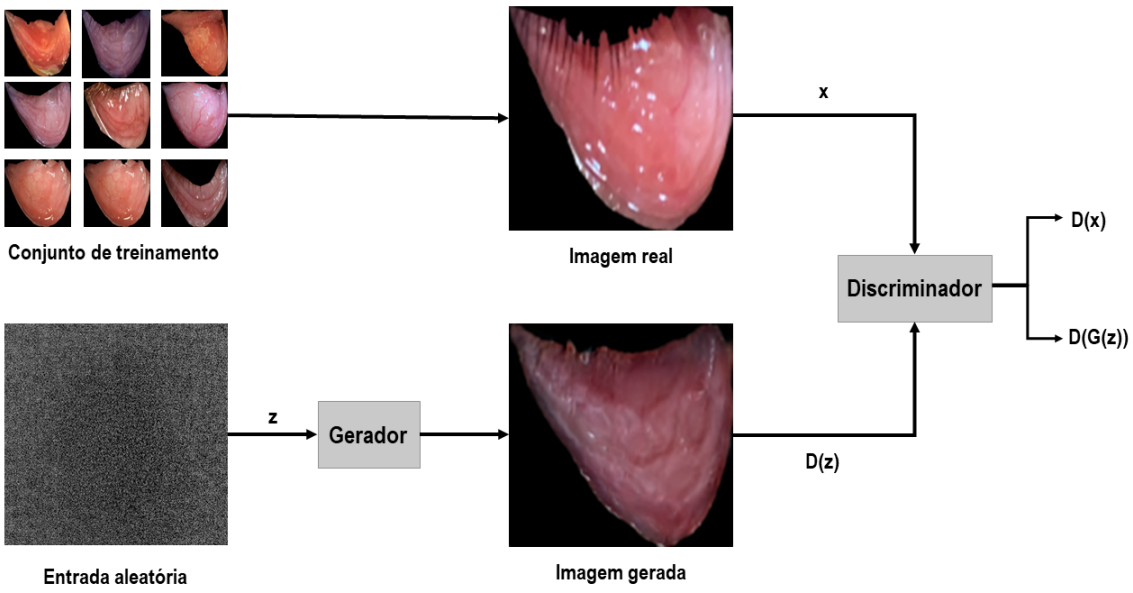
As *GAN*s, introduzidas por Goodfellow *et al.* (2014), é uma das vias de pesquisa mais importantes na área de IA. Consistem em uma nova abordagem para modelagem generativa, com uma excelente capacidade de geração de dados. O principal objetivo dessa rede neural profunda é aprender uma distribuição de pontos de dados reais e gerar novas instâncias semelhantes ao conjunto de dados original. Dessa forma, com base na teoria dos jogos, as *GAN*s são compostas por duas redes, denominadas de rede geradora e rede discriminadora, treinadas por algoritmos

de retropropagação e de *dropout* (PAN *et al.*, 2019).

A proposta da *GAN* é treinar simultaneamente dois modelos: um modelo generativo G e um modelo discriminativo D. O princípio do gerador G é gerar dados sintéticos que se adequem aos dados reais, enquanto o discriminador D, estima a probabilidade de uma amostra pertencer aos dados de treinamento ou aos dados gerados por G.

Desse modo, o gerador G captura uma distribuição de dados desconhecida representada por um vetor de ruído aleatório de entrada z, e realiza o mapeamento para um novo espaço de dados, um vetor multidimensional $G(z)$, para obter uma amostra falsa. Já o discriminador D, um classificador binário, recebe como entrada a amostra real do conjunto de dados e a amostra sintética e retorna como saída a probabilidade de a amostra analisada ser falsa ou real. A estrutura geral do sistema *GAN* é apresentada na Figura 8 (PAN *et al.*, 2019; SWIDERSKI *et al.*, 2021).

Figura 8 – Representação da estrutura do sistema *GAN*.



Fonte: Adaptado de (de Souza *et al.*, 2020).

O procedimento de treinamento para G é maximizar a probabilidade de D cometer um erro, enquanto o processo de treinamento de D é maximizar a probabilidade de atribuir o rótulo correto aos exemplos de treinamento e amostras de G. Portanto, a tarefa de aprendizagem para o sistema *GAN* consiste no problema de otimização *minimax* da função $V(G,D)$ (GOODFELLOW *et al.*, 2014; SWIDERSKI *et al.*, 2021; de Souza *et al.*, 2020):

$$\min_{G} \max_{D} V(D,G) = E_{x\sim p_{data}(x)}[\log D(x|y)] + E_{z\sim p_z(z)}[\log(1 - D(G(z|y)))] \tag{2.10}$$

Onde V representa a função objetivo e as variáveis p_x e p_z representam os dados reais e a distribuição dos dados do gerador, respectivamente. Ademais, $D(x)$ e $G(x)$ representam os valores definidos pelo discriminador D e gerador G. E_x simboliza a variável de ruído de entrada e E_z indica a distribuição de dados. Logo, o processo de otimização busca o valor mínimo em relação a G e o máximo em relação a D.

Por fim, é evidente o sucesso significativo em relação aos desafios de geração de imagens plausíveis, entretanto, as *GAN*s ainda apresentam limitações quanto a problemas do mundo real, como geração de imagens de alta qualidade, diversidade de geração de imagens e treinamento estável. Em vista disso, vários modelos derivados de *GAN*s foram propostos, como a arquitetura *StyleGAN*, responsável por gerar imagens em altíssima resolução (WANG *et al.*, 2019).

2.2.4.1.1 *StyleGAN*

A arquitetura *StyleGAN*, proposta por Karras *et al.* (2018), é uma alternativa para as *GAN*s, baseada em transferência de estilo. A rede geradora foi redefinida para ocorrer o controle do processo de síntese de imagens por meio de modificações específicas de escala nos estilos. Dessa forma, a nova arquitetura do gerador concentra-se em melhorar o estado da imagem, em termos de métricas tradicionais de qualidade de distribuição.

Ademais, em Karras *et al.* (2019), propõe-se a *StyleGan2*, um modelo aprimorado que define mudanças na arquitetura *StyleGAN* e nos métodos de treinamento. As principais modificações consistem em remodelar a normalização do gerador e regularizá-lo para incentivar um bom condicionamento no mapeamento de códigos latentes para imagens. Como resultado, a nova arquitetura consegue melhorar a qualidade da imagem, tanto em termos de métricas de qualidade de distribuição existentes quanto em qualidade de imagem percebida.

Outrossim, um desafio presente em cenários de modelagem generativa é a coleta de um conjunto grande o suficiente de imagens para alimentar uma rede específica, visto que existem restrições relacionadas ao tipo de assunto, qualidade da imagem, privacidade, entre outros. Dessa forma, adquirir, processar e distribuir as imagens necessárias para treinar uma *GAN* de alta qualidade pode se tornar um empreendimento caro. Em conjuntos de dados limitados

pode ocorrer o ajuste excessivo do discriminador aos exemplos de treinamento, fazendo com que o treinamento divirja. Portanto, para findar o empecilho de *overfitting* do discriminador, causado pelos poucos dados no treinamento, Karras *et al.* (2020) define a arquitetura *StyleGAN2-ADA*. Desse modo, trata-se mecanismo de aumento de discriminador adaptativo que estabiliza significativamente o treinamento em regimes de dados limitados. Portanto, a versão conhecida como *StyleGAN2-ADA* foi lançada como forma de resolver a escassez de dados de treinamento, presentes nas versões anteriores.

2.2.4.1.2 *FID*

A avaliação de modelos generativos é uma área ativa de pesquisa. Diversas métricas de avaliação foram propostas e são comumente utilizadas para avaliar o desempenho das *GAN*s. Dentre as métricas de avaliação mais influentes, o *FID* tem sido amplamente adotado devido à sua consistência com a inspeção humana e sensibilidade a pequenas mudanças na distribuição real (BORJI, 2022).

A pontuação *FID*, introduzida por Heusel *et al.* (2018), propõe avaliar o desempenho da *GAN* na geração de imagens, capturando a similaridade das imagens geradas com as reais, como uma melhoria em relação ao *Inception Score (IS)*. Desse modo, trata-se de uma métrica para calcular a distância entre vetores de recursos calculados para as imagens reais e geradas. Em suma, o principal objetivo da métrica é avaliar a qualidade das imagens sintéticas geradas por *GAN*, na qual, pontuações mais baixas sugerem imagens de qualidade mais alta.

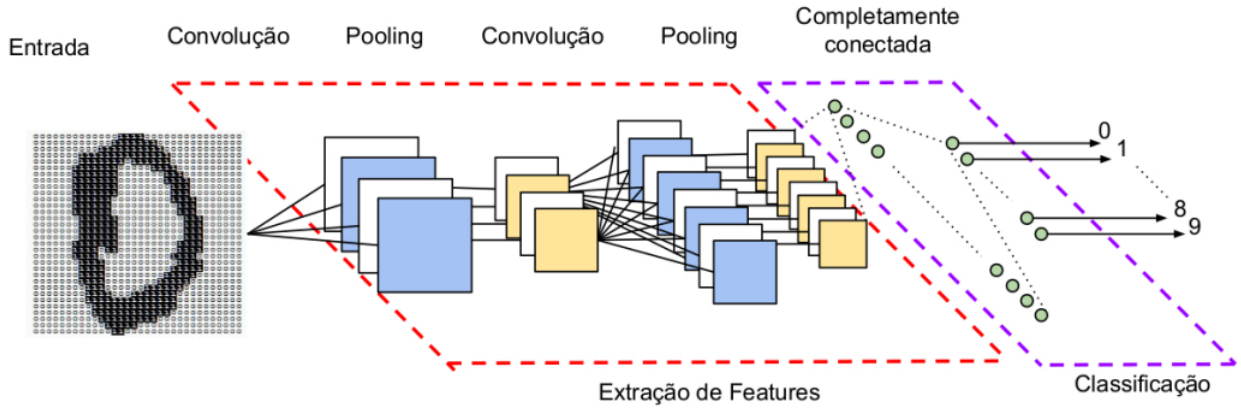
Ademais, a medida *FID* faz uso da rede InceptionV3 pré-treinada, utilizada na classificação de imagens, para calcular estatisticamente a semelhança entre dois grupos de imagens, distribuições original e sintética (de Souza *et al.*, 2020). O resultado desse cálculo determina que pontuações mais baixas se referem a dois grupos mais semelhantes, correspondendo a boas imagens sintéticas visuais. Uma pontuação igual a zero indica uma pontuação perfeita, ou seja, que os dois grupos de imagens são idênticos. Portanto, a pontuação *FID*, comumente utilizada como uma métrica para avaliar a qualidade das imagens geradas por *GAN*, determina o desempenho da rede, correlacionando o valor da pontuação com a qualidade da imagem (YU *et al.*, 2021).

2.2.4.2 CNN

Introduzidas por Lecun *et al.* (1998), trata-se do algoritmo de aprendizado profundo mais popular e comumente empregado nas aplicações. Apresentam vantagem em relação aos seus antecessores por detectar automaticamente os recursos significativos sem qualquer supervisão humana. As *CNNs* podem ser aplicadas em diversos campos, incluindo visão computacional, processamento de fala, reconhecimento facial e aplicações de classificação (ALZUBAIDI *et al.*, 2021).

As *CNNs* foram inspiradas no processo biológico de processamento de dados visuais, e de forma semelhante aos processos tradicionais de visão computacional, são capazes de aplicar filtros em dados, mantendo a relação de vizinhança entre os *pixels* da imagem ao longo do processamento da rede. A estrutura básica de uma *CNN* consiste em múltiplas camadas com diferentes funções. A priori, aplica-se sobre o dado de entrada camadas de convolução, responsáveis por aplicar filtros (*kernel*) em um pedaço específico da imagem. Em seguida, são aplicadas as camadas de *pooling*, capazes de reduzir a dimensionalidade dos dados na rede. Por fim, para realizar uma tarefa de classificação, acrescenta-se ao menos uma camada totalmente conectada. A Figura 9 ilustra uma arquitetura *CNN* e suas diferentes camadas (VARGAS *et al.*, 2016).

Figura 9 – Exemplo de arquitetura *CNN* para classificação de imagens.



Fonte: (VARGAS *et al.*, 2016).

2.2.4.2.1 Camadas *CNN*

A arquitetura *CNN* é composta pelo empilhamento de vários blocos de construção: camadas de convolução e de agrupamento, que realizam a extração de recursos, e camadas totalmente conectadas (FC), responsáveis por mapear os recursos extraídos na saída final, como classificação. As camadas da arquitetura *CNN* são descritas em detalhes abaixo.

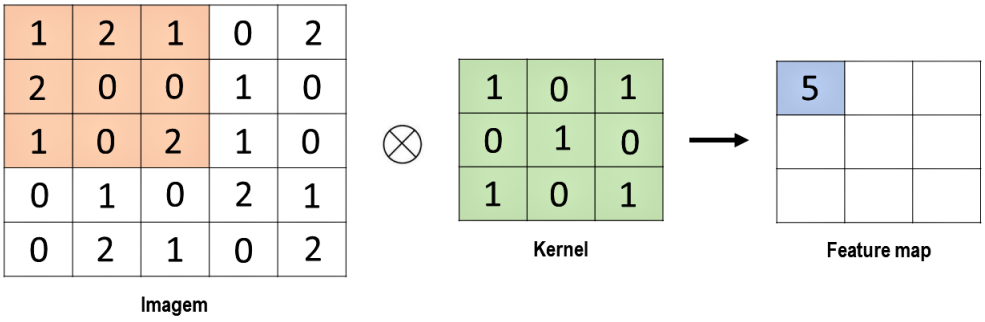
2.2.4.2.2 Convolução

A camada convolucional é um fundamental componente da arquitetura *CNN*. É responsável por realizar a extração de características, que consiste em uma coleção de operações lineares e não lineares, ou seja, operação de convolução e função de ativação. Na camada de convolução a imagem de entrada é representada por uma matriz de *pixels*, chamada de *input feature maps* (tensor). Nesse contexto, ocorre o processo de convolução, onde uma pequena matriz de filtro, chamada de *kernel* é aplicada na entrada. Desse modo, as matrizes de filtro e a *input feature maps* são sobrepostas por meio de passos, denominados *strides*. É realizado um produto entre cada elemento das matrizes sobrepostas que são somados para obter o valor de saída. O resultado é adicionado em uma posição correspondente ao mapa de característica, chamado de *feature map* (ALZUBAIDI *et al.*, 2021; YAMASHITA *et al.*, 2018).

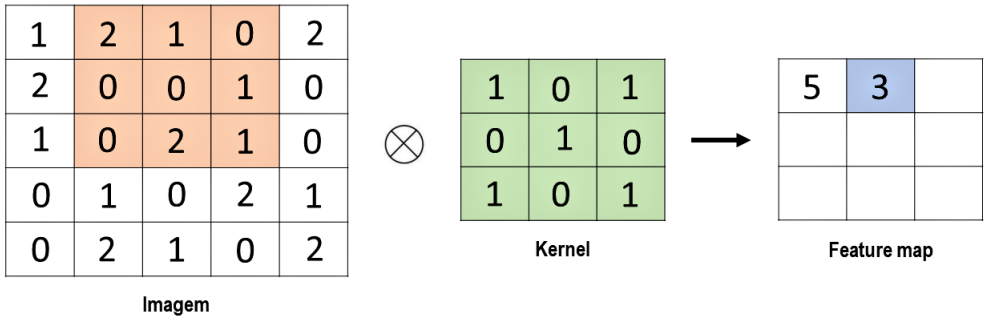
Para representar a operação convolucional, é possível considerar um exemplo com uma matriz de pixels 5 x 5, representando a imagem de entrada, um *kernel* de tamanho 3 x 3 e passo igual a 1. A priori, o *kernel* é aplicado ao *input feature maps*. Dessa forma, é determinado o produto entre a imagem de entrada e o *kernel*, onde seus valores correspondentes são multiplicados e somados para obter o valor de saída na posição correspondente do *output feature maps*. Logo, a Figura 10 ilustra os cálculos executados em cada etapa. Portanto, na primeira iteração da operação, representada pela Figura 10(a), é realizado o cálculo $(1 \cdot 1) + (2 \cdot 0) + (1 \cdot 1) + (2 \cdot 0) + (0 \cdot 1) + (0 \cdot 0) + (1 \cdot 1) + (0 \cdot 0) + (2 \cdot 1) = 5$, resultando no valor de saída na posição correspondente do *feature map*. Já na Figura 10(b), é apresentada a segunda iteração da operação, o cálculo consiste em $(2 \cdot 1) + (1 \cdot 0) + (0 \cdot 1) + (0 \cdot 0) + (0 \cdot 1) + (1 \cdot 0) + (0 \cdot 1) + (2 \cdot 0) + (1 \cdot 1) = 3$. O processo se repete até toda a matriz de pixel ser percorrida, como exibido na Figura 10(c) (YAMASHITA *et al.*, 2018; DUMOULIN; VISIN, 2016).

Portanto, no que se refere à camada de convolução, no processo de treinamento do modelo *CNN*, é necessário identificar os *kernels* que possuem o melhor desempenho para uma

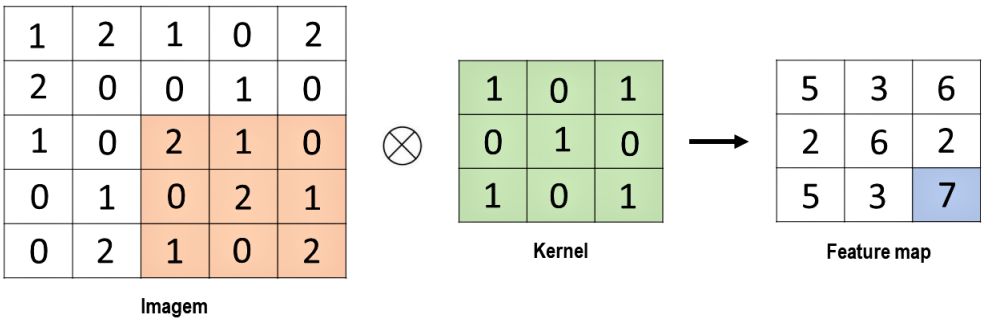
Figura 10 – Cálculos executados em cada etapa da camada convolucional.



(a) Primeira Iteração



(b) Segunda Iteração.



(c) Última Iteração.

Fonte: Adaptado de (ALZUBAIDI *et al.*, 2021)

determinada tarefa conforme o conjunto de dados de treinamento. Ademais, os *kernels* são os únicos parâmetros aprendidos automaticamente durante o processo de treinamento, enquanto que, parâmetros como tamanho do *kernel* e *stride* precisam ser definidos antes do processo de treinamento (YAMASHITA *et al.*, 2018).

Outrossim, para adicionar a não linearidade a rede, são aplicadas funções de ativação após as saídas de uma operação linear, como a convolução. Dessa forma, as funções de ativação aplicam uma transformação nos dados recebidos. Atualmente, a função de ativação mais bem-sucedida e amplamente utilizada é a ReLU, principalmente no contexto de imagens, visto que, as *CNN*s com ReLU realiza o treinamento várias vezes mais rápido do que seus equivalentes

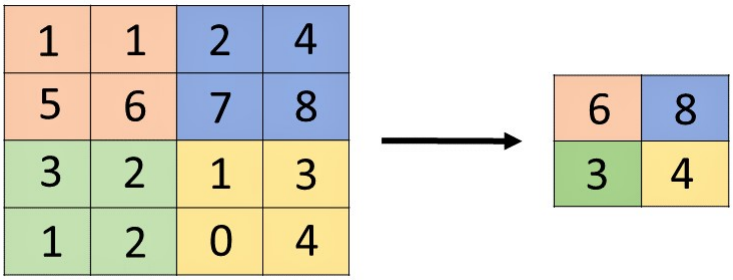
com unidades tanh (KRIZHEVSKY *et al.*, 2012; RAMACHANDRAN *et al.*, 2017).

2.2.4.2.3 *Pooling*

A camada de *pooling*, aplicada após as camadas de convolução e de ativação, é responsável por reduzir a dimensionalidade do *feature maps* para introduzir a invariância espacial e diminuir o número de parâmetros subsequentes que podem ser aprendidos, gerando uma agilidade no treinamento. Existem diferentes operações de *pooling*, sendo, o método de agrupamento máximo o mais conhecido e frequentemente utilizado (ALZUBAIDI *et al.*, 2021).

A operação *Max pooling*, extrai o maior elemento dos valores de uma vizinhança, considerando a grade de *pooling* e, em seguida, é realizado o deslizamento conforme o parâmetro *stride*. Dessa forma, considerando o exemplo, ilustrado na Figura 11, uma matriz de *pixels* 4 x 4, uma camada de *pooling* com filtros de tamanho 2x2 aplicados com um passo de 2, terá como saída para cada iteração o valor máximo dos elementos.

Figura 11 – Representação da operação *Max pooling*.



Fonte: Adaptado de (CS231N, 2022).

2.2.4.2.4 Camada Totalmente Conectada

Para realizar uma tarefa de classificação é acrescentada uma Camada Totalmente Conectada (FC) na arquitetura *CNN*. Dentro dessa camada, cada neurônio está conectado a todos os neurônios da camada anterior. A entrada da FC, possui a forma de um vetor, proveniente da última camada de *pooling* ou da camada convolucional. A entrada da FC é criada a partir dos mapas de características. Ademais, a saída da FC representa as probabilidades de cada classe nas tarefas de classificação da *CNN* (ALZUBAIDI *et al.*, 2021).

A camada final totalmente conectada normalmente possui o número de nós de saída igual ao número de classes e é seguida por uma função não linear. Dessa forma, a função de

ativação aplicada à última camada é selecionada de acordo com a tarefa desejada. A função de ativação *sigmóide* é aplicada para tarefas de classificação binária, em contrapartida, a função *softmax* é aplicada para tarefas de classificação multiclasse (YAMASHITA *et al.*, 2018).

2.2.4.2.5 Treinamento da Rede

O treinamento da rede consiste em apresentar os dados ao algoritmo para que o aprendizado ocorra. Dessa forma, durante o processo são definidos os *kernels* e os pesos, em camadas de convolução e em camadas totalmente conectadas, respectivamente. Isto posto, busca-se minimizar as saídas previstas e os rótulos originais do conjunto de dados de treinamento. O algoritmo *backpropagation* é amplamente utilizado para realizar o treinamento de redes neurais, na qual a função de perda e o algoritmo de otimização de gradiente descendente desempenham papéis fundamentais. Portanto, os *kernels* e os pesos são atualizados de acordo com o valor da perda por meio do algoritmo de otimização *backpropagation* (YAMASHITA *et al.*, 2018).

2.2.4.3 Arquiteturas CNN

Com o aprimoramento das *CNN*s foram propostas diversas arquiteturas adequadas ao número de camadas, quantidade de classes e imagens, proporcionando um aprendizado mais robusto (SILVA *et al.*, 2020). As arquiteturas EfficienteNetB0V2 (TAN; LE, 2021), InceptionV3 (SZEGEDY *et al.*, 2015) e ResNet101V2 (HE *et al.*, 2015) são detalhadas nas subseções seguintes.

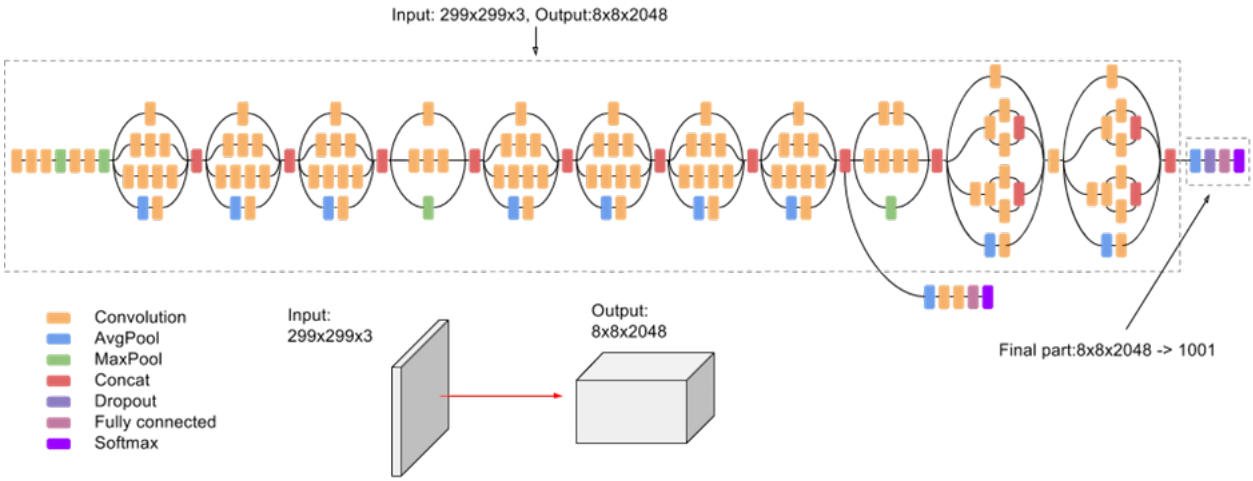
2.2.4.3.1 EfficientNetV2

A arquitetura EfficientNetV2 é uma nova abordagem de redes convolucionais, proposta por Tan e Le (2021). Apresentam uma melhor eficiência de parâmetros e uma velocidade de treinamento mais rápida em relação aos modelos anteriores. A arquitetura do modelo usa o *MBConv* (bloco de construção do EfficientNets) e *fused-MBConv* nas camadas iniciais. Ademais, a EfficientNetV2 utiliza um *kernel* menor de tamanho 3x3, mas adiciona mais camadas para compensar o campo receptivo reduzido resultante do tamanho de *kernel*. Neste trabalho será utilizada a arquitetura EfficientNetV2B0, pertencente à família de modelos da EfficientNetV2, carregada com pesos pré-treinados do ImageNet.

2.2.4.3.2 InceptionV3

A arquitetura InceptionV3, proposta por Szegedy *et al.* (2015), é um modelo de reconhecimento de imagem. Trata-se de uma evolução das arquiteturas GoogLeNet e InceptionV1. O principal objetivo dessa arquitetura é atuar como um extrator de características em vários níveis. O modelo InceptionV3 atua em problemas de classificação de imagens e foi treinado com base no conjunto de dados ImageNet. A rede apresenta um bom desempenho e um baixo custo computacional em comparação com as redes VGG. O modelo é composto por elementos básicos simétricos e assimétricos, incluindo camadas de convolução, *pooling* médio (*Average Pooling*), *pool* máximo (*Max pooling*), concatenações, *dropouts* e cama camadas totalmente conectadas. Ademais, a normalização em lote é usada em todo o modelo e aplicada às entradas de ativação. A classificação consiste no uso da função de ativação *Softmax*. A Figura 12 apresenta um diagrama da arquitetura InceptionV3 (CLOUD, 2022; SILVA *et al.*, 2020).

Figura 12 – Diagrama do modelo InceptionV3.



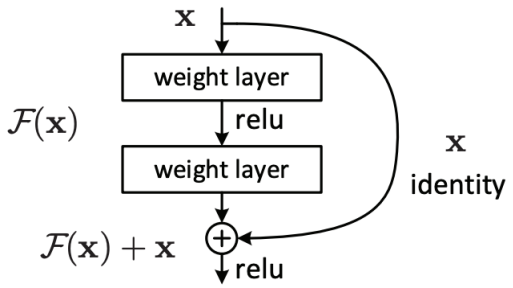
Fonte: (CLOUD, 2022).

2.2.4.3.3 ResNet

Introduzida por He *et al.* (2015), a arquitetura *Deep Residual Network* ou ResNet, facilita o treinamento de redes neurais extremamente profundas. A arquitetura básica da ResNet consiste em blocos residuais, que facilita o treinamento da rede e evita problemas causados pelo aumento da profundidade da rede, como saturação da acurácia e geração de um maior erro de treinamento. A Figura 13 apresenta uma representação de um bloco residual, na qual

uma entrada passa por uma série de operações e o resultado é adicionado à entrada original. Neste estudo, será utilizado o modelo ResNet101V2, uma abordagem contendo 101 camadas de profundidade, carregada com pesos pré-treinados do ImageNet.

Figura 13 – Representação de um bloco residual de uma rede ResNet.



Fonte: (CLOUD, 2022).

2.2.5 Métricas de avaliação

No desenvolvimento de algoritmos de *ML* é imprescindível a utilização de métricas de avaliação adequadas para avaliar o desempenho. A qualidade do modelo é definida pelo valor das métricas de avaliação, dessa forma, é crucial a escolha de métricas que avaliem corretamente se o modelo está de fato atendendo aos requisitos necessários.

2.2.5.1 Métricas para classificação

Problemas de classificação referem-se à categorização dos dados em diferentes classes. Em geral, a classificação realizada pelos modelos pode ser binária ou multiclasses. Dessa forma, a classificação binária tem como objetivo decidir em qual classe uma nova observação pertence dentre dois rótulos possíveis, enquanto a multiclasses apresentam mais dois rótulos (NASER; ALAVI, 2020).

Nessa perspectiva, um modelo de classificação binária possui duas classes, denominadas de positiva (+) e negativa (-), que indicam a ocorrência ou não de determinado evento. Isto posto, a avaliação de um modelo de classificação consiste em comparar as classes preditas pelo modelo e as classes verdadeiras. Assim, o desempenho dos classificadores é listado pela matriz de confusão, que apresenta estatísticas sobre classificações reais e previstas.

2.2.5.1.1 Matriz de confusão

A matriz de confusão resume o desempenho de um modelo de classificação em relação aos dados de testes. É uma matriz bidimensional indexada pela classe verdadeira e pela classe predita em cada dimensão (TING, 2010).

Considerando um problema de classificação binária, usualmente rotuladas com uma classe positiva (+) e uma classe negativa (-). Neste caso, os dois erros possíveis são denominados de Falso Positivo (FP) e Falso Negativo (FN). Já os dois acertos possíveis são definidos como Verdadeiro Positivo (VP) e Verdadeiro Negativo (VN). Portanto, a matriz de confusão ilustrada na Tabela 2, está estruturada para predizer a ocorrência ou não de um evento simples, onde (MONARD; BARANAUSKAS, 2003):

- VP: corresponde ao número de exemplos da classe positiva classificados corretamente;
- VN: corresponde ao número de exemplos da classe negativa classificados corretamente;
- FP: corresponde ao número de exemplos da classe negativa, mas que foram classificados incorretamente como pertencendo à classe positiva;
- FN: corresponde ao número de exemplos da classe positiva, mas que foram classificados incorretamente como pertencendo à classe negativa;

Por fim, o número total de exemplos do conjunto é definido como (FACELI *et al.*, 2011):

$$n = VP + FN + FP + VN \tag{2.11}$$

Tabela 2 – Matriz de confusão.

	Classe predita -	Classe predita +
Classe verdadeira -	VN	FP
Classe verdadeira +	FN	VP

Fonte: Adaptado de (FACELI *et al.*, 2011).

Outrossim, diversas medidas de desempenho podem ser derivadas da matriz de confusão, entre elas (NASER; ALAVI, 2020; TING, 2010; FACELI *et al.*, 2011):

- Acurácia (acc): Avalia a proporção do número de previsões corretas de um modelo para o número total de amostras.

$$acc = \frac{VP + VN}{n} \tag{2.12}$$

- Taxa de erro total (err): Refere-se a medida do grau de erro de previsão de um modelo em relação ao número total de amostras.

$$err = \frac{FP + FN}{n} \tag{2.13}$$

- Precisão (prec): Mede a proporção de observações positivas classificadas corretamente entre todos aqueles preditos como positivos.

$$prec = \frac{VP}{VP + FP} \tag{2.14}$$

- Sensibilidade (sens) ou revocação (*recall*): Mede a proporção de observações positivas reais que são identificados corretamente como positivos. Corresponde à taxa de acerto na classe positiva, também chamada de taxa de verdadeiros positivos.

$$sens = recall = \frac{VP}{VP + FN} \tag{2.15}$$

- Especificidade (esp): Mede as proporções de observações negativas que são verdadeiros positivos. Corresponde à taxa de acerto na classe negativa.

$$esp = \frac{VN}{VN + FP} \tag{2.16}$$

- F_1 Score: Define a média harmônica da precisão e da revocação, dando o mesmo grau importância às duas medidas.

$$F_1 = \frac{2 \times prec \times recall}{prec + recall} \tag{2.17}$$

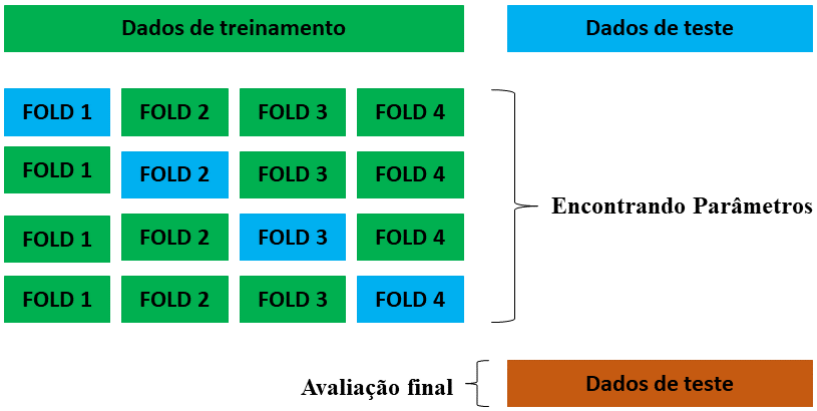
2.2.6 Validação Cruzada

A validação cruzada é um método de reamostragem de dados usado para avaliar a capacidade de generalização de modelos e evitar *overfitting*. Problemas de *overfitting* ocorrem

quando o modelo se adapta perfeitamente ao conjunto de dados disponível, entretanto, é incapaz de generalizar de forma adequada para novos dados. Em suma, a validação cruzada consiste em avaliar a capacidade de generalização de um modelo, estimar o verdadeiro erro de previsão e ajustar os parâmetros da rede. Na validação cruzada os dados são divididos em dois segmentos, definidos como conjunto de treinamento e de validação (BERRAR, 2018; REFAEILZADEH *et al.*, 2016).

O método básico comumente usado é a validação cruzada *k-fold*. Nesse tipo de validação o conjunto de dados é particionado aleatoriamente em k subconjuntos, denominadas *folds*. O modelo é treinado a partir de um conjunto de dados de treinamento composto por $k-1$ *folds*. Posteriormente, para medir o desempenho do modelo, utiliza-se o *fold* restante, denominado de conjunto de validação. O procedimento ocorre até que cada um dos k subconjuntos tenham sido aplicados como conjunto de validação. Portanto, a média do desempenho para os k subconjunto de validação determina o desempenho da validação cruzada (BERRAR, 2018). A Figura 14 ilustra o processo da validação cruzada *k-fold* para $k = 4$.

Figura 14 – Validação Cruzada *k-fold* para $k = 4$.



Fonte: Adaptado de (SCIKIT-LEARN, 2007 - 2022).

A priori, na primeira iteração o *fold* 1 é utilizado como conjunto de validação, enquanto que os subconjuntos restantes são utilizados como conjunto de treinamento. Em seguida, na segunda iteração usa-se o *fold* 2 como conjunto de validação e novamente os subconjuntos restantes são utilizados para realizar o treinamento do modelo. O processo termina quando todos os subconjuntos forem usados como conjunto de validação (BERRAR, 2018; SCIKIT-LEARN, 2007 - 2022).

2.3 Aumento de dados

O aumento de dados é uma técnica amplamente utilizada para expandir o tamanho de um conjunto de dados de treinamento, principalmente, em cenários em que existe uma escassez de informações, relacionada aos tamanhos das bases de dados disponíveis. Desse modo, o aumento de dados objetiva gerar instâncias de dados adicionais para o conjunto de treinamento, mitigar a falta de dados rotulados e melhorar o desempenho do modelo quando validado por amostras separadas (CHLAP *et al.*, 2021).

Redes neurais profundas exigem um grande volume de dados de treinamento para alcançar o aprendizado efetivo do modelo, contudo, a coleta desses dados geralmente é custosa. Dessa forma, os métodos de DA é uma solução em potencial para aumentar os dados de treinamento, especialmente, quando os dados são limitados ou trabalhosos para coletar e também para evitar o *overfitting* do modelo. Portanto, o método DA realiza artificialmente o aumento do conjunto de treinamento e pode ser representado como o mapeamento (TAYLOR; NITSCHKE, 2017):

$$\phi : S \rightarrow T \tag{2.18}$$

Onde, S é o conjunto de treinamento da base de dados original e T é o conjunto expandido de S. Dessa forma, o conjunto de treinamento aumentado artificialmente $S^{'}$, composto pelos dados originais e pelas respectivas transformações ϕ, pode ser representado como:

$$S^{'} = S \cup T \tag{2.19}$$

Logo, existem diferentes abordagens para o aumento de dados. A priori, os métodos aplicados para realizar o aumento de dados consistem em transformações em imagens, seja transformações geométricas ou fotométricas, responsáveis por alterar, respectivamente, a geometria e os canais de cores das imagens (TAYLOR; NITSCHKE, 2017; MASI *et al.*, 2016). Outrossim, outra abordagem para o aumento de dados é a aplicação de modelos generativos profundos, como as *GAN*s, cuja finalidade é gerar novas instâncias de dados após um processo de treinamento com as amostras extraídas de alguma distribuição (TAYLOR; NITSCHKE, 2017).

2.3.1 Transformações em Imagens

As primeiras aplicações que apresentaram eficácia como método de DA foram as transformações simples em imagens, que compreende as transformações geométricas e as fotométricas. Nesse contexto, são utilizadas transformações afins para manipular os dados de treinamento. Por fim, um conjunto de dados expandido é gerado a partir das transformações aplicadas nas imagens de entrada (PEREZ; WANG, 2017; SHORTEN; KHOSHGOFTAAR, 2019).

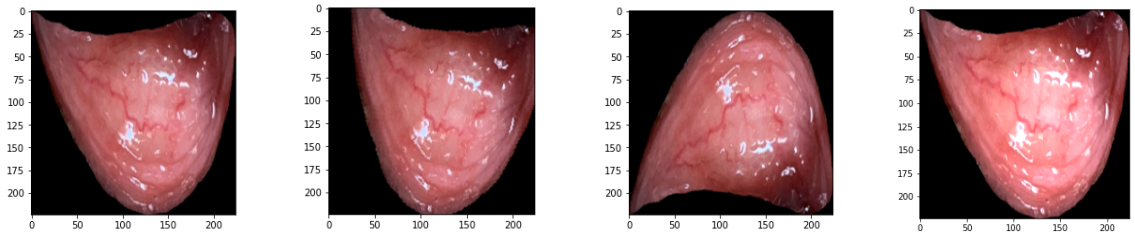
A priori, as transformações geométricas são técnicas que alteram a geometria da imagem mapeando os valores de pixels individuais para novos destinos. Desse modo, rotação, giro, zoom e deslocamento são alguns exemplos de transformações geométricas. Em contrapartida, as transformações fotométricas, definidas pela aplicação de efeitos como ruído, brilho e saturação, alteram os canais RGB, deslocando cada valor de *pixel* (r, g, b) para novos valores de *pixel* de acordo com heurísticas pré-definidas (TAYLOR; NITSCHKE, 2017).

Em suma, as transformações geométricas e fotométricas comumente utilizadas (SHORTEN; KHOSHGOFTAAR, 2019; BARELLI, 2018):

- Rotação: Consiste em girar a imagem em um ângulo determinado;
- Zoom: Ocorre uma ampliação da imagem;
- Flipping: Aplica uma inversão na imagem, podendo ser uma inversão na horizontal ou na vertical;
- Translação: Consiste em deslocar uma imagem na direção vertical ou horizontal;
- Brilho: Modificações referentes às condições de iluminação da imagem;
- Ruído: Aplicação de uma matriz de valores aleatórios extraídos de uma distribuição gaussiana.

Na Figura 15 é exibido uma exemplificação de transformações geométricas e fotométricas aplicadas à uma imagem da mucosa ocular de ovinos. Dessa forma, a Figura 15(a) apresenta a imagem original sem transformação, seguida da Figura 15(b) que consiste na imagem original submetida à uma rotação de 30°. Ademais, na Figura 15(c) foi aplicada uma inversão vertical na imagem, finalizando com a Figura 15(d) com alteração no brilho.

Figura 15 – Exemplos de transformações geométricas e fotométricas aplicadas em uma imagem.



(a) Original. (b) Rotação. (c) Flip Vertical. (d) Brilho.

Fonte: Elaborado pela autora.

2.3.2 Aumento de dados com GANs

Os modelos generativos são interessantes no contexto de aumento de dados. Dessa forma, esses modelos consistem em gerar instâncias artificiais a partir de um conjunto de dados, com finalidade de obter a máxima similaridade ao conjunto original. As *GAN*s destacam-se no processo generativo, liderando em fatores de velocidade de computação e qualidade dos resultados, comparada a outros técnicas de modelagem generativa que existem, como os *autoencoders variacionais*. Portanto, as *GAN*s apresentam um notável desempenho em tarefas de aumento de dados, devido a capacidade da rede de gerar novas amostras para o conjunto de treinamento, resultando em modelos de classificação com melhor desempenho (SHORTEN; KHOSHGOFTAAR, 2019).

Por fim, neste trabalho será analisado o comportamento das *GAN*s como um método de aumento de dados em comparação às transformações em imagens tradicionais. Portanto, o objetivo é expandir a base de dados de imagens da mucosa ocular de ovinos e verificar o impacto no desempenho dos classificadores responsáveis por detectar o grau de anemia no animal.

3 METODOLOGIA

Nesta seção está concentrada a metodologia utilizadas para a elaboração deste trabalho. Serão abordados detalhes sobre a base de dados utilizada, os *softwares* e bibliotecas e o fluxo dos experimentos realizados no desenvolvimento deste projeto.

3.1 Abordagens dos Experimentos

Este projeto consiste em duas etapas principais: a utilização de *GAN* para geração de imagens artificiais da mucosa ocular de ovinos e a classificação do estado anêmico de animais por meio de modelos *CNN*. A priori, a seção 3.2 apresenta a base de dados usada para a construção do trabalho. Em seguida, na seção 3.3 são expostas as linguagens, bibliotecas e plataformas utilizadas na construção deste projeto. Ademais, na seção 3.4 é explicado o procedimento de geração de imagens com *GAN*. Posteriormente, na seção 3.5 são expostas as transformações aplicadas aos dados. Por fim, na seção 3.6 contém os detalhes da etapa de classificação. Nessa etapa, para realizar o treinamento dos modelos são utilizadas as imagens presentes no conjunto de dados original, a base de dados aumentada por métodos de DA tradicionais e, por fim, o conjunto de dados expandido com imagens as imagens geradas pela *GAN*.

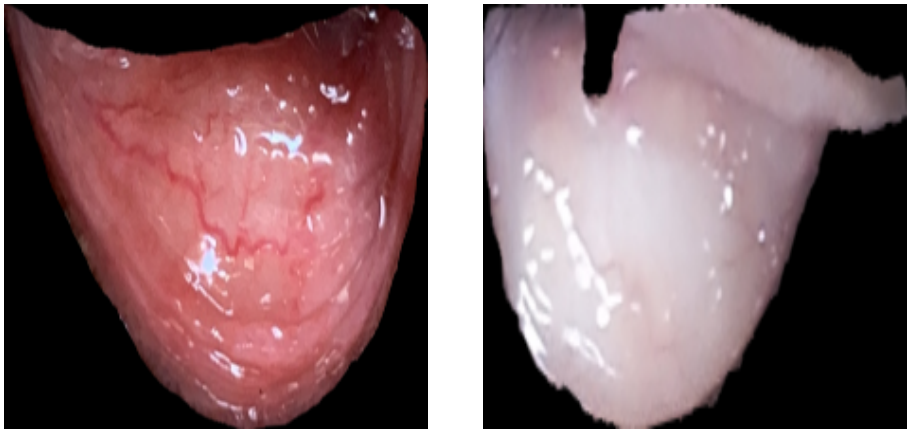
3.2 Base de dados

A base de dados utilizada para o desenvolvimento deste trabalho é composta por 106 imagens da mucosa ocular de ovinos das raças Santa Inês, Morada Nova e Somalis. As imagens estão divididas em duas classes, sendo 53 amostras para a classe “Tratar”, referentes aos animais anêmicos (graus 3, 4 e 5 do método FAMACHA) e 53 instâncias para a classe “Não Tratar”, ou seja, os animais não anêmicos (graus 1 e 2 do método FAMACHA).

A Figura 16 apresenta duas imagens que compõem a base de dados. A Figura 16(a) corresponde à classe “Não Tratar”, ausência de anemia, enquanto a Figura 16(b) pertence à classe "Tratar", presença de anemia.

As imagens foram coletadas na EMBRAPA, localizada na região de Sobral – Ceará em uma colaboração entre os discentes da Universidade Federal do Ceará (UFC) e os colabores da EMBRAPA. Após o processo de coleta, as imagens foram segmentadas, para eliminar regiões desnecessárias, como os dedos do tratador e pelo do animal, preservando apenas a região de interesse. Dessa forma, as imagens segmentadas possuem tamanho igual a 224 x 224 pixels,

Figura 16 – Amostras da base de dados de imagens da mucosa ocular de ovinos.



(a) Ausência de anemia. (b) Presença de anemia.

Fonte: Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA) e (ALMEIDA, 2021).

RGB de 3 canais e formato PNG (ALMEIDA, 2021).

3.3 Linguagens e bibliotecas

Para o desenvolvimento das aplicações foi utilizada a linguagem de programação *Python*[1] (Versão 3.7.1), por se tratar de uma linguagem *Open-Source* com uma ampla variedade de bibliotecas comumente utilizadas para desenvolver aplicações de *ML*.

Ademais, a plataforma *TensorFlow*[2], desenvolvida pelo *Google* em 2015, é a principal biblioteca de código aberto para desenvolver e criar modelos de *ML*. A *API Keras*[3], projetada com base no *TensorFlow2*, é comumente utilizada para experimentações com redes neurais profundas, por ser fácil de usar, modular e extensível. A biblioteca de código aberto *scikit-learn*[4], é amplamente empregada em aplicações de aprendizado de máquina. Dessa forma, as bibliotecas foram utilizadas no desenvolvimento dos modelos de *ML* presentes neste projeto.

Outrossim, a biblioteca multiplataforma *OpenCV*[5], desenvolvida pela Intel, foi projetada para o desenvolvimento de aplicações na área de Visão Computacional, Processamento de Imagens, Estrutura de Dados e Álgebra Linear. A biblioteca possui um conjunto de funções que simplificam a tarefa de manipular e processar imagens digitais. Neste projeto, a biblioteca está sendo usada para realizar o pré-processamento das imagens.

Desse modo, para potencializar a linguagem *Python* no desenvolvimento de sistemas

[1] Disponível em: <https://www.python.org/>. Acesso em: 28 maio 2022.
[2] Disponível em: <https://www.tensorflow.org/>. Acesso em: 28 maio 2022.
[3] Disponível em: <https://keras.io/>. Acesso em: 28 maio 2022.
[4] Disponível em: <https://scikit-learn.org/stable/>. Acesso em: 28 maio 2022.
[5] Disponível em: <https://opencv.org/>. Acesso em: 28 maio 2022.

de visão computacional, algumas bibliotecas podem ser instaladas. A biblioteca *NumPy*[6] facilita trabalhar com arranjos, vetores e matrizes. Isto posto, a biblioteca *NumPy* fornece um conjunto de operações que ajudam na execução de cálculos numéricos usados em tarefas de *ML* e processamento de imagem. A biblioteca *Matplotlib*[7] possui ferramentas que permite gerar diferentes gráficos, contribuindo para a apresentação dos resultados obtidos.

Por fim, o ambiente de desenvolvimento utilizado foi o *Google Colaboratory*[8], adequado para tarefas de aprendizado de máquina e análise de dados. A plataforma consiste em um serviço de notebooks hospedados do *Jupyter*[9]. Possibilita escrever e executar código *Python* pelo navegador e oferece acesso a recursos de computação como GPUs. Os notebooks são armazenados no *Google Drive*[10], um serviço de armazenamento e sincronização de arquivos.

3.4 Geração de dados com GAN

Para gerar as novas instâncias de dados de imagens da mucosa ocular de ovinos foi utilizada a arquitetura *StyleGAN2-ADA*. Nesse processo foram geradas 400 imagens com tamanho igual a 64 x 64 *pixels*. As imagens estão divididas de acordo com o rótulo de classes, sendo 200 imagens para classe "Tratar", referentes aos animais que apresentam sinais de anemia, e 200 imagens para o rótulo de classe "Não Tratar", ou seja, os animais saudáveis. Durante o processo de geração de imagens foi realizado o cálculo para definir a métrica *FID*, responsável por capturar a similaridade das imagens geradas com as reais. Por fim, foram realizadas 2000 iterações, obtendo um tempo total de treinamento de 18h 02m 37s. As imagens geradas foram armazenadas no serviço de armazenamento *Google Drive*.

3.5 Transformações dos Dados

Essa fase consiste em aplicar transformações nos dados. Em geral, para aumentar a eficiência, as imagens de entrada são redimensionadas para uma resolução espacial relativamente pequena (TALEBI; MILANFAR, 2021). Neste projeto, todas as imagens foram redimensionadas para o tamanho 128 x 128, por meio da função *cv2.resize()*[11] fornecida pela biblioteca *OpenCV*.

[6] Disponível em: <https://numpy.org/>. Acesso em: 28 mai. 2022.
[7] Disponível em: <https://matplotlib.org>. Acesso em: 28 maio 2022.
[8] Disponível em: <https://colab.research.google.com/>. Acesso em: 28 maio 2022.
[9] Disponível em: <https://jupyter.org/>. Acesso em: 28 maio 2022.
[10] Disponível em: <https://www.google.com/drive/>. Acesso em: 28 maio 2022.
[11] Disponível em: <https://bit.ly/3zh9aAA>. Acesso em: 3 jun. 2022.

Posteriormente, é usada a função *cv2.normalize()*[12] para realizar a normalização dos dados. A normalização é comumente realizada com o intuito de aumentar a velocidade com que os parâmetros aprendem seus valores ideais (FERLITSCH, 2021). Dessa forma, a imagem, definida como uma matriz de valores inteiros de 0 a 255, quando normalizada possui seus valores comprimidos para um intervalo $[0, 1]$.

3.6 Classificação

Para o reconhecimento do estado anêmico do animal, realizou-se o processo de classificação por meio de uma *CNN*. O classificador é responsável por definir a probabilidade de uma saída pertencer à mesma classe de uma determinada entrada. Nesse projeto, para realizar a classificação foram utilizadas as arquiteturas EfficientNetV2B0, InceptionV3 e ResNet101V2, carregadas com pesos pré-treinados do ImageNet. A priori, a base de dados foi dividida em 80% para dados de treinamento (84 imagens), 15% para dados de validação (17 imagens) e 5% para dados de teste (5 imagens).

3.6.1 Criação do modelo

No processo de classificação foi admitido o uso de aprendizagem profunda. Nesse caso, aplicou-se as arquiteturas EfficientNetV2B0, InceptionV3 e ResNet101V2. Ademais, foram definidos alguns os parâmetros para construção do modelo:

- Camadas de convolução e de *pooling*: Aplicadas de acordo com as arquiteturas pré-fabricadas;
- *Batch Normalization*: Responsável por acelerar o treinamento, melhorar o desempenho da rede e diminuir problemas de *overfitting*;
- Camada *dropout* de 0.2: Responsável por selecionar aleatoriamente neurônios que serão temporariamente descartados durante o treinamento;
- Camada Totalmente Conectada: Camada de saída da rede. Possui dois nós de saída, representando o número de classes;
- Função de ativação *sigmóide*: Aplicada em tarefas de classificação binária.

[12] Disponível em: <https://bit.ly/3aoFUgW>. Acesso em: 3 jun. 2022.

3.6.2 Compilação e treinamento do modelo

Após a criação do modelo é necessário realizar o processo de compilação e de treinamento do modelo. Para isso foram usados:

- Otimizador Adam: Aplicado com a finalidade de obter a função de custo mínima. Taxa de aprendizado de 0.001;
- Função de perda Entropia cruzada *"CategoricalCrossentropy"*: Utilizada em problemas com duas ou mais classes;
- Métrica: Para visualizar a precisão de treinamento e validação aplicou-se a métrica *"accuracy"*;
- Quantidade de épocas: 50.

3.6.3 Aumento de dados para conjunto de treinamento

Em sistemas de classificação a quantidade de dados de treinamento é importante para o desempenho do modelo. Em geral, o problema de *overfitting* ocorre quando existe um pequeno número de dados de treinamento. Neste trabalho, o *dataset* é composto por 106 imagens, divididas em 80% para o conjunto de treinamento (84 imagens), 15% para o conjunto de validação (17 imagens) e 5% para o conjunto de teste (5 imagens).

Portanto, para solucionar os problemas causados pela escassez de dados rotulados, adotou-se a abordagem de técnicas de aumento de dados para gerar dados de treinamento adicionais, por meio técnicas tradicionais de transformações em imagens e aplicação de *GAN*s.

A priori, como método de aumento de dados foram aplicadas transformações em imagens no conjunto de treinamento. Dessa forma, o objetivo ao aplicar o aumento de dados é melhorar a capacidade de generalização do modelo treinado. Nesse contexto, são utilizadas transformações afins para manipular os dados de treinamento, porém, as transformações não são aplicadas para os conjuntos de validação e teste. Neste trabalho, foram aplicadas as seguintes transformações afins ao conjunto de treinamento:

- Zoom = 0.08;
- Rotação = 1;
- Flip = Vertical e horizontal;
- Translação vertical e horizontal = 0.08.

Ademais, a segunda alternativa utilizada como método de aumento de dados foi o

uso de *GAN*. Isto posto, novamente o processo de aumento de dados ocorreu apenas nos dados de treinamento, preservando os conjuntos de validação e teste. Logo, foram adicionadas 400 imagens ao conjunto de dados de treinamento, sendo 200 imagens pertencentes ao rótulo de classe "Tratar" e 200 imagens referentes ao rótulo "Não tratar". A base de dados consiste nos seguintes valores:

- Conjunto de treinamento: 484 imagens;
- Conjunto de validação: 17 imagens;
- Conjunto de teste: 5 imagens.

Por fim, o processo de treinamento das arquiteturas ocorreu de três formas distintas. A princípio, para realizar o treinamento dos modelos utilizou-se o conjunto de dados originais sem o processo de aumento de dados. Em seguida, o treinamento foi realizado com base no conjunto de treinamento aumentado por meio de técnicas de transformações de imagens. Por fim, foi utilizado o conjunto de dados de treinamento expandido com as imagens geradas pela *GAN*.

3.6.4 Validação Cruzada

Neste procedimento foi utilizada a validação cruzada *k-fold*. Nessa abordagem, o conjunto de dados é particionado aleatoriamente em $k = 4$ *folds*. Dessa forma, em cada iteração um subconjunto dos dados é utilizado para validação, enquanto os demais subconjuntos são usados para realizar o treinamento dos dados. O processo finaliza quando cada um dos subconjuntos tiver sido aplicado como conjunto de validação. Por fim, o desempenho do modelo é determinado pela média do desempenho para os $k = 4$ *folds* de validação.

3.6.5 Análise dos resultados

A matriz de confusão foi construída para resumir o desempenho do modelo em relação aos dados de testes, sendo responsável por comparar as previsões do modelo com as respostas verdadeiras. Portanto, a matriz de confusão definida neste trabalho é uma matriz bidimensional indexada pela classe verdadeira (linhas) e pela classe predita (colunas) em cada dimensão, referenciando os rótulos de classe "Tratar" e "Não Tratar".

Dessa forma, a partir da matriz de confusão, é possível calcular as métricas para avaliar se o algoritmo está ou não conseguindo bons resultados. Logo, para analisar os resultados do modelo para o conjunto de teste foram aplicadas as métricas de avaliação: Precisão, *Recall* e *F1-Score*.

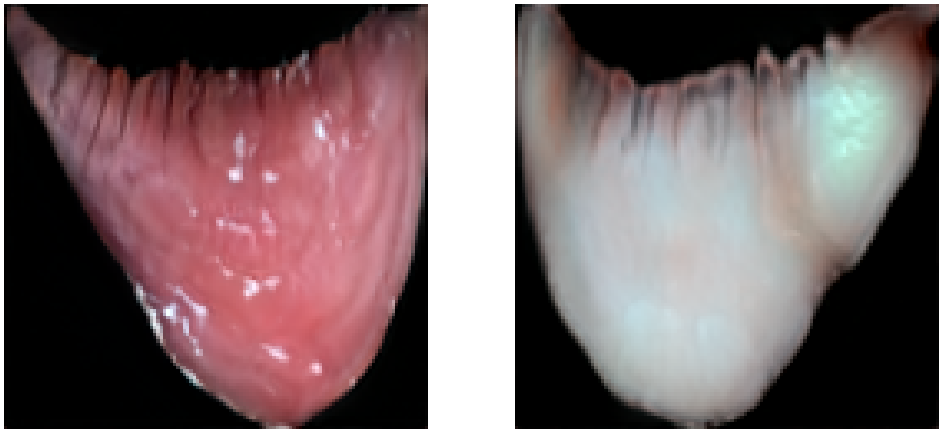
4 RESULTADOS

Nesta seção serão apresentados os resultados obtidos para os experimentos realizados neste projeto.

4.1 Geração de dados com *GAN*

A arquitetura *StyleGAN2-ADA* foi utilizada para gerar imagens sintéticas da mucosa ocular de ovinos. Dessa forma, foram geradas 400 imagens, com tamanho igual a 64 x 64 *pixels*, divididas em 200 imagens representando o rótulo de classe "Tratar", ou seja, animais que apresentam sinais de anemia, e 200 imagens que indicam o rótulo de classe "Não Tratar", referentes aos animais saudáveis. As Figuras 17(a) e 17(b) apresentam as amostras geradas pela arquitetura *StyleGAN2-ADA*, correspondente à classe "Não Tratar"e à classe “Tratar”.

Figura 17 – Exemplos de imagens geradas pela arquitetura *StyleGAN2-ADA*.



(a) Rótulo de classe "Não Tratar" (b) Rótulo de classe "Tratar"

Fonte: Elaborado pela autora.

Por fim, para avaliar o desempenho da *GAN* na geração de imagens, calculou-se a métrica de avaliação *FID*, responsável por capturar a similaridade das imagens geradas com as reais. Dessa forma, o *FID* para as imagens geradas pela *GAN* foi de $74,44$. Por fim, foram realizadas 2000 interações, obtendo um tempo total de treinamento de 18h 02m 37s.

4.2 Classificação

4.2.1 Base de dados original

Para realizar a classificação foram utilizadas as arquiteturas EfficientNetV2B0, InceptionV3 e ResNet101V2. A priori, o treinamento dos modelos foi realizado com a base de dados original, composta por 106 imagens, divididas em treino (84 imagens), teste (5 imagens) e validação (17 imagens). A Tabela 3 apresenta os valores de acurácia de validação e perda do modelo para a validação cruzada *k-fold* com $k = 4$. A arquitetura EfficientNetV2B0 apresentou uma acurácia média de validação de $54,14\%$ e um erro médio de $0,7249$. Já a arquitetura InceptionV3 obteve acurácia média de validação de $53,48\%$ e um erro de $1,7558$. Por fim, a arquitetura ResNet101V2 apresentou os melhores valores para acurácia média de validação e para perda, sendo, respectivamente, $65,38\%$ e $1,2261$.

Tabela 3 – Resultado médio de acurácia de validação e perda para os modelo treinados com a base de dados original.

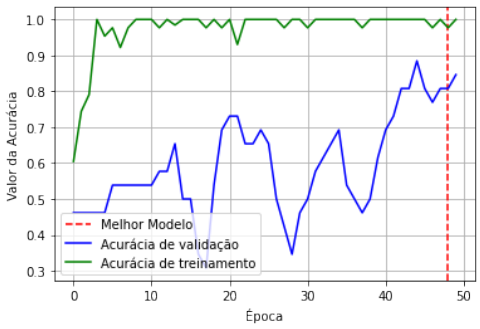
		Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Média**
EfficientNetV2B0	**Acurácia**	59,38%	47,76%	54,15%	55,27%	54,14%±0,0416
	Loss	0,6614	0,7639	0,7161	0,7583	0,7249
InceptionV3	**Acurácia**	65,38%	51,59%	55,03%	41,92%	53,48%±0,0838
	Loss	0,9312	2,5311	1,3012	2,2596	1,7558
ResNet101V2	**Acurácia**	71,30%	71,76%	61,44%	57,03%	**65,38%**±0.0634
	Loss	0,7841	0,8645	1,850	1,4055	1,2261

Fonte: Elaborado pela autora.

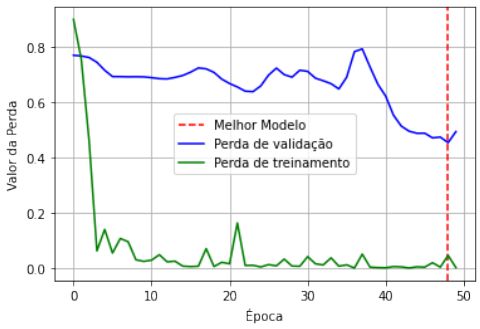
Ademais, a Figura 18 apresenta os gráficos com o comportamento das acurácias e das perdas obtidas pelos modelos durante a etapa de treinamento, utilizando a base de dados original. Dessa forma, as Figuras 18(a), 18(c) e 18(e) exibem os gráficos com a acurácia média para o treinamento realizado pelos modelos EfficientNetV2B0, InceptionV3 e ResNet101V2, respectivamente. Em contrapartida, as Figuras 18(b), 18(d) e 18(f) apresentam os valores de perda para o treinamento dessas arquiteturas.

Outrossim, para resumir o desempenho do modelo em relação ao conjunto de dados testes foram analisadas as matrizes de confusão e as métricas de avaliação. A priori, as Figuras 19(a), 19(b) e 19(c) apresentam as matrizes de confusão das arquiteturas EfficientNetV2B0, InceptionV3 e ResNet101V2. As matrizes de confusão apresentam a quantidade de acertos que as arquiteturas alcançaram para a classe “Tratar” (VP) e para a classe “Não Tratar” (VN).

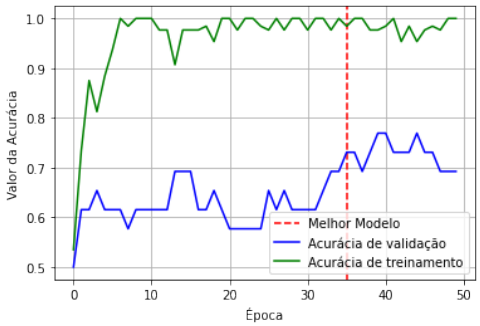
Figura 18 – Gráficos das acurácias e das perdas obtidas pelos modelos durante o treinamento, realizado com a base de dados original.



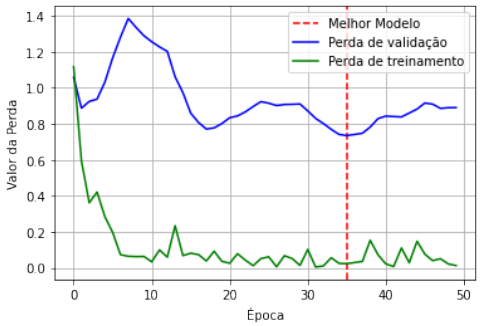
(a) Acurácia: Arquitetura EfficientNetV2B0.



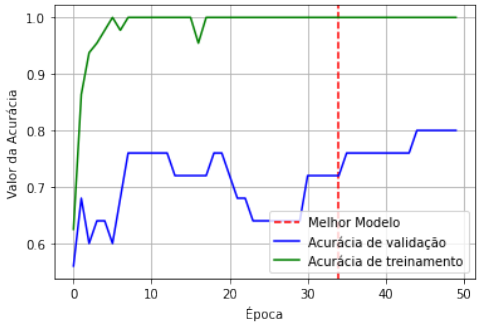
(b) Perda: Arquitetura EfficientNetV2B0.



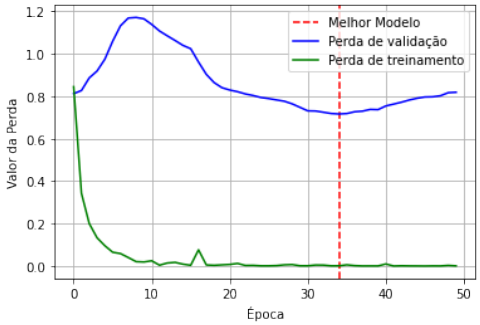
(c) Acurácia: Arquitetura InceptionV3.



(d) Perda: Arquitetura InceptionV3.



(e) Acurácia: Arquitetura ResNet101V2.



(f) Perda: Arquitetura ResNet101V2.

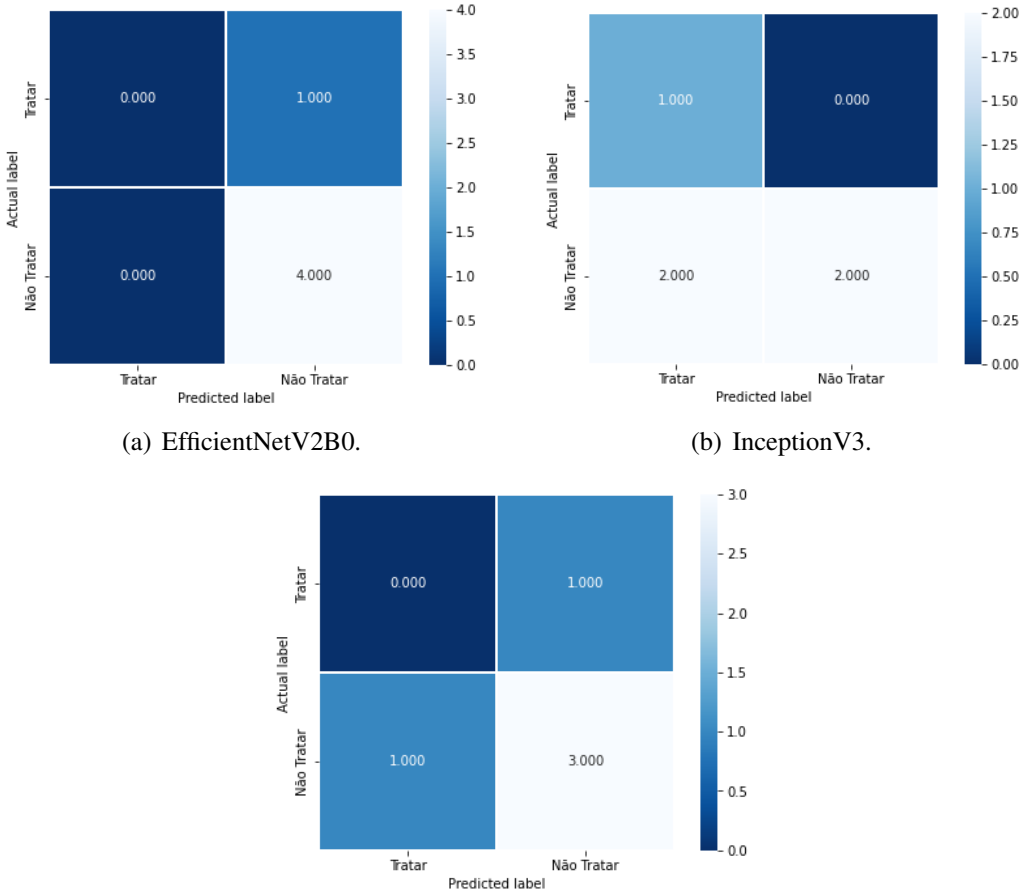
Fonte: Elaborado pela autora.

Portanto, foram utilizadas 5 imagens para realizar os testes. Dessa forma, o número de exemplos da classe positiva classificados corretamente (VP) e o número de exemplos da classe negativa classificados corretamente (VN) para cada arquitetura foram:

- EfficientNetV2B0: 4 imagens (VP) e 0 imagem (VN);
- InceptionV3: 2 imagens (VP) e 1 imagem (VN);
- ResNet101V2: e 3 imagens (VP) 0 imagem (VN);

Por fim, a Tabela 4 apresenta os valores obtidos para avaliar se o algoritmo está ou não conseguindo bons resultados. Logo, foram aplicadas as métricas de avaliação Precisão, *Recall* e *F1-Score* para determinar os resultados de cada arquitetura para o conjunto de teste.

Figura 19 – Matrizes de Confusão dos modelos treinados com a base de dados original.



(a) EfficientNetV2B0.

(b) InceptionV3.

(c) ResNet101V2.

Fonte: Elaborado pela autora.

Tabela 4 – Resultados das métricas de avaliação para os modelos treinados com a base de dados original.

	Precisão	**Recall**	**F1-Score**
EfficientNetV2B0	0,8	1,0	0,8889
InceptionV3	1,0	0,5	0,6666
ResNet101V2	0,75	0,75	0,75

Fonte: Elaborado pela autora.

4.2.2 Base de dados aumentada com métodos DA

Posteriormente, para realizar o treinamento dos modelos foram aplicados métodos de aumento de dados no conjunto de treino, aplicando transformações específicas nas imagens. As arquiteturas *CNN*s utilizadas para realizar a classificação foram a EfficientNetV2B0, a InceptionV3 e a ResNet101V2. Além disso, a validação cruzada *k-fold* foi realizada com o parâmetro $k = 4$. Desse modo, a Tabela 5 exibe os valores médios para acurácia de validação e para a perda dos modelos. Os resultados para a acurácia de validação e perda da arquitetura EfficientNetV2B0

foram, respectivamente, $55,83\%$ e $0,6954$. A arquitetura InceptionV3 obteve uma acurácia de validação de $59,86\%$ e um erro médio de 1.65. Ademais, a arquitetura ResNet101V2 apresentou o melhor resultado para acurácia de validação, obtendo valor igual a $62,08\%$ e $1,1409$ para o valor de perda.

Tabela 5 – Resultado médio de acurácia de validação e perda para os modelo treinados com a base de dados aumentada com métodos de DA tradicionais.

		Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Média**
EfficientNetV2B0	**Acurácia**	56,30%	61,03%	53,60%	52,40%	55,83%±0,0332
	Loss	0,7120	0,6596	0,7165	0,6936	0,6954
InceptionV3	**Acurácia**	59,07%	56,64%	62,64%	61,12%	59,86%±0,0225
	Loss	1,7231	1,8575	1,0401	2,0140	1,6587
ResNet101V2	**Acurácia**	69,76%	64,07%	46,80%	67,68%	**62,08%**±0,0905
	Loss	0,7810	0,9675	1,9819	0,8331	1,1409

Fonte: Elaborado pela autora.

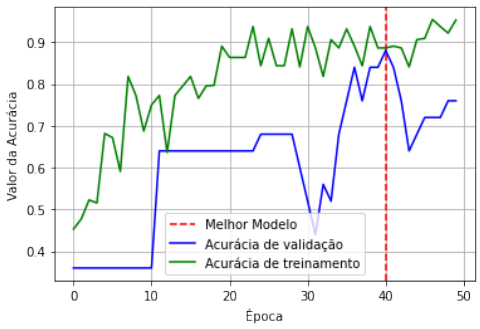
A Figura 20 exibe os gráficos representando os valores médios para acurácia e para a perda dos modelos durante o treinamento, com base no conjunto de treino expandido por transformações de imagens. Isto posto, a acurácia média das arquiteturas EfficientNetV2B0, InceptionV3 e ResNet101V2 são apresentados nas Figuras 20(a), 20(c) e 20(e). Dessarte, as representações das perdas durante o treinamento das arquiteturas EfficientNetV2B0, InceptionV3 e ResNet101V2 são exibidas nas Figuras 20(b), 20(d) e 20(f).

Além disso, é importante analisar o desempenho do modelo em relação aos dados de teste. Isto posto, as matrizes de confusão das arquiteturas EfficientNetV2B0, InceptionV3 e ResNet101V2 são apresentadas nas Figuras 21(a), 21(b) e 21(c). Portanto, de acordo com a matrizes de confusão, o número de exemplos da classe positiva classificados corretamente (VP), representando a classe "Não Tratar" e o número de exemplos da classe negativa classificados corretamente (VN), indicando a classe "Tratar" foram:

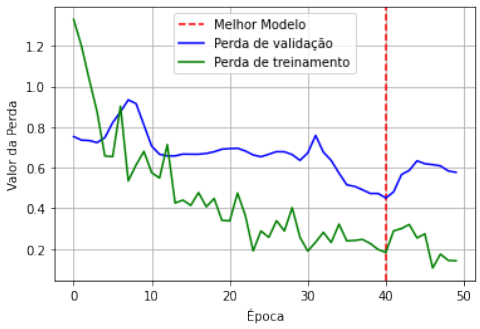
- EfficientNetV2B0: 3 imagem (VP) e 1 imagens (VN);
- InceptionV3: 3 imagem (VP) e 1 imagens (VN);
- ResNet101V2: 2 imagem (VP) e 0 imagens (VN);

Em relação às métricas de avaliação, foram calculados os valores de Precisão, *Recall* e *F1-Score*, a fim de determinar o desempenho dos modelos para o conjunto de teste. Portanto, a Tabela 6 apresenta os resultados obtidos para as métricas de avaliação.

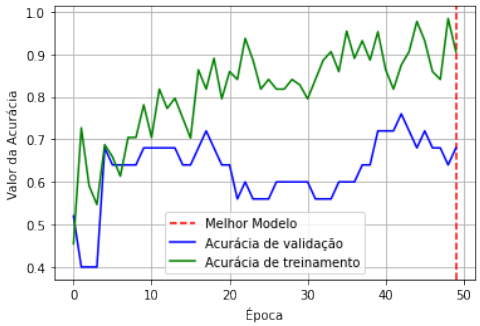
Figura 20 – Gráficos das acurácias e das perdas obtidas pelos modelos durante o treinamento, realizado com a base de dados aumentada com os métodos de DA tradicionais.



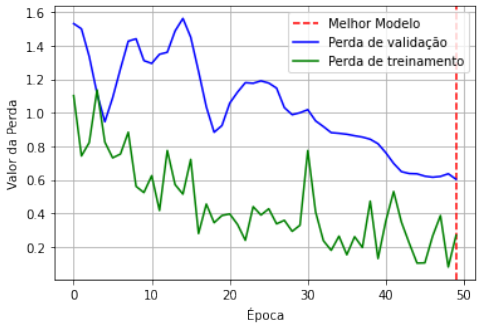
(a) Acurácia: Arquitetura EfficientNetV2B0.



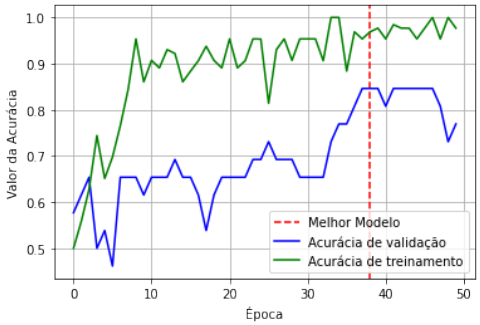
(b) Perda: Arquitetura EfficientNetV2B0.



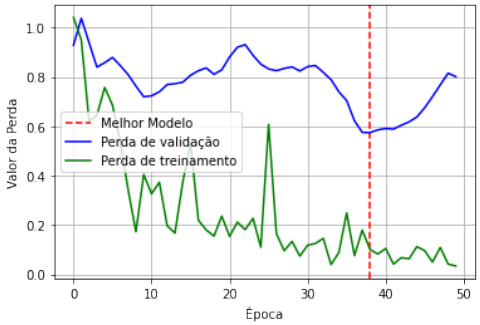
(c) Acurácia: Arquitetura InceptionV3.



(d) Perda: Arquitetura InceptionV3.



(e) Acurácia: Arquitetura ResNet101V2.



(f) Perda: Arquitetura ResNet101V2.

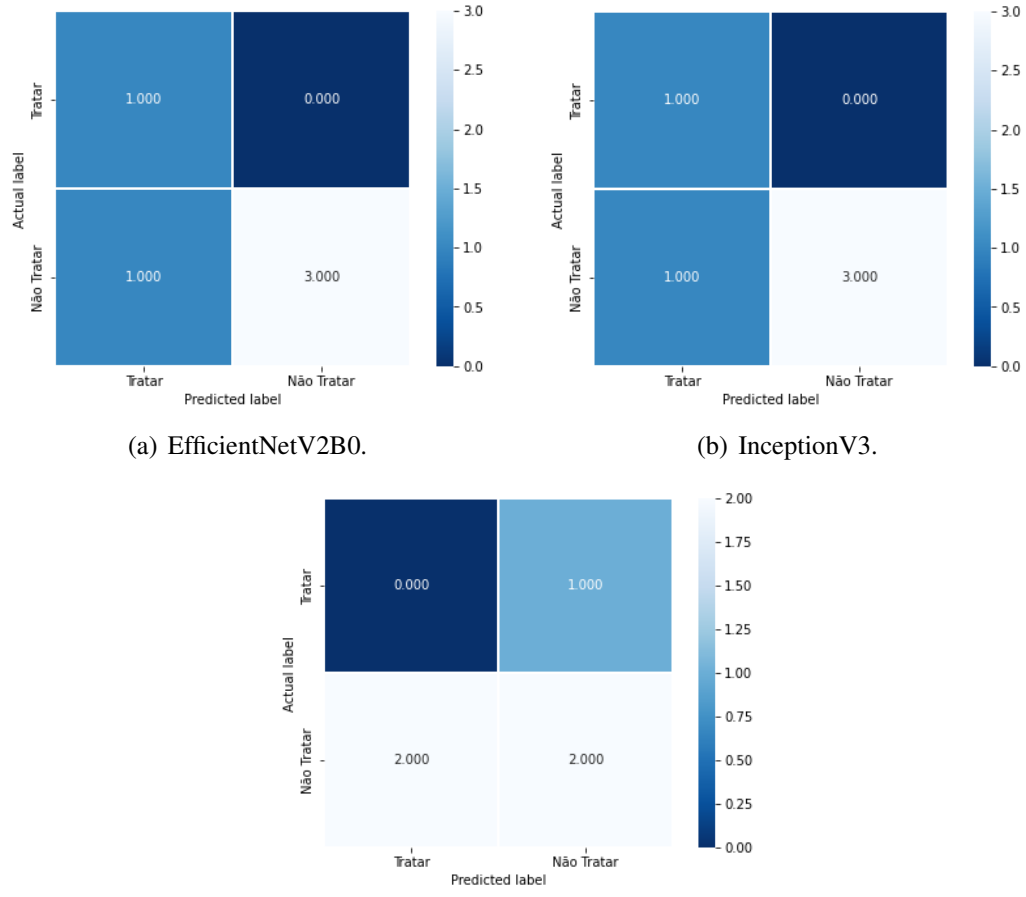
Fonte: Elaborado pela autora.

Tabela 6 – Resultados das métricas de avaliação para os modelos treinados com a base de dados aumentada com os métodos de DA tradicionais.

	Precisão	**Recall**	**F1-Score**
EfficientNetV2B0	1,0	0,75	0,8571
InceptionV3	1,0	0,75	0,8571
ResNet101V2	0,75	0,75	0,75

Fonte: Elaborado pela autora.

Figura 21 – Matrizes de Confusão dos modelos treinados com a base de dados aumentada com os métodos de DA tradicionais.



(a) EfficientNetV2B0.

(b) InceptionV3.

(c) ResNet101V2.

Fonte: Elaborado pela autora.

4.2.3 Base de dados com imagens da GAN

Ademais, para realizar treinamento dos modelos o conjunto de dados de treinamento foi expandido com as imagens geradas pelas *GAN*, totalizando em 484 imagens de treinamento, 5 imagens de teste e 17 imagens de validação. Novamente, foram usadas as arquiteturas EfficientNetV2B0, InceptionV3 e ResNet101V2 para realizar a classificação e a validação cruzada *k-fold* com $k = 4$. A Tabela 7 apresenta os valores de acurácia de validação e perda para os modelos. A priori, a arquitetura EfficientNetV2B0 obteve uma acurácia média de validação de $89,70\%$ e um erro médio de $0,2179$. A InceptionV3 apresentou acurácia média de validação de $92,61\%$ e um erro médio de $0,2159$. Já a arquitetura ResNet101V2 atingiu os valores de $95,41\%$ para acurácia média de validação e $0,1457$ para a perda.

Dando continuidade, a Figura 22 exibe os gráficos com o comportamento das acurácias e das perdas obtidas pelos modelos durante a etapa de treinamento, realizada com a

Tabela 7 – Resultado médio de acurácia de validação e perda para os modelo treinados com a base de dados aumentada com imagens geradas pela *GAN*.

		Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Média**
EfficientNetV2B0	**Acurácia**	86,68%	89,62%	91,97%	90,52%	89,70%±0,0193
	Loss	0,2928	0,2177	0,1719	0,1893	0.2179
InceptionV3	**Acurácia**	90,56%	94,14%	95,10%	90,66%	92,61%±0,0203
	Loss	0,2690	0,1464	0,1375	0,3105	0,2159
ResNet101V2	**Acurácia**	97,31%	93,31%	95,56%	95,45%	**95,41%**±0,0141
	Loss	0,0603	0,1823	0,1881	0,1881	0,1457

Fonte: Elaborado pela autora.

base de dados original expandida com imagens geradas pela *GAN*. As Figuras 22(a) e 22(b) apresentam os gráficos com as acurácias e os valores de perda, respectivamente, para a arquitetura EfficientNetV2B0. Em seguida, a Figura 22(c) indica o comportamento das acurácias para a arquitetura InceptionV3, enquanto a Figura 22(d) apresenta o gráfico de perda. Por fim, as Figuras 22(e) e 22(f) apresentam, respectivamente, os gráficos de acurácia e perda para a arquitetura ResNet101V2.

Em seguida, as Figuras 23(a), 23(a) e 23(a) exibem as matrizes de confusão para as arquiteturas EfficientNetV2B0, InceptionV3 e ResNet101V2, respectivamente. Logo, o número de exemplos representados como VP e o número de amostras classificados como VN na matriz de confusão foram:

- EfficientNetV2B0: 3 imagem (VP) e 1 imagens (VN);
- InceptionV3: 3 imagem (VP) e 1 imagens (VN);
- ResNet101V2: 4 imagem (VP) e 1 imagens (VN);

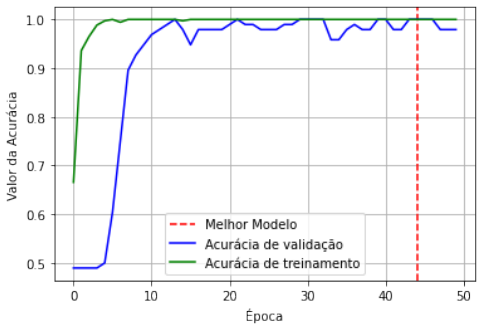
Já os valores encontrados para as métricas de avaliação Precisão, *Recall* e *F1-Score* são apresentados na Tabela 8.

Tabela 8 – Resultados das métricas de avaliação para os modelos treinados com a base de dados aumentada com imagens geradas pela *GAN*.

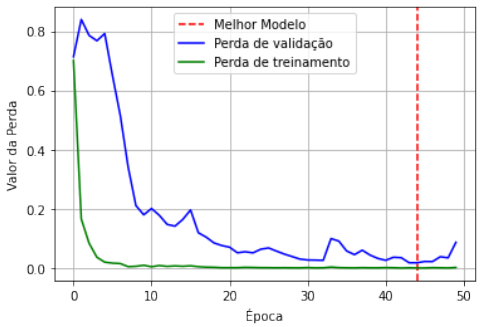
	Precisão	**Recall**	**F1-Score**
EfficientNetV2B0	1.0	0.75	0.8571
InceptionV3	1.0	0.75	0.8571
ResNet101V2	1.0	1.0	1.0

Fonte: Elaborado pela autora.

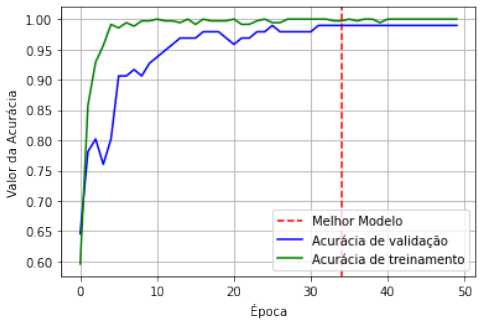
Figura 22 – Gráficos das acurácias e das perdas obtidas pelos modelos durante o treinamento, realizado com a base de dados aumentada com imagens geradas pela *GAN*.



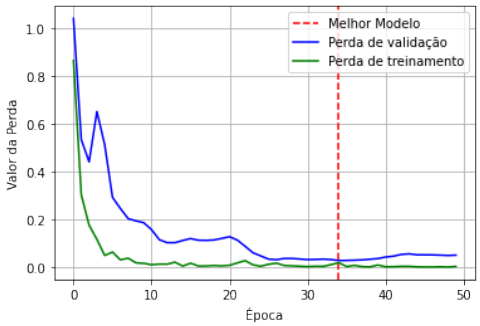
(a) Acurácia: Arquitetura EfficientNetV2B0.



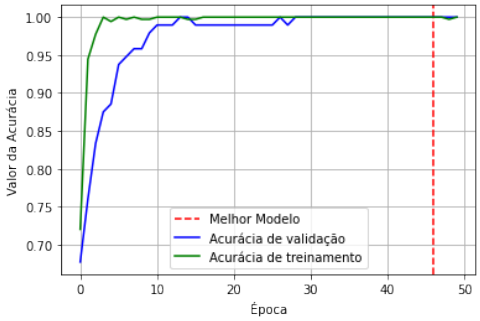
(b) Perda: Arquitetura EfficientNetV2B0.



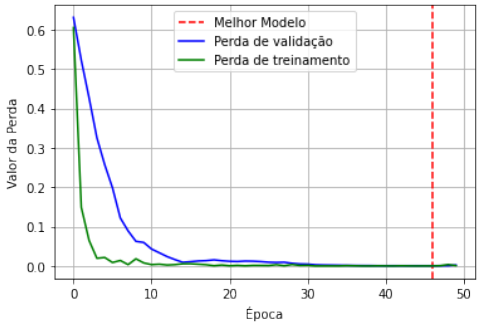
(c) Acurácia: Arquitetura InceptionV3.



(d) Perda: Arquitetura InceptionV3.



(e) Acurácia: Arquitetura ResNet101V2.



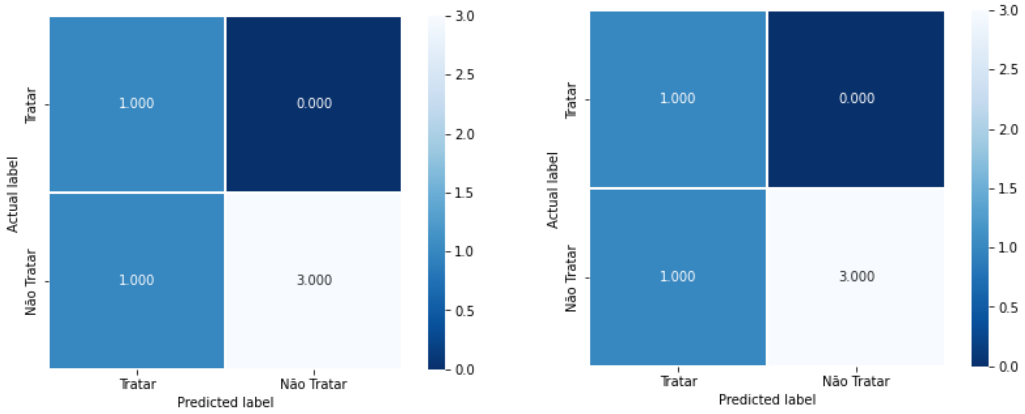
(f) Perda: Arquitetura ResNet101V2.

Fonte: Elaborado pela autora.

4.2.4 Discussão

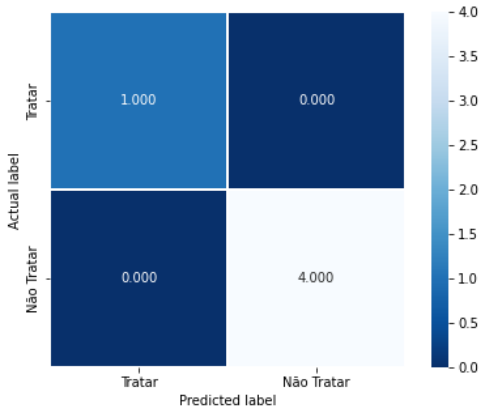
Nesse contexto, a arquitetura ResNet101V2 apresentou os melhores resultados para a acurácia média de validação para as três análises realizadas. Ademais, o treinamento realizado a partir da base de dados aumentada com transformações em imagens apresentou melhores resultados de acurácia média de validação e perda para as redes EfficientNetV2B0 e InceptionV3 em relação ao treinamento realizado com a base de dados original. Entretanto, a arquitetura ResNet101V2 apresentou resultados mais significativos ao ser treinada apenas com a base de dados original. Em contrapartida, os resultados de acurácia média de validação e perda foram

Figura 23 – Matrizes de Confusão dos modelos treinados com a base de dados aumentada com imagens geradas pela *GAN*.



(a) EfficientNetV2B0.

(b) InceptionV3.



(c) ResNet101V2.

Fonte: Elaborado pela autora.

superiores para todas as arquiteturas treinadas com a base de dados aumentada com as imagens geradas pela *GAN*, comprovando a eficiência da *GAN* como um método de aumento de dados.

Em relação à análise do desempenho do algoritmo, treinado com a base de dados original, em relação aos dados de teste, a arquitetura EfficientNetV2B0 se mostrou mais precisa em classificar imagens de animais saudáveis, enquanto a InceptionV3 classificou com mais precisão os animais com anemia. Outrossim, para o conjunto de dados expandido com aumento de dados tradicionais as arquiteturas EfficientNetV2B0 e InceptionV3 obtiveram o mesmo resultado de classificação. Por fim, já para a classificação realizada a partir do conjunto de dados expandido com imagens da *GAN*, todas as arquiteturas apresentaram excelência em classificar os animais com anemia, obtendo 100% de precisão. Além disso, a arquitetura ResNet101V2 avaliou corretamente todos os dados de teste.

5 CONSIDERAÇÕES FINAIS

Neste trabalho foi realizada a análise do impacto de métodos de aumento de dados em relação ao desempenho de classificadores, responsáveis por detectar a presença de anemia em ovinos, por meio das imagens da mucosa ocular desses animais. A priori, foi implementada a geração de imagens sintéticas, por meio da arquitetura *StyleGAN2-ADA*. Dessa forma, foram geradas 400 imagens distribuídas em rótulos de classe "Tratar", o que representa a presença de anemia, e "Não Tratar", indicando que o animal está saudável. Isto posto, para avaliar a qualidade das imagens geradas pela *GAN*, foi calculado a pontuação *FID*, que indica similaridade das imagens sintéticas com as imagens reais. O *FID* obtido nessa experiência foi de $74,44$.

Dessarte, para verificar o verdadeiro desempenho da *GAN* como um método de aumento de dados, foram analisadas o comportamento de diferentes arquiteturas *CNN* com a função de detectar a anemia em animais. A priori, foram utilizadas três as arquiteturas EfficientNetV2, InceptionV3 e ResNet101V2 com função de realizar o processo de classificação de imagens. Portanto, o treinamento dos modelos consistiu em três etapas: treinamento realizado com a base de dados original, em seguida, com o conjunto de treinamento expandido por técnicas de aumento de dados tradicionais e, por fim, com os dados de treinamento aumentados com as imagens geradas pela *GAN*. Com isso, foram analisados os resultados para os três cenários aqui expostos, possibilitando a comparação da influência dos métodos de aumento de dados em relação ao desempenho do classificador final.

Portanto, a arquitetura ResNet101V2 apresentou os melhores resultados de acurácia média de validação para os três cenários. Desse modo, obteve acurácia de validação de $65,38\%$ para o treinamento realizado com a base de dados original, $62,08\%$ para as técnicas de DA tradicionais e $95,41\%$ para a proposta da utilização de *GAN* como um método de DA. Ademais, as arquiteturas EfficientNetV2 e InceptionV3 demostraram melhores resultados de acurácia para o treinamento com as técnicas de DA, em relação ao treinamento feito com a base de dados original. Nesse contexto, as três arquiteturas utilizadas neste projeto obtiveram um melhor desempenho ao serem treinadas com as imagens geradas pela *GAN*.

Por fim, o desempenho dos modelos em relação aos dados de testes foi avaliado através das métricas de avaliações de Precisão, *Recall* e *F1-Score*. Dessa forma, a rede ResNet101V2 avaliou corretamente 100% dos dados de teste para o treinamento realizado com a base de dados expandida com imagens sintéticas. Logo, é possível concluir a eficiência da *GAN* como um método de aumento de dados para este projeto.

5.1 Trabalhos Futuros

Como trabalhos futuros pretende-se utilizar outras arquiteturas *GAN* para realizar a geração de imagens sintéticas da mucosa ocular de ovinos e comparar os resultados obtidos para a métrica de avaliação *FID* com o valor obtido neste trabalho, a fim de aumentar a similaridade entre as imagens geradas e originais.

O próximo passo seria utilizar as imagens geradas para balancear a base de dados utilizada, visto que, o rótulo de classe de animais saudáveis é mais pertinente em comparação ao rótulo de classe de animais doentes. Dessa forma, as imagens da *GAN* serviriam para balancear as duas classes, e consequentemente, gerar um conjunto de dados maior do que o utilizado neste projeto.

Por fim, pretende-se analisar outras arquiteturas *CNN* utilizadas para classificar as imagens de conjuntiva ocular dos animais, além de buscar diferentes técnicas para avaliar mais precisamente o desempenho do classificador.

REFERÊNCIAS

ABIRAMI, R. N.; VINCENT, P. M. D. R.; SRINIVASAN, K.; TARIQ, U.; CHANG, C.-Y. Deep cnn and deep gan in computational visual perception-driven image analysis. **Complexity**, v. 2021, p. 1–30, 2021.

ACADEMY, D. S. **Deep Learning Book.** [*S. l.*: *s. n.*], 2022. https://www.deeplearningbook.com.br/.

ACADEMY, I. E. **Funções de ativação: definição, características, e quando usar cada uma**. 2020. Urlhttps://iaexpert.academy/2020/05/25/funcoes-de-ativacao-definicao-caracteristicas-e-quando-usar-cada-uma/.

ALMEIDA, A. M. A. **Detecção de Anemia em Ovinos através de Aprendizagem Profunda em Imagens de Mucosa Ocular.** Dissertação (Mestrado) – Universidade Federal do Ceará, 2021.

ALZUBAIDI, L.; ZHANG, J.; HUMAIDI, A. J.; AL-DUJAILI, A.; DUAN, Y.; AL-SHAMMA, O.; SANTAMARíA, J.; FADHEL, M. A.; AL-AMIDIE, M.; FARHAN, L.; AL. et. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. **Journal of Big Data**, v. 8, n. 1, 2021.

ALZUBI, J.; NAYYAR, A.; KUMAR, A. Machine learning from theory to algorithms: An overview. **Journal of Physics: Conference Series**, IOP Publishing, v. 1142, p. 012012, nov 2018. Disponível em: https://doi.org/10.1088/1742-6596/1142/1/012012.

ANAYA-ISAZA, A.; MERA-JIMéNEZ, L.; ZEQUERA-DIAZ, M. An overview of deep learning in medical imaging. **Informatics in Medicine Unlocked**, v. 26, p. 100723, 2021. ISSN 2352-9148. Disponível em: https://www.sciencedirect.com/science/article/pii/S2352914821002033.

BARELLI, F. **Introdução à Visão Computacional: Uma abordagem prática com Python e OpenCV**. [*S. l.*]: Casa do Código, 2018. ISBN 9788594188588.

BATH, G. F.; F.S., M.; WYK, J. A. V. The "famacha "ovine anaemia guide to assist with the control of haemonchosis. In: **Proceedings of the 7th Annual Congress of the Livestock Health and Production Group of the South African Veterinary Association**. [*S. l.*: *s. n.*], 1996. p. 5.

BERRAR, D. Cross-validation. In: ____. [*S. l.*: *s. n.*], 2018. ISBN 9780128096338.

BIRGEL, D. B. Estudo da anemia em ovinos decorrente a verminose gastrintestinal. **Tese (Doutorado em Clínica Veterinária) - Faculdade de Medicina Veterinária e Zootecnia, Universidade de São Paulo**, Tese (Doutorado em Clínica Veterinária) - Faculdade de Medicina Veterinária e Zootecnia, Universidade de São Paulo, v. 1, n. 1, 2013.

BISSOTO, A.; VALLE, E.; AVILA, S. Gan-based data augmentation and anonymization for skin-lesion analysis: A critical review. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops**. [*S. l.*: *s. n.*], 2021. p. 1847–1856.

BORJI, A. Pros and cons of gan evaluation measures: New developments. **Computer Vision and Image Understanding**, Elsevier, v. 215, p. 103329, 2022.

BROWNLEE, J. **Loss and Loss Functions for Training Deep Learning Neural Networks**. 2019. Urlhttps://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/.

BROWNLEE, J. **How to Choose an Activation Function for Deep Learning**. 2021. Urlhttps://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/.

CHAGAS, A. C. de S.; CARVALHO, C. O. de; MOLENTO, M. B. **Método famacha: um recurso para o controle da verminose em ovinos.** 2007. Disponível em: https://ainfo.cnptia.embrapa.br/digital/bitstream/item/37274/1/Circular52.pdf.

CHLAP, P.; MIN, H.; VANDENBERG, N.; DOWLING, J.; HOLLOWAY, L.; HAWORTH, A. A review of medical image data augmentation techniques for deep learning applications. **Journal of Medical Imaging and Radiation Oncology**, Wiley Online Library, v. 65, n. 5, p. 545–563, 2021.

CLOUD, G. **Advanced Guide to Inception v3**. 2022. Urlhttps://cloud.google.com/tpu/docs/inception-v3-advanced.

CS231N. **Convolutional Neural Networks (CNNs / ConvNets)**. 2022. Urlhttps://cs231n.github.io/convolutional-networks/pool.

CSÁJI, B. C. *et al.* Approximation with artificial neural networks. **Faculty of Sciences, Etvs Lornd University, Hungary**, v. 24, n. 48, p. 7, 2001.

CUNNINGHAM, P.; CORD, M.; DELANY, S. J. Supervised learning. In: ____. **Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 21–49. ISBN 978-3-540-75171-7. Disponível em: https://doi.org/10.1007/978-3-540-75171-7_2.

de Souza, L. A.; PASSOS, L. A.; MENDEL, R.; EBIGBO, A.; PROBST, A.; MESSMANN, H.; PALM, C.; PAPA, J. P. Assisting barrett's esophagus identification using endoscopic data augmentation based on generative adversarial networks. **Computers in Biology and Medicine**, v. 126, p. 104029, 2020. ISSN 0010-4825. Disponível em: https://www.sciencedirect.com/science/article/pii/S0010482520303607.

DEMOLINER, G.; ALVES, R. J. F. Anemimetro: app móvel para implementação do método famacha. **Unoesc amp; Ciência - ACET**, v. 8, n. 1, p. 25–32, jun. 2017.

DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. **arXiv preprint arXiv:1603.07285**, 2016.

ELOY Ângela M. X.; COSTA, A. L. da; CAVALCANTE, A. C. R.; SILVA, E. R.; SOUSA, F. B. de; SILVA, F. L. R. da; ALVES, F. S. F.; VIEIRA, L. da S.; BARROS, N. N.; PINHEIRO, R. R. **Criação de caprinos e ovinos.** 2007. Disponível em: https://ainfo.cnptia.embrapa.br/digital/bitstream/item/11945/2/00081710.pdf.

EMBRAPA. **Centro de Inteligência e Mercado de Caprinos e Ovinos**. 2020. Disponível em: https://www.embrapa.br/cim-inteligencia-e-mercado-de-caprinos-e-ovinos/apresentacao. Acesso em: 17 jan. 2021.

EMBRAPA, E. C. e O. **Método FAMACHA**. 2017. Urlhttps://www.embrapa.br/paratec-controle-integrado-verminoses/vermes/caprinos-ovinos/famacha.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. d. L. F. d. **Inteligência artificial: uma abordagem de aprendizado de máquina**. [*S. l.*]: LTC, 2011.

FERLITSCH, A. **Deep Learning Patterns and Practices**. edição. Endereço: Manning Publications Co. LLC, 2021. volume. (serie, volume). Anotação. ISBN 9781617298264.

FRID-ADAR, M.; KLANG, E.; AMITAI, M.; GOLDBERGER, J.; GREENSPAN, H. Synthetic data augmentation using gan for improved liver lesion classification. In: **2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)**. [*S. l.*: *s. n.*], 2018. p. 289–293.

GARDNER, M.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. **Atmospheric Environment**, v. 32, n. 14, p. 2627–2636, 1998. ISSN 1352-2310. Disponível em: https://www.sciencedirect.com/science/article/pii/S1352231097004470.

GOODFELLOW, I. J.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. **Generative Adversarial Networks**. 2014.

GUO, G.; ZHANG, N. A survey on deep learning based face recognition. **Computer Vision and Image Understanding**, v. 189, p. 102805, 2019. ISSN 1077-3142. Disponível em: https://www.sciencedirect.com/science/article/pii/S1077314219301183.

GUSTINELI, M. **A survey on recently proposed activation functions for Deep Learning**. arXiv, 2022. Disponível em: https://arxiv.org/abs/2204.02921.

HAYKIN, S. S. **Neural networks and learning machines**. Third. Upper Saddle River, NJ: Pearson Education, 2009.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. **CoRR**, abs/1512.03385, 2015. Disponível em: http://arxiv.org/abs/1512.03385.

HEUSEL, M.; RAMSAUER, H.; UNTERTHINER, T.; NESSLER, B.; HOCHREITER, S. **GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium**. 2018.

HINTON, G.; SRIVASTAVA, N.; SWERSKY, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. **Cited on**, v. 14, n. 8, p. 2, 2012.

IBGE, I. B. de Geografia e E. Pesquisa da pecuária municipal - ppm. 2020. ISSN 01014234.

Jan A. Van Wyk; Gareth F. Bath. The famacha system for managing haemonchosis in sheep and goats by clinically identifying individual animals for treatment. **Vet. Res.**, v. 33, n. 5, p. 509–529, 2002. Disponível em: https://doi.org/10.1051/vetres:2002036.

KARRAS, T.; AITTALA, M.; HELLSTEN, J.; LAINE, S.; LEHTINEN, J.; AILA, T. Training generative adversarial networks with limited data. **CoRR**, abs/2006.06676, 2020. Disponível em: https://arxiv.org/abs/2006.06676.

KARRAS, T.; LAINE, S.; AILA, T. A style-based generator architecture for generative adversarial networks. **CoRR**, abs/1812.04948, 2018. Disponível em: http://arxiv.org/abs/1812.04948.

KARRAS, T.; LAINE, S.; AITTALA, M.; HELLSTEN, J.; LEHTINEN, J.; AILA, T. Analyzing and improving the image quality of stylegan. **CoRR**, abs/1912.04958, 2019. Disponível em: http://arxiv.org/abs/1912.04958.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v. 25, 2012.

LACVET, L. de A. C. V. **Hematócrito**. 2022. Disponível em: https://www.ufrgs.br/lacvet/hematocrito/.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998.

MAGDALENA, R.; SAIDAH, S.; UBAIDAH, I. D. S.; FUADAH, Y. N.; HERMAN, N.; IBRAHIM, N. Convolutional neural network for anemia detection based on conjunctiva palpebral images. v. 3, p. 349–354, Apr. 2022. Disponível em: http://jutif.if.unsoed.ac.id/index.php/jurnal/article/view/197.

MALAN, F.; WYK, J. V.; WESSELS, C. Clinical evaluation of anaemia in sheep: Early trials. **The Onderstepoort journal of veterinary research**, v. 68, p. 165–74, 10 2001.

MASI, I.; TRAN, A. T.; LEKSUT, J. T.; HASSNER, T.; MEDIONI, G. G. Do we really need to collect millions of faces for effective face recognition? **CoRR**, abs/1603.07057, 2016. Disponível em: http://arxiv.org/abs/1603.07057.

MIN, S.; LEE, B.; YOON, S. Deep learning in bioinformatics. **Briefings in Bioinformatics**, v. 18, n. 5, p. 851–869, 07 2016. ISSN 1467-5463. Disponível em: https://doi.org/10.1093/bib/bbw068.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. In: **Sistemas Inteligentes Fundamentos e Aplicações**. 1. ed. Barueri-SP: Manole Ltda, 2003. p. 89–114. ISBN 85-204-168.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: **Icml**. [*S. l.*: *s. n.*], 2010.

NASER, M. Z.; ALAVI, A. Insights into performance fitness and error metrics for machine learning. **CoRR**, abs/2006.00887, 2020. Disponível em: https://arxiv.org/abs/2006.00887.

PAN, Z.; YU, W.; YI, X.; KHAN, A.; YUAN, F.; ZHENG, Y. Recent progress on generative adversarial networks (gans): A survey. **IEEE Access**, v. 7, p. 36322–36333, 2019.

PEREZ, L.; WANG, J. The effectiveness of data augmentation in image classification using deep learning. **CoRR**, abs/1712.04621, 2017. Disponível em: http://arxiv.org/abs/1712.04621.

QIAN, N. On the momentum term in gradient descent learning algorithms. **Neural Networks**, v. 12, n. 1, p. 145–151, 1999. ISSN 0893-6080. Disponível em: https://www.sciencedirect.com/science/article/pii/S0893608098001166.

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. V. Searching for activation functions. **CoRR**, abs/1710.05941, 2017. Disponível em: http://arxiv.org/abs/1710.05941.

RASAMOELINA, A. D.; ADJAILIA, F.; SINčáK, P. A review of activation function for artificial neural network. In: **2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)**. [*S. l.*: *s. n.*], 2020. p. 281–286.

RAVì, D.; WONG, C.; DELIGIANNI, F.; BERTHELOT, M.; ANDREU-PEREZ, J.; LO, B.; YANG, G.-Z. Deep learning for health informatics. **IEEE Journal of Biomedical and Health Informatics**, v. 21, n. 1, p. 4–21, 2017.

REFAEILZADEH, P.; TANG, L.; LIU, H. Cross-validation. In: ____. **Encyclopedia of Database Systems**. New York, NY: Springer New York, 2016. p. 1–7. ISBN 978-1-4899-7993-3. Disponível em: https://doi.org/10.1007/978-1-4899-7993-3_565-2.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, p. 65–386, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Representations by Back-propagating Errors. **Nature**, v. 323, n. 6088, p. 533–536, 1986. Disponível em: http://www.nature.com/articles/323533a0.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of Research and Development**, v. 3, n. 3, p. 210–229, 1959.

SCIKIT-LEARN. **3.1. Cross-validation: evaluating estimator performance**. 2007 – 2022. Urlhttps://scikit-learn.org/stable/modules/cross$_v$*alidation.html*.

SHARMA, S.; SHARMA, S.; ATHAIYA, A. Activation functions in neural networks. **International Journal of Engineering Applied Sciences and Technology**, v. 04, p. 310–316, 05 2020.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. **Journal of big data**, SpringerOpen, v. 6, n. 1, p. 1–48, 2019.

SILVA, G.; AMORIM, B.; QUIRINO, A.; SILVA, A.; FARIAS, L. Haemonchus contortus em ovinos e caprinos. **Pubvet**, v. 13, p. 1–4, 09 2019.

SILVA, L.; ARAUJO, L.; SOUZA, V.; SANTOS, A.; NETO, R. Redes neurais convolucionais aplicadas na detecção de pneumonia através de imagens de raio-x. In: . [*S. l.*: *s. n.*], 2020. p. 1–8.

SWIDERSKI, B.; GIELATA, L.; OLSZEWSKI, P.; OSOWSKI, S.; KOłODZIEJ, M. Deep neural system for supporting tumor recognition of mammograms using modified gan. **Expert Systems with Applications**, v. 164, p. 113968, 2021. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417420307491.

SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; WOJNA, Z. Rethinking the inception architecture for computer vision. **CoRR**, abs/1512.00567, 2015. Disponível em: http://arxiv.org/abs/1512.00567.

TALEBI, H.; MILANFAR, P. Learning to resize images for computer vision tasks. **CoRR**, abs/2103.09950, 2021. Disponível em: https://arxiv.org/abs/2103.09950.

TAN, M.; LE, Q. V. Efficientnetv2: modelos menores e treinamento mais rápido. abs/2104.00298, 2021. Disponível em: https://arxiv.org/abs/2104.00298.

TAYLOR, L.; NITSCHKE, G. **Improving Deep Learning using Generic Data Augmentation**. arXiv, 2017. Disponível em: https://arxiv.org/abs/1708.06020.

TING, K. M. Confusion matrix. In: ____. **Encyclopedia of Machine Learning**. Boston, MA: Springer US, 2010. p. 209–209. ISBN 978-0-387-30164-8. Disponível em: https://doi.org/10.1007/978-0-387-30164-8_157.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: SN. **Proceedings of the xxix conference on graphics, patterns and images**. [*S. l.*], 2016. v. 1, n. 4.

VIEIRA, L. da S. Métodos alternativos de controle de nematóides gastrintestinais em caprinos e ovinos. In: **In: SIMPÓSIO INTERNACIONAL SOBRE CAPRINOS E OVINOS DE CORTE, 3; FEIRA NACIONAL DO AGRONEGÓCIO DA CAPRINO-OVINOCULTURA DE CORTE, 2007, João Pessoa. Anais... João Pessoa: EMEPA-PB, 2007. 12 f. 1 CD-ROM.** João Pessoa: [*S. n.*], 2007. v. 2, p. 49–56.

WAHEED, A.; GOYAL, M.; GUPTA, D.; KHANNA, A.; AL-TURJMAN, F.; PINHEIRO, P. R. Covidgan: Data augmentation using auxiliary classifier gan for improved covid-19 detection. **IEEE Access**, v. 8, p. 91916–91923, 2020.

WANG, S.-C. Artificial neural network. In: ____. **Interdisciplinary Computing in Java Programming**. Boston, MA: Springer US, 2003. p. 81–100. ISBN 978-1-4615-0377-4. Disponível em: https://doi.org/10.1007/978-1-4615-0377-4_5.

WANG, Z.; SHE, Q.; WARD, T. E. Generative adversarial networks: A survey and taxonomy. **CoRR**, abs/1906.01529, 2019. Disponível em: http://arxiv.org/abs/1906.01529.

WOOLF, B. P. Chapter 7 - machine learning. In: WOOLF, B. P. (Ed.). **Building Intelligent Interactive Tutors**. San Francisco: Morgan Kaufmann, 2009. p. 221–297. ISBN 978-0-12-373594-2. Disponível em: https://www.sciencedirect.com/science/article/pii/B9780123735942000071.

WYK, J. A. V.; MALAN, F. S.; BATH, G. F. Rampant anthelmintic resistance in sheep in south africa - what are the options? **MANAGING ANTHELMINTIC RESISTANCE IN ENDOPARASITES. [S.l.]: Conference of the World Association for the Advancement of Veterinary Parasitology (WAAVP)**, p. 51–63, 1997.

YAMASHITA, R.; NISHIO, M.; DO, R.; TOGASHI, K. Convolutional neural networks: an overview and application in radiology. **Insights into Imaging**, v. 9, 06 2018.

YU, Y.; ZHANG, W.; DENG, Y. **Frechet Inception Distance (FID) for Evaluating GANs**. 2021.