Avaliação do Uso de visão computacional para Detecção de Filhotes de Tartarugas Marinhas

Evaluation of the Computer Vision for Detecting Sea Turtles Hatchling

Odin M. dos Santos Oliveira[1], Izabel M. dos Santos[1], Flávio R. S. Oliveira[1]

[1] Curso Tecnológico em Análise e Desenvolvimento de Sistemas
Instituto Federal de Ensino, Ciência e Tecnologia de Pernambuco (IFPE)
Paulista, PE

omso@discente.ifpe.edu.br, ims11@discente.ifpe.edu.br

flavio.oliveira@paulista.ifpe.edu.br

Resumo. *No Brasil pode-se encontrar cinco das sete espécies de tartarugas marinhas existentes no mundo. Nos ninhos, a contagem de filhotes nascidos, ovos rachados e natimortos é realizada por profissionais especializados. Como uma ferramenta poderosa, a Inteligência Artificial pode ser utilizada para auxiliar nesse processo, melhorando a coleta de dados para segurança dessa espécie. Este artigo apresenta um estudo utilizando Aprendizagem Profunda como ferramenta para auxiliar profissionais de segurança ambiental na contagem de tartarugas marinhas. Mais especificamente, foram aplicadas quatro arquiteturas de Rede Neural Convolucional (VGG16, ResNet, MobileNet e DenseNet) para avaliar a viabilidade de sua aplicação no problema em questão. Foram obtidos valores satisfatórios nos experimentos, atingindo 0,97 no F1-Score, medida de desempenho que avalia a precisão e a recall, indicando um desempenho promissor na detecção das tartarugas marinhas. Esses resultados encorajam a continuidade de pesquisas nesta área, visando avançar na aplicação de Visão Computacional para criar sistemas de monitoramento e segurança ambiental.*
Palavras-chave: Tartarugas Marinhas; Monitoramento de ninhos; Aprendizagem Profunda; Redes Neurais Convolucionais.

Abstract. *In Brazil, five out of the seven existing species of marine turtles can be found. Professionals specialized in this field carry out the counting of hatched offspring, cracked eggs, and stillbirths in nests. As a powerful tool, Artificial Intelligence can be used to assist in this process, enhancing the data collection for the safety of this species. This article presents a study using Deep Learning as a tool to aid environmental safety professionals in the counting of marine turtles. Specifically, four Convolutional Neural Network architectures (VGG16, ResNet, MobileNet, and DenseNet) were applied to assess their feasibility in addressing this problem. Satisfactory results were achieved in the experiments, reaching 0.97 in the F1-Score, a performance measure evaluating precision and recall, indicating a promising performance in detecting marine turtles. These findings encourage further research in this area, aiming to advance the application of Computer Vision in creating monitoring systems for environmental safety.*
Keywords: Sea Turtles, Nest Monitoring, Deep Learning, Convolutional Neural Networks

---

1. Introdução

Como répteis que surgiram há 150 milhões de anos, resistindo a drásticas mudanças na Terra, que causaram a extinção dos dinossauros, as tartarugas marinhas mantiveram sua morfologia intacta, sem mudanças significativas para os tempos atuais (Baptistotte, 1992).

São animais de natureza migratória, retornando a mesma praia na que nasceram para desovar. Pesam cerca de 20 gramas ao nascer e podem chegar até 900 quilogramas quando adultas. Uma fêmea pode colocar cerca de 500 ovos, divididos em 3 a 6 posturas intercaladas por intervalos de 15 dias, deixando o Sol e a umidade se encarregarem de chocar os ovos após 50 dias da postura. A taxa de eclosão varia entre 50% a 80% (Baptistotte, 1992).

Além disso,

> As cinco espécies que habitam são encontradas no litoral do estado de Pernambuco, sendo elas: Tartaruga-verde (Chelonia mydas), Tartaruga-cabeçuda (Caretta caretta), Tartaruga-oliva (Lepidochelys olivacea), Tartaruga de couro (Dermochelys Coriacea), finalizando a lista temos a Tartaruga-de-pente (Eretmochelys imbricata), uma das especies mais abundantes na costa pernambucana (Bonfim *et al.*, 2021).

Vale salientar que cinco das setes espécies que se encontram ao redor do mundo, se reproduzem no Brasil, apesar de todas estarem ameaçadas de extinção há décadas. E devido a isso, são protegidas por leis nacionais e internacionais.

Segundo Marcovaldi (1999) com a promulgação da proteção integral para todas as espécies de tartarugas em 1986, e à conta das grandes pressões internacionais, foi criado o Programa Nacional de Conservação de Tartarugas Marinhas no Brasil (Projeto TAMAR), que está vinculado ao Instituto Brasileiro de Meio Ambiente e Recursos Naturais Renováveis (Ibama), com o objetivo de quantificar o número de espécies, distribuição e abundância de tartarugas marinhas, assim como a sazonalidade e abrangência geográfica da postura e as principais ameaçasà sobrevivência das tartarugas.

Atualmente, segundo registros do Projeto TAMAR e do Programa de monitoramento de Tartarugas Marinhas do Parque Nacional Marinho dos Abrolhos, estes registros de nascimento são feitos manualmente. O ninho encontrado é numerado e marcado (com uma estaca de madeira) e no 45° dia após a postura, o ninho deve ser monitorado diariamente para se obter o dia de nascimento dos filhotes. Visto que o período de incubação dos filhotes varia de 45 a 60 ou até mesmo 66 dias. O que demanda tempo e um time composto por cerca de 52 pesquisadores, 105 estagiários e trainees, além de 59 tartarugueiros e agentes locais, que ficam dedicados exclusivamente ao monitoramento das praias para garantir a proteção das fêmeas, ninhos e filhotes até a eclosão dos ovos.

De acordo com o objetivo do Projeto TAMAR citado anteriormente, foi percebida a oportunidade de utilizar Inteligência Artificial, mais especificamente as Redes Neurais Convolucionais, para monitorar os ninhos e fazer a classificação e contagem de tartarugas marinhas após a eclosão.

As Redes Neurais Convolucionais (CNN — *Convolutional Neural Networks*), desde a última década, se tornaram uma abordagem amplamente dominante para reconhecimento de objetos (Huang *et al.*, 2017). Tendo sido propostas há mais de vinte anos atrás (LeCun *et al.*, 1989), sua aplicação foi finalmente possível recentemente, com o avanço da Internet e melhoria dos componentes de Hard-

ware. Este trabalho propõe avaliar o uso de CNNs como auxílio em estratégias de preservação da fauna local. Nesse caso, realizando a classificação de imagens contendo o nosso objeto de estudo: as tartarugas marinhas.

O objetivo é explorar e avaliar o uso de Redes Neurais Convolucionais na classificação de imagens de tartarugas marinhas, visto que esta área tem crescido rapidamente, e que possui potencial para agregar valor na solução do problema em questão. Foram escolhidas as arquiteturas ResNet, DenseNet, MobileNet e VGG16 pois foram as consideradas mais viáveis em relação ao seu custo computacional e histórico de trabalhos acerca de seu uso, utilizados como base para a esta pesquisa.

Este artigo está organizado da seguinte forma: na Seção 2 será apresentada a fundamentação teórica para a realização deste trabalho, serão abordados temas como Aprendizagem de Máquina, Redes Neurais Convolucionais e o uso de Redes Neurais Convolucionais em projetos ambientais. Na Seção 3 serão evidenciados os materiais e métodos de estudo utilizados neste artigo, como a base de dados foi montada e detalhes sobre a avaliação realizada. Na Seção 4 serão exibidos os experimentos feitos e seus resultados. Por fim, na Seção 5 serão realizadas as considerações finais sobre trabalho desenvolvido.

2. Referencial Teórico

Neste capítulo será apresentado a fundamentação teórica do trabalho em questão. Estarão englobados conceitos que definem Aprendizagem de Máquina, Redes Neurais Convolucionais e o Uso de Redes Neurais Convolucionais em projetos ambientais, como abordado por (Gray *et al*., 2019), com utilização de imagens tiradas por drones ou VANTs (Veículo Aéro Não-Tripulado) para detecção de tartarugas marinhas no litoral da Costa-rica.

2.1. Aprendizagem de máquina

Para melhor compreendermos a Aprendizagem de Máquina (AM), precisamos primeiro entender o que é aprendizagem. Segundo (Junior; Borges-Andrade, 2008) os principais usos cotidianos do conceito de aprendizagem remetem à aquisição de algum tipo de conhecimento ou habilidade, por meio de atividades formais de instrução. Tais conhecimentos e habilidades são direcionados a algum tipo de desempenho. Os indivíduos aplicam o que foi adquirido para alguma finalidade, ou seja, a aprendizagem pode ser lida como a aquisição e retenção de determinado conhecimento, conduzido à alguma finalidade.

No presente contexto, Aprendizagem de Máquina é uma sub-área da Inteligência Artificial, que estuda métodos computacionais para adquirir novos conhecimentos, novas habilidades e novos meios de organizar o conhecimento já existente (Mitchell, 1997). Também podemos definir AM como processo de resolução de problemas práticos por meio de coleta de dados e da construção algorítmica de um modelo estatístico baseado nos dados coletados (Burkov, 2019).

Na resolução de problemas de classificação, é necessária atribuição de um rótulo — um elemento de um conjunto de classes — a um exemplo não rotulado. A classificação necessita de uma coleção de rótulos iniciais e a partir deles irá inferir a classe das novas entradas. Nos classificadores usados no reconhecimento facial, por exemplo, os rótulos de saída dos programas não são originalmente rótulos, são uma probabilidade de a face fornecida como entrada corresponda a uma outra presente nos dados. A partir de um liminar desta probabilidade — por exemplo, 95% — o classificador produz uma saída binária: verdadeiro (para exemplos com chances menores do que 95%) ou falso (para chances a a partir de 95%) (Ruback. Avila; Cantero; 2021).

Burkov (2019) afirma que, "após a construção do modelo através dos dados de treinamento, os dados de teste, que são compostos por exemplos diferentes dos aprendidos antes, são os responsáveis por avaliar se o modelo em questão generaliza bem".

Em resumo, AM para detecção de objetos, trata-se de treinar o modelo fazendo a inserção de dados com um número relevante de imagens já rotuladas (por ex: centenas de imagens da classe "tartarugas"; milhares de imagens da classe "gatos"; etc.) para que o modelo, a partir do treinamento, consiga detectar e classificar as classes em novas imagens que não foram vistas anteriormente.

Uma das formas de fazer classificação é através de Redes Neurais, um sub-conjunto da AM, que tem como característica marcante sua arquitetura inspirada em como o as redes de neurônios do cérebro humano funciona.

2.2. Redes Neurais Convolucionais

Com a evolução da tecnologia e o '*Boom*' das Redes Neurais Artificiais (RNA) na década de 80, no final da década houve o surgimento das Redes Neurais Convolucionais (CNN). Em 1989 LeCun *et al.* apresentaram pela primeira vez a chamada ConvNet, uma Rede Neural Convolucional que obteve resultados bem-sucedidos para problemas de reconhecimento de dígitos e CEP escritos a mão. Anos mais tarde, essa arquitetura foi melhorada: chamada de LeNet-5, essa rede já era capaz de reconhecer variantes rotacionais das imagens. Sua boa performance permitiu que fossem usadas comercialmente em caixas eletrônicos e em bancos, em 1993 e 1996 respectivamente (Lecun *et al*., 1998; KHAN et al., 2020 *apud*. (Cunha, 2020)).

De acordo com (Cunha, 2020), a ascenção das CNN's se deu por volta de 2012 – 2014, com uma série de tentativas de aprimoramento. No ano de 2014, em Oxford um grupo apresentava uma arquitetura chamada VGG, com o tamanho de filtros muito menores, em comparação às anteriores, além do número de camadas que passou de 9 para 16. Ainda em 2014 a GoogleNet foi vencedora do *ImageNet Large Scale Visual Recognition Challenge* – Desafio de Reconhecimento Visual em Grande Escala ImageNet –, se destacando por mostrar ganhos na performance de classificação e consequentemente, ganhos de desempenho em tarefas de visão computacional, por conta das melhoras arquitetônicas (Szegedy *et al.*, 2015).

Anos depois, em 2017, a Google apostou na publicação de outra CNN, a MobileNet, desenvolvida pelos engenheiros da própria empresa Andrew G. Howard e Menglong Zhu. Segundo (Rodrigues, 2020) a MobileNet possui a proposta de ser uma Rede Neural profunda e leve para aplicações móveis e embarcadas de visão computacional, com eficiência no quesito de uso de recursos computacionais e consumo de energia.

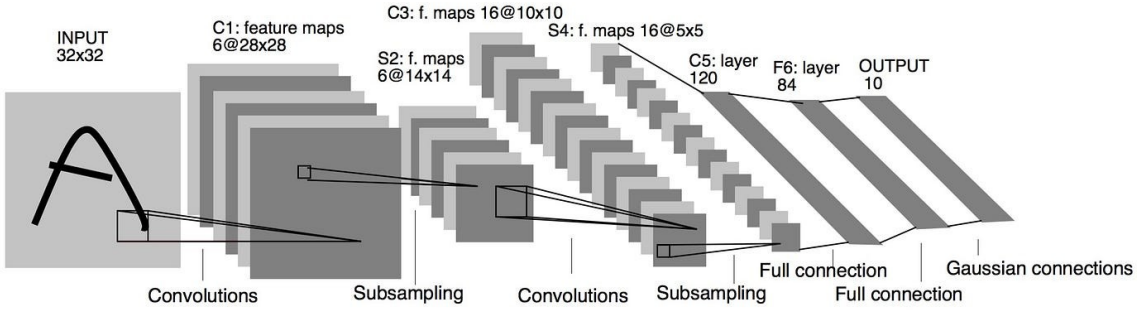
Conforme dito anteriormente, uma imagem passa por uma série de camadas dentro da rede e retorna uma saída, que pode ser uma divisão de probabilidades. No fim das contas, o arranjo dessas camadas e componentes têm um papel fundamental na criação de diferentes arquiteturas de rede, cada uma com suas vantagens e desvantagens (Khan et al., 2020).

A arquitetura básica de uma CNN é composta por camadas alternadas de convolução e *pooling*, acompanhado de uma ou mais camadas totalmente conectadas (figura 1).

2.2.1. Camada de convolução

É na camada de convolução que os atributos da imagem de entrada são extraídos através de operações de convoluções de matrizes. Uma matriz menor (*kernel*) serve como filtro, onde será lido todos os

Figura 1. Camadas de uma Rede Neural Convolucional de classificação.



Fonte: (Filho; Rosa; Guimarães., 2021)

pixels para produzir uma matriz de dimensões menores que a matriz de entrada (Filho *et al.*, 2021).

O cálculo da matriz a ser convolucionada é feita com a soma da multiplicação entre primeiro valor da matriz *kernel* com o primeiro valor da matriz maior, pelo segundo valor da matriz *kernel* multiplicado com segundo valor da matriz maior, e assim sucessivamente, como podemos observar na **Figura 2 e 3**.

Há dois parâmetros importantes nessa etapa: A distância a ser percorrida entre um campo receptivo e outro é chamada de *Stride*, que possui valor variável e maior que 0 (zero) e o Padding. O Padding mostra como os filtros atuam em entradas com tamanho não múltiplo (em especial nas bordas da matriz): aplicando a convolução apenas onde o filtro consiga sobrepor pixels válidos ou completando com valores nulos nos espaços faltantes (Remus, 2023). Ou seja, a combinação dessas variáveis permite controlar o tamanho do mapa de ativação que será gerado.

Segundo (Goodfellow; Bengio; Courville, 2016), Um mapa de ativação é gerado para cada filtro usado na camada de convolução, e representa as regiões onde as características do filtro foram encontradas.

Algo muito importante de ressaltar é que os pesos dos filtros são reutilizados em toda a imagem, chamado de compartilhamento de parâmetros. A partir disso permite assegurar um número muito menor de parâmetros únicos, e, consequentemente, aumenta o tamanho da rede sem que seja necessário aumentar a quantidade de dados de treinamento, o que é uma importante propriedade das Redes Neurais Convolucionais.

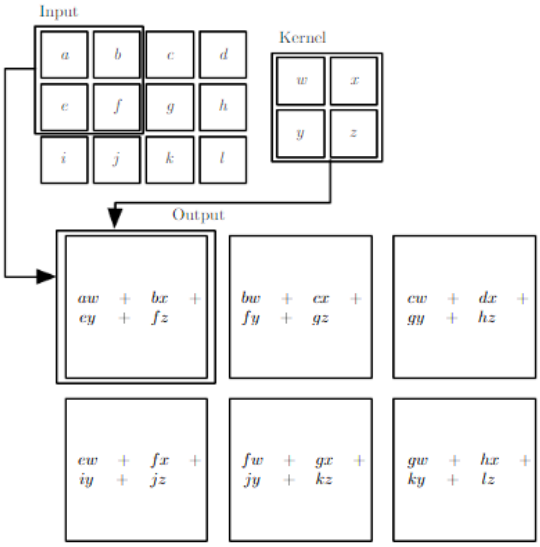
2.2.2. Camada não linear

Após as camadas de convolução, é convenção aplicar uma camada não-linear (ou camada de ativação) logo em seguida. O propósito desta camada é introduzir não-linearidade a um sistema que basicamente só tem computado operações lineares na camada de convolução (multiplicações e somatórios) (Goodfellow; Bengio; Courville, 2016).

Introduzir não-linearidade a um modelo, torna-o capaz de criar associações mais complexas entre entradas e saídas da rede, o queé essencial para aprendizado de dados complexos, como imagens (Goodfellow; Bengio; Courville, 2016).

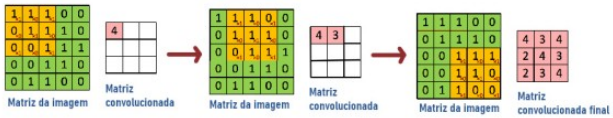
Nesta camada é aplicada a função de ativação Unidade Linear Retificada (*Rectified Linear Unit* - ReLU), com o propósito de deixar a convolução sem números negativos (Nair, Hinton, 2010).

Figura 2. Exemplo de convolução onde uma matriz 3x4 se tornará uma matriz 2x3.



Fonte: (Goodfellow; Bengio; Courville, 2016)

Figura 3. Exemplo de construção de uma matriz convolucional a partir de uma matriz imagem.



Fonte: (Cunha, 2020)

A saída da função ReLU é: f(x) = max(0,x), desta forma os numeros positivos se manterão e os negativos se tornam zero. Em outros termos, essa camada aumenta as propriedades não-lineares do modelo sem afetar as ativações da camada de convolução como exemplificado na figura 4.

2.2.3. Camada de *Pooling*

No terceiro estágio, usamos uma função pooling para modificar ainda mais a saída da camada. Na camada de pooling, a ideia é reduzir drasticamente o número de parâmetros, para que haja a condensação do tamanho espacial da representação da imagem. O processo, que pode ser chamado de *downsampling*, permite acelerar os cálculos computacionais da rede (Ranzato *et al*., 2007).

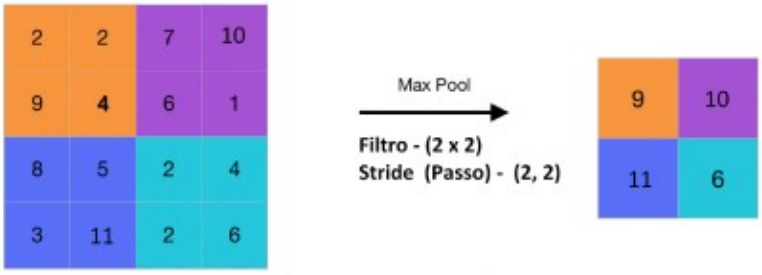
Seu funcionamento é parecido com a convolução: é definida uma área e um stride, mas desta vez, ao invés de extrair informações, o objetivo é comprimir as informações já extraídas.

O Max Pooling, o mais popular, funciona com base na área definida e no stride definido, desta forma ele utiliza o maior valor para aplicar na nova matriz e fazer o redimensionamento, que torna-a mais leve sem perder informações importantes, como pode ser observado na Figura 5.

Figura 4. Exemplo de aplicação da função de ativação ReLU em um mapa de ativação.

Fonte: Elaborada pelos autores

Figura 5. Exemplo de Max Pooling com o área = (2x2) e Stride = 2.



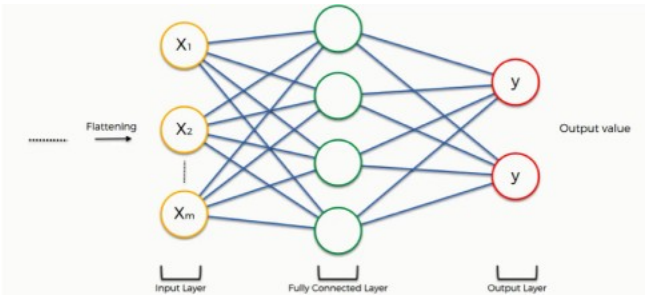
Fonte: (Filho *et al.*, 2021)

2.2.4. Camada totalmente conectada

A saída da camada anterior é responsável por achatar a matriz em um vetor que irá alimentar a camada totalmente conectada, que cumpre o papel de retornar uma decisão quanto a classificação da imagem como um todo.

A camada totalmente conectada irá interligar todas as características e atributos extraídos da imagem através de uma estrutura neuronal tradicional de camadas ocultas, entretanto, onde todos os neurônios da camada estão interconectados (Cunha, 2020) Figura 6.

Figura 6. Exemplo de uma camada totalmente conectada



Fonte: (Filho *et al.*, 2021)

A partir da análise das saídas da camada anterior, a camada totalmente conectada determina quais classes mais se correlacionam com determinadas características. Por exemplo, se a rede está

classificando uma imagem como tartaruga, ela certamente obteve altos valores nos mapas de ativação que representam características de casco (Deshpande, 2016).

2.3. Uso de Redes Neurais Convolucionais em projetos ambientais

As Redes Neurais convolucionais no uso de detecção e classificação de objetos são usadas em diversos meios, para obter os melhores resultados possíveis de acordo com o propósito do projeto. Mas utilizar as CNN's em projetos ambientais é algo necessário que tem crescido exponencialmente ao longo dos anos.

Em (Gray *et al*., 2019), tinham como objetivo a criação de uma CNN robusta para detecção de vida selvagem, capaz de funcionar bem em condições variáveis, a fim de evitar a imposição de restrições adicionais aos voos de drones.

Noutra proposta feita por (Dujon *et al*., 2021), eles tiraram fotos e vídeos por drone de três espécies: gaivotas, tartarugas marinhas e leões-marinhos; sobre uma colônia terrestre na Ilha Kanowna, Austrália e acima do mar no local de reprodução da ilha de Zakynthos, na Grécia. Aplicaram um aumento de dados para terem mais amostras e depois testaram o modelo, encontrando desafios como sombras, rochas e até reflexo de objetos na água. Chegando até a mapear a trajetória de tartarugas marinhas em vídeo.

No estudo de (Oliveira *et al*., 2019), três algoritmos de *machine learning* e *Convolutional Neural Network* (CNN) foram testados para a classificação de imagens da frota *SkySat* da *Planet* com alta resolução espacial visando à identificação de áreas queimadas, na tentativa de mapear incêndios florestais que podem provocar danos ecológicos, econômicos, sociais e à saúde.

Dito isso, podemos refletir a importância e a factibilidade do uso de CNN em projetos ambientais. O atual experimento tem o foco maior em auxiliar na classificação binária de tartarugas marinhas.

3. Metodologia

A abordagem utilizada neste trabalho foi a de pesquisa exploratória, que consistiu em investigar os resultados obtidos com o treinamento, teste e validação das classificações de tartarugas marinhas utilizando as arquiteturas VGG, ResNet, MobileNet e DenseNet.

A pesquisa exploratória tem como principal objetivo o aprimoramento de idéias ou a descoberta de intuições ou formulação de novas idéias. As pesquisas exploratórias são extremamente flexíveis, de modo que quaisquer aspectos relativos ao fato estudado têm importância (Oliveira; Ponte; Barbosa., 2006).

3.1. Base de dados utilizada

O conjunto de imagens usado para treinar os modelos é uma base de dados própria, criada pelos discentes que executaram este projeto.

Essa base foi criada, principalmente, através de *frames* de vídeos de tartarugas nascendo e indo em direção ao mar, disponibilizados pela Prefeitura do município do Paulista aos discentes do IFPE Campus Paulista. Além desse meio, também foram buscadas manualmente imagens de não-tartarugas para equilibrar a quantidade de imagens entre as classes. É importante destacar que não foi localizada nenhuma outra base apropriada para o trabalho em questão, tratando especificamente de filhotes de tartarugas. O uso de imagens de tartarugas adultas, nesse caso, poderia criar ruído de identificação.

Neste contexto, o termo ’ru ído de identificac¸ão’ refere-se à interfer ência ou confus ão que poderia surgir ao incluir imagens de tartarugas adultas na base de dados. Isso pode desafiar a capacidade do modelo de distinguir com precis ão entre filhotes e adultos, uma vez que as caracter ísticas e aparências entre as duas fases podem ser distintas. Isso resultaria em um possível desempenho comprometido do modelo, pois ele poderia confundir as caracter ísticas dos filhotes com as dos adultos, diminuindo a precisão na identificação específica de filhotes de tartarugas.

Esta base cont ém fotos de tartarugas na areia da praia, ao eclodir dos ovos, indo em direc¸ ão ao mar, que são distribuídas de forma aleatória em relação às espécies existentes no litoral brasileiro, com a finalidade de melhorar a precis ão do modelo, j á que algumas esp écies de tartarugas marinhas apresentam características diferentes, como, por exemplo, o formato ou a colorac¸ão do casco. Além disso, contém também imagens de não-tartarugas, que são elementos como conchas, pedras e alguns animais da fauna costeira, que podem ser encontrados na faixa de areia da praia. O conjunto de treino original possui 414 imagens. Foi utilizada a ferramenta Roboflow (Dwyer; Nelson; Solawetz., 2022) para auxiliar no armazenamento e manipulac¸ ão de imagens, tal como *data augmentation*. Após o processo de *data augmentation* o conjunto de treinamento totalizou 615 amostras.

Para o data augmentation, foram utilizadas as operações de *Flip*, que consiste em virar/refletir aleatoriamente a imagem verticalmente ou horizontalmente; *90 degree rotation*, que gira aleatoriamente a imagem em 90 ou 180 graus, podendo ser no sentido hor ário, anti-horário e de cabec¸a para baixo; *Exposure*, que ajusta a exposição da imagem para ficar mais clara ou mais escura; e por fim o *Blur*, que introduz o desfoque gaussiano na imagem.

Durante o treinamento dos modelos as imagens foram divididas entre tr ês pastas onde em cada uma delas havia duas pastas nomeadas ’c0’ representando n ão-tartarugas e ’c1’ representando as tartarugas. Os modelos consumiam as imagens dessas pastas durante o treinamento, validação e teste.

Vale ressaltar que, neste momento, a quantidade exposta para o modelo é de 539 imagens, pois filtramos as imagens mais relevantes [Figura 7, Figura 8, Figura 9, Figura 10], para melhor assertividade. A divisão dos conjuntos de treinamento, validac¸ão e teste foram de 363 amostras, 144 amostras e 62 amostras, respectivamente.

Figura 7. Exemplo de imagens do conjunto de dados da classe n ão-tartaruga

Fonte: Elaborada pelos autores

Figura 8. Exemplo de imagens do conjunto de dados da classe não-tartaruga

Fonte: Elaborada pelos autores

Figura 9. Exemplo de imagens do conjunto de dados da classe tartaruga

Fonte: Elaborada pelos autores

Figura 10. Exemplo de imagens do conjunto de dados da classe tartaruga

Fonte: Elaborada pelos autores

3.2. Detalhes sobre a avaliação realizada

Como principais bibliotecas, optou-se por utilizar Tensorflow juntamente com o Keras (Abadi *et al.*, 2015) e o Scikit-learn (Pedregosa *et al.*, 2011).

O TensorFlow é uma biblioteca flexível, amplamente reconhecida por sua versatilidade na expressão de diversos algoritmos, como na inferência para modelos de redes neurais profundas. Sua popularidade decorre do amplo uso para conduc¸ão de pesquisas e para implementac¸ão de sistemas de Aprendizagem de Máquina (AM) e Visão Computacional. Por sua vez, o Keras, como uma API de alto nível do TensorFlow, tem a capacidade de criar e treinar modelos de aprendizado profundo, simplificando etapas como construção, treinamento e avaliação dos modelos.

A escolha dessas bibliotecas foi motivada, em parte, pela capacidade do Keras em suportar modelos pré-treinados. O que viabiliza a utilizac¸ão de redes neurais pré-treinadas em grandes conjuntos de dados, como a base Imagenet, permitindo sua adaptac¸ão para tarefas específicas de Visão Computacional. Em outras palavras, é possível realizar *transfer learning*, um aspecto crucial para nosso escopo de estudo.

Já o Scikit-learn é uma biblioteca de código aberto em Python, reconhecida por oferecer recursos eficazes em Aprendizagem de Máquina. Sua integração com outras bibliotecas Python, como NumPy, SciPy e Matplotlib, é uma grande vantagem. Optamos por sua utilização por sua capacidade de aplicabilidade na etapa de pré-processamento e na de visualização dos dados, além de oferecer um conjunto de ferramentas robustas para avaliação das métricas de classificação dos modelos.

Ao selecionar as arquiteturas para este estudo, a eficiência computacional foi um critério essencial. Nesse sentido, a arquitetura ResNet se destacou, pois possui estruturas residuais que reduzem a carga computacional em comparação com outras arquiteturas profundas (He *et al.*, 2016).

A arquitetura MobileNet, por sua vez, foi selecionada pela sua capacidade de oferecer soluções embarcadas, tornando-a adequada para sistemas com recursos limitados. Isso é possível pois a MobileNet utiliza operac¸ões de convolução profundas com filtros de tamanho reduzido, conhecidas como convoluções *depthwise separable*(Howard *et al.* , 2017), esse tipo de convolução visa reduzir drasticamente o número de parâmetros e a carga computacional, tornando a rede mais eficiente em termos de recursos, especialmente em dispositivos móveis e sistemas embarcados. Além disso, o fato desta arquitetura, por padrão, utilizar o dataset ImageNet, que é composto por mais de 1.500.000 imagens em 1.000 categorias, e a disponibilizac¸ão do código-fonte pelo Google com 16 pontos de verificação, permitiram o treinamento de modelos personalizados reutilizando a estrutura da MobileNet. Ademais, o modelo MobileNet é o primeiro modelo de visão de computador móvel do TensorFlow (Pujara, 2020) *apud* (Rosas, 2021).

A arquitetura DenseNet foi uma escolha baseada em sua capacidade de aprimorar a eficiência de treinamento e a precisão na classificação de imagens. Ao permitir que cada camada tenha acesso direto às informações de todas as camadas anteriores, o DenseNet facilita o fluxo de gradientes durante o treinamento, ajudando a evitar o problema de desvanecimento de gradientes (Huang *et al.*, 2017). Adicionalmente, sua abordagem para mitigar o *overfitting*, problema comum em redes profundas, promovendo uma maior troca de informações entre as camadas, melhorando a representação das características das imagens.

Exploramos também a arquitetura VGG, cuja adequac¸ão para soluções embarcadas depende da versão e do tamanho da rede utilizada. Embora a VGG original com 19 camadas seja bastante profunda e tenha uma grande quantidade de parâmetros, existem versões mais leves e compactas,

tornando-as viáveis em dispositivos com recursos limitados, como dispositivos móveis. A VGG se destacou em competic¸ões importantes, como o desafio de classificac¸ão de imagens ImageNet em 2014, onde alcanc¸ou uma alta precisão de classificação de objetos. (Howard *et al.* , 2017)

Quanto à otimização de hiperparâmetros, inicialmente consideramos o método *grid search*, que é um algoritmo de busca que recebe um conjunto de valores de um ou mais hiperparâmetros e testa todas as combinações dentro dessa vizinhanc¸a PEDREGOSA et al., 2011 *apud* (Hammerschmitt *et al.*, 2022).

No entanto, apesar da simplicidade na implementac¸ão por não requerer um conhecimento probabilístico mais avanc¸ado, a busca exaustiva por combinac¸ões dentro da vizinhanc¸a tornou-se inviável devido aos recursos computacionais limitados disponíveis para esta pesquisa.

Posteriormente, realizamos um estudo com *Bayesian Search*, um algoritmo de busca que tenta estimar qual é a combinação de hiperparâmetros que resultará na maior performance, com base numa distribuição criada a partir das combinações testadas anteriormente KUMAR et al., 2017 *apud* (Hammerschmitt *et al.*, 2022). O *Bayesian Search* se mostrou mais viável em comparação com o *Grid Search* quando se trata de recursos computacionais limitados. Sua capacidade de convergir para soluções aceitáveis com menos experimentos, juntamente com sua adaptac¸ão contínua ao longo das iterações, permitiu uma melhoria progressiva na busca por melhores hiperparâmetros.

3.3. Métricas utilizadas na avaliação

As métricas de avaliação desempenham um papel crucial na análise da performance dos modelos de Aprendizado de Máquina (AM). Embora sua função primária seja medir a capacidade do modelo em generalizar o aprendizado quando confrontado com dados desconhecidos, é importante destacar que as métricas oferecem uma visão multifacetada da eficácia do modelo. Ou seja, cada métrica pode revelar ou abordar diferentes características ou aspectos do desempenho do modelo.

No presente trabalho temos como classes: tartaruga ou não-tartaruga. Portanto, o problema se enquadra entre os problemas de classificac¸ão binárias. Em um problema de classificação binária, de acordo com Ferrari e Silva (2017), existe uma classe alvo, também chamada de classe positiva - a classe cujo valor se deseja predizer. No trabalho em questão, a classe alvo é a classe c0 que representa as tartarugas. Entretanto, com a existência da classe positiva, existe também classe negativa.

A partir desse raciocínio, as classificac¸ões binárias, em suas predic¸ões podem ter quatro possíveis tipos:

- VP (Verdadeiro Positivo): objeto da classe positiva classificado como positivo – por exemplo, uma tartaruga classificada como tartaruga.
- VN (Verdadeiro Negativo): objeto da classe negativa classificado como negativo –por exemplo, um objeto não-tartaruga classificado como não-tartaruga.
- FP (Falso Positivo): objeto da classe negativa classificado como positivo – por exemplo, um objeto não-tartaruga classificado como tartaruga.
- FN (Falso Negativo): objeto da classe positiva classificado como negativo – por exemplo, um objeto tartaruga classificado como não-tartaruga.

No escopo deste trabalho, a avaliac¸ão do desempenho dos modelos de classificac¸ão será realizada por meio de métricas criteriosas, buscando uma análise abrangente e detalhada. As métricas selecionadas para essa avaliação incluem a acurácia, que oferece a relação entre os acertos do modelo e o total de padrões avaliados. Além disso, para uma compreensão mais completa, serão utilizadas

métricas como o F1 Score [Figura 12], que considera tanto a precisão (número de amostras previstas como positivo e corretas), quanto o recall (número de amostras previstas como verdadeiro positivo e corretas em relação a todas as amostras daquela classe), fornecendo uma medida combinada de ambos. Para uma análise detalhada da distribuição das predições, também será empregada a matriz de confusão [Figura 11], permitindo visualizar de forma clara e concisa os acertos e erros do modelo em relação a cada classe.

Figura 11. Um exemplo de matriz de confusão, uma das métricas de classificação binária.

Fonte: Elaborada pelos autores

Figura 12. Exemplo de fórmula *F1 Score*.

$$F1\ Score = 2 * \frac{Precisão * Recall}{Precisão + Recall}$$

Fonte: Elaborada pelos autores

4. Experimentos e Resultados

Para o desenvolvimento, utilizou-se o Google Colaboratory, ou Colab. O Colab é um serviço em nuvem gratuito hospedado pela empresa Google sem necessidade de qualquer configuração, para ter acesso é necessário apenas conter um endereço eletrônico da empresa. Acessado via navegador web, é uma ótima ferramenta para fins didáticos e documentação de conhecimento, pois contém uma série de células iterativas que podem conter textos explicativos ou códigos executáveis e suas respectivas saídas armazenadas (Mello, 2022).

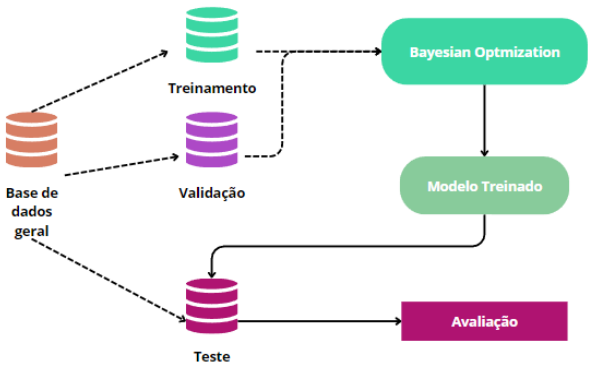
Cada arquivo com células iterativas é chamado de notebook. O grande diferencial do serviço é que os notebooks do Colab são automaticamente armazenados na conta do Google Drive, sendo possível compartilhar os notebooks com outras pessoas, permitindo que elas façam comentários ou até editem o documento. Isso possibilita a difusão e evolução do conhecimento para o desenvolvimento da área de Aprendizado de Máquina e Inteligência Artificial (Google, 2023).

Inicialmente o treinamento foi realizado 3 vezes para cada arquitetura, para que fosse encontrado um modelo que obtivesse um melhor resultado em relações às métricas. Em todas as arquiteturas foi utilizado o algoritmo ADAM, que é uma otimização do gradiente estocástico que combina ideias do algoritmo RMSprop e do momento de gradiente. Ele é capaz de adaptar a taxa de aprendizado de forma individual para cada parâmetro da rede neural (Kingma; Ba; 2014).

Como as arquiteturas utilizadas neste trabalho são preparadas para o uso de transferência de conhecimento, somente foi necessário alguns ajustes nos hiperparâmetros, como os pesos do treinamento na base Imagenet e a remoção da camada final responsável pela classificação, para a inclusão de uma nova camada densamente conectada ajustada à base de dados do problema em questão.

Todos os modelos foram treinados entre 10 e 50 épocas e entre 10 e 80 neurônios ocultos, para avaliar qual a melhor configuração para cada modelo e evitar, principalmente, o *overfitting* e *underfitting*. Durante o processo, os modelos foram treinados e, em seguida, avaliados como podemos ver no fluxo da figura 13. Logo após foram selecionados os modelos com os melhores valores das métricas de cada iteração, para compará-los entre si e verificar qual modelo se saiu melhor.

Figura 13. Fluxo de treinamento do modelo.



Após os resultados, foi percebida a necessidade de melhorias de otimização dos hiperparâmetros devido a percepção de *overfitting* de alguns modelos, representados na Tabela 1. Diante desse cenário, a abordagem adotada para aprimorar a eficiência do treinamento envolveu a aplicação da técnica de Bayesian Optimization (Otimização Bayesiana) no conjunto de treinamento. Para este fim, estabelecemos de maneira específica a função objetivo para essa otimização, bem como os inter-

valos de hiperparâmetros utilizados como espac¸o de busca.

Por meio da Otimizac¸ão Bayesiana, buscamos encontrar os melhores conjuntos de hiperparâmetros para otimizar o desempenho do modelo. A definic¸ão clara da func¸ão objetivo e a delimitação precisa dos intervalos de hiperpar âmetros permitiram direcionar eficientemente a busca por configuracões ótimas, promovendo um treinamento mais eficaz e, potencialmente, mitigando problemas como o overfitting ao ajustar melhor os parâmetros do modelo aos dados disponíveis.

Tabela 1. Resultados - Utilizando otimizador ADAM

Arquitetura	n. de épocas	Neurônios	acurácia treino	acurácia teste
DenseNet	50	20	1,0	0,70
ResNet	50	20	0,87	0,73
MobileNet	50	20	0,83	0,11
VGG16	50	20	0,7	0,7

Após a implementac¸ão da Otimizac¸ão Bayesiana, os modelos foram treinados com o intervalo de 10 a 50 épocas e de 10 a 80 neur ônios escondidos em uma camada, onde o foram feitas 10 rodadas para cada um, com a finalidade de encontrar os valores mais adequados para os par âmetros dentro desses limites. Estes parâmetros foram escolhidos dentro do limite computacional disponível, tornando viável a investigacão e permitindo a obtencão de resultados mais robustos.

Ao final de cada execucão, um arquivo com os pesos e os modelos foi salvo, o que possibilitou que os modelos fossem depois restaurados para a avaliacão do desempenho, e com isso podemos obter os seguintes resultados:

Tabela 2. Resultados - Bayesian Optimization para MobileNet

Iteracão	n. de épocas	Neurônios	acurácia teste	treino f1 score	teste f1 score
1	25	22	0,95	0,952	0,931
2	37	39	**0,96**	**0,980**	**0,966**
3	46	71	0,98	0,980	0,931
4	44	69	0,96	0,986	0,912
5	22	74	0,96	0,980	0,931
6	45	70	0,95	0,980	0,949
7	26	17	0,96	0,973	0,912
8	26	60	0,95	0,952	0,949
9	48	39	0,967	0,973	0,872
10	48	18	0,93	0,986	0,931

Tabela 3. Resultados - Bayesian Optimization para ResNet

Iteração	n. de épocas	Neurônios	acurácia teste	treino f1 score	teste f1 score
1	13	20	0,5	0,696	0,666
2	38	39	0,95	0,993	0,931
3	32	39	0,98	0,973	0,931
4	45	71	**0,96**	**0,993**	**0,966**
5	43	58	0,98	0,993	0,830
6	50	53	0,98	0,993	0,912
7	31	68	0,96	0,993	0,949
8	26	60	0,95	0,993	0,949
9	42	19	0,96	0,993	0,966
10	48	18	0,98	0,987	0,966

Tabela 4. Resultados - Bayesian Optimization para DenseNet

Iteração	n. de épocas	Neurônios	acurácia teste	treino f1 score	teste f1 score
1	23	23	0,98	0,993	0,966
2	37	39	0,96	0,993	0,949
3	37	31	**0,98**	**0,987**	**0,983**
4	40	11	0,98	1,0	0,983
5	41	29	0,98	0,987	0,949
6	20	24	1,00	0,993	0,949
7	26	17	0,98	1,0	0,983
8	37	21	0,98	1,0	0,983
9	23	62	0,98	1,0	0,966
10	33	41	0,98	0,993	0,983

Tabela 5. Resultados - Bayesian Optimization para VGG16

Iteração	n. de épocas	Neurônios	acurácia teste	treino f1 score	teste f1 score
1	13	20	0,95	0,980	0,912
2	37	39	0,96	0,986	0,931
3	48	66	0,98	0,980	0,931
4	25	52	**0,98**	**0,980**	**0,973**
5	41	29	0,93	0,986	0,931
6	20	33	0,96	0,986	0,966
7	36	23	0,98	0,993	0,931
8	42	19	0,96	0,993	0,966
9	34	56	0,96	0,974	0,931
10	43	25	0,98	0,986	0,912

Ao aplicar o método de Bayesian Optimization na arquitetura MobileNet, pode-se observar, na Tabela 2, variações consideráveis no desempenho do modelo conforme os diferentes conjuntos de hiperparâmetros e iterações foram avaliados. Por exemplo, em diversas iterações, alcançou-se uma acurácia de teste em torno de 95% a 98%, acompanhada por pontuações consistentes no treino e teste para o F1 Score, indicando uma boa capacidade do modelo de generalizar o aprendizado para novos dados. No entanto, certas combinações de parâmetros resultaram em discrepâncias entre os resultados

Tabela 6. média de resultados utilizando Bayesian Optimization para cada modelo

Modelo	treino f1 score	teste f1 score
ResNet	0,994	0907
DenseNet	0,993	0,970
MobileNet	0,974	0,977
VGG16	0,976	0,951

de treino e teste, sugerindo um possível overfitting ou falta de generalização em certos casos.

Já na Tabela 3 os resultados da aplicação do Bayesian Optimization na arquitetura ResNet revelaram uma tendência consistente de desempenho superior em comparação com a MobileNet. Em várias iterações, alcançamos altos níveis de acurácia de teste, oscilando entre 95% e 98%. Além disso, observamos um equilíbrio relativamente estável entre as pontuações de treino e teste para o F1 Score, indicando uma capacidade confiável do modelo de generalizar para novos dados.

No contexto da arquitetura DenseNet, na Tabela 4, os resultados da otimização Bayesian sugerem um desempenho consistente e robusto. Observa-se uma acurácia de teste geralmente alta, variando entre 96% e 100% em diferentes iterações. Os valores de F1 Score para treino e teste mantiveram-se elevados, indicando uma boa capacidade de generalização do modelo. Esses resultados sugerem uma tendência de desempenho sólido e confiável da DenseNet sob diferentes configurações de hiperparâmetros.

Ao otimizar a arquitetura VGG16 utilizando o Bayesian Optimization, identificamos, na Tabela 5, um desempenho geralmente sólido, embora com algumas variações. A acurácia de teste variou entre 93% e 98%, demonstrando uma performance considerável, mas com algumas oscilações. Os valores do F1 Score para treino e teste permaneceram relativamente altos, sugerindo uma capacidade razoável de generalização do modelo, apesar de algumas variações nos resultados de diferentes iterações.

Com base na análise comparativa dos resultados médios obtidos através da aplicação do Bayesian Optimization em diferentes modelos de redes neurais - ResNet, DenseNet, MobileNet e VGG16 - contidos na Tabela 6, é possível traçar algumas conclusões significativas.

Os resultados médios indicam que a ResNet e a DenseNet se destacam como as arquiteturas com desempenhos mais robustos e consistentes, enquanto a MobileNet e a VGG16 demonstram bons resultados, mas com possíveis desafios em relação à capacidade de generalização para novos dados. Essas conclusões ressaltam a importância de selecionar adequadamente a arquitetura do modelo de acordo com a natureza e complexidade dos dados a serem processados.

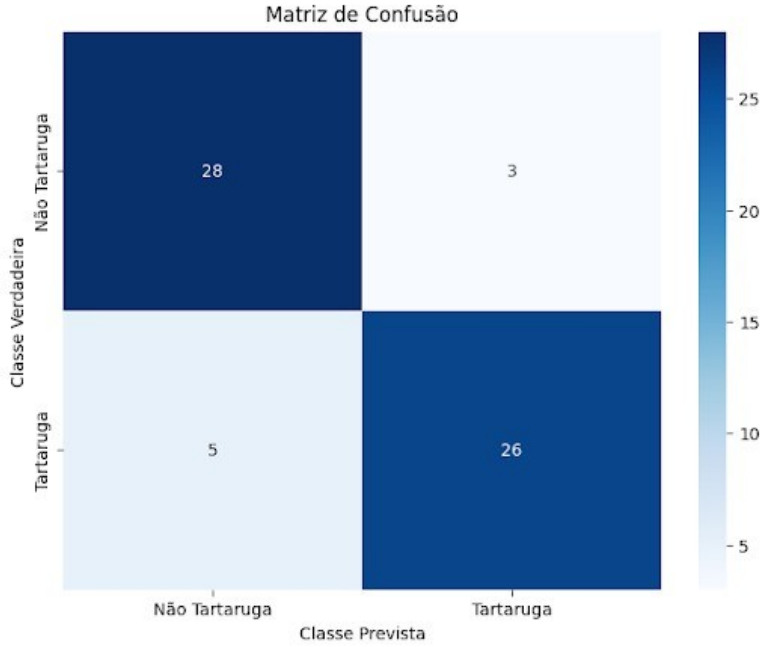
Na Figura 14 pode-se contemplar o resultado obtido pelo modelo VGG16 através de uma matriz de confusão, onde 28 das 31 amostras de não-tartaruga e 26 das 31 amostras de tartaruga foram classificadas de maneira correta, respectivamente.

Através da análise da Figura 12, é evidente um exemplo de precisão na predição realizada pelo modelo ResNet. Nesta instância específica, a classe c0, que representava corretamente um não-tartaruga, foi prevista com precisão, enquanto a classe c1, correspondente à categoria de tartaruga, foi igualmente identificada com acurácia.

Já na Figura 15, observa-se uma predição realizada pelo modelo DenseNet. Neste contexto, a classe c0, representando a categoria de não-tartaruga, e a classe c1, associada à categoria de tartaruga,

foram incorretamente atribuídas a amostras com maior ”similaridade”, como conchas com areia da praia e tartarugas cobertas de areia da praia. Essa imprecisão na classificação indica uma dificuldade do modelo em distinguir entre elementos que possuem características visuais altamente similares, resultando em erros de predição para casos específicos.

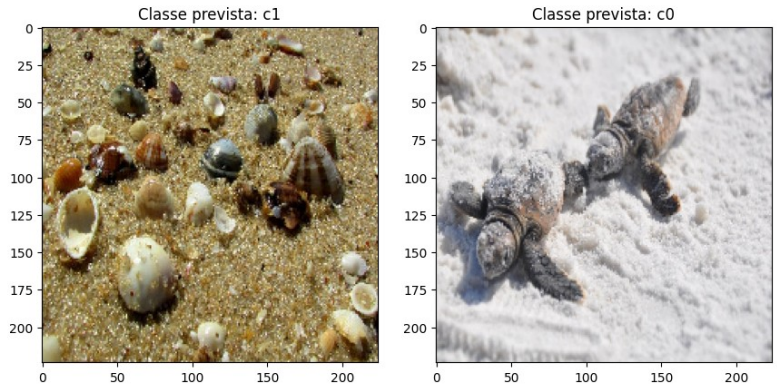
Figura 14. Matriz de confusão obtida utilizando modelo VGG16 treinado com configurações obtidas pelo Bayesian Optimization.



Fonte: Elaborada pelos autores

Figura 15. predição feita pelo ResNet.

Figura 16. predicão feita pelo DenseNet.



As arquiteturas apresentaram resultados promissores; no entanto, torna-se evidente que a utilizacão de uma base de dados mais robusta poderia aprimorar as métricas de treinamento da aplicacão, aumentando a confianca nos resultados desses modelos. A transferência de aprendizado surge como um elemento crucial a ser abordado nesta secão. De acordo com (Albuquerque, 2019), um sistema de classificacão treinado em um conjunto de dados de um domínio diferente do problema em questão pode ser refinado e modificado para se ajustar mais adequadamente à nova área de aplicacão. Dessa forma, ao incorporar essas consideracões, é possível vislumbrar melhorias substanciais na eficácia e adaptabilidade dos modelos em questão.

5. Conclusão e Trabalhos Futuros

Na conducão deste estudo, inicialmente foram realizadas experimentacões manuais, consistindo na alteracão dos valores referentes às épocas e neurônios ocultos, a fim de otimizar o desempenho dos modelos utilizados. Essa abordagem permitiu uma compreensão aprofundada do comportamento do sistema diante de diferentes configuracões, oferecendo insights valiosos.

Reconhecendo a necessidade de explorar de maneira mais abrangente o espaco de hiperparâmetros, optamos pela aplicacão de Grid Search. Essa técnica sistemática proporcionou uma varredura mais ampla, permitindo-nos identificar combinacões ótimas de parâmetros para aprimorar as métricas de desempenho dos modelos.

Diante da complexidade computacional envolvida e buscando otimizar ainda mais a eficiência do processo de busca de hiperparâmetros, decidimos incorporar o Bayesian Optimization. Essa abordagem baseada em modelos probabilísticos mostrou-se eficaz em guiar a pesquisa de maneira inteligente, reduzindo o número de experimentos necessários para alcancar resultados satisfatórios.

Contudo, é crucial destacar que, ao longo do desenvolvimento desta pesquisa, nos deparamos com limitacões computacionais significativas. O volume de recursos necessários para explorar de maneira abrangente o espaco de hiperparâmetros, especialmente ao empregar técnicas avancadas como o Bayesian Optimization, exigiu uma capacidade computacional substancial.

Como perspectiva para trabalhos futuros, é imperativo reavaliar estratégias computacionais, considerando alternativas para superar as limitacões encontradas. A busca por métodos de otimizacão mais eficientes e a exploracão de abordagens paralelas podem ser exploradas para enfrentar os desafios computacionais e proporcionar resultados mais promissores. Essa reconsideracão estratégica é essencial para maximizar a utilidade e a generalizacão dos modelos propostos, contribuindo assim para avancos significativos no campo de estudo em questão.

Além disso, outra perspectiva interessante é a aplicação do modelo em diferentes contextos e cenários. Por exemplo, o modelo poderia ser adaptado para realizar a contagem de outras espécies de animais ou mesmo para identificar características específicas, como a presença de animais feridos ou em risco.

É válido considerar a ampliação da base de dados utilizada para treinamento, incluindo uma maior diversidade de imagens e uma quantidade significativa de amostras. Isso permitiria que o modelo aprendesse de forma mais abrangente as características das tartarugas filhotes, melhorando sua capacidade de generalização e precisão.

Por fim, também é importante destacar a importância da colaboração com instituições e especialistas na área de conservação e pesquisa de tartarugas marinhas. O compartilhamento de conhecimentos e a troca de informações podem enriquecer o projeto e permitir uma abordagem mais completa e abrangente em relação ao monitoramento e preservação dessas espécies.

Referências

ALBUQUERQUE, L. D. S. An álise de técnicas de transfer learning aplicadas ao problema de covariate shift. 2019.

BAPTISTOTTE, C. Tartarugas marinhas: Projeto tamar. 1992.

BOMFIM, A. d. C. et al. Monitoramento de longo prazo de ninhos de tartarugas marinhas no nordeste do brasil. Biota Neotropica, SciELO Brasil, v. 21, 2021.

CUNHA, L. C. d. Redes neurais convolucionais e segmentac¸ão de imagens: uma revisão bibliográfica. 2020.

DESHPANDE, A. A Beginer's Guide to Understanding Convolutional Neural Networks. [S.l.], 2016. Disponível em: ⟨https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/⟩.

DUJON, A. M. et al. Machine learning to detect marine animals in uav imagery: Effect of morphology, spacing, behaviour and habitat. Remote Sensing in Ecology and Conservation, Wiley Online Library, v. 7, n. 3, p. 341–354, 2021.

DWYER, B.; NELSON, J.; SOLAWETZ, J. 2022. Disponível em: ⟨https://roboflow.com/⟩.

FILHO, A. C. M. et al. Redes neurais convolucionais para classificação e avaliação da maturação de frutos de café. Universidade Federal de Uberlândia, 2021.

FILHO, L. R. A.; ROSA, R. R.; GUIMARÃES, L. N. Intelligent supernovae classification systems in the kdust context. Anais da Academia Brasileira de Ci ências, Academia Brasileira de Ciências, v. 93, p. e20200862, 2021. ISSN 0001-3765. Dispon ível em: ⟨https://doi.org/10.1590/0001-3765202120200862⟩.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep Learning. [S.l.]: MIT Press, 2016. ⟨http://www.deeplearningbook.org⟩.

GOOGLE. Welcome To Colaboratory. [S.l.], 2023. Disponível em: ⟨https://colab.research.google.com/notebooks/intro.ipynb⟩.

GRAY, P. C. et al. A convolutional neural network for detecting sea turtles in drone imagery. Methods in Ecology and Evolution, Wiley Online Library, v. 10, n. 3, p. 345–355, 2019.

HAMMERSCHMITT, B. K. et al. Identificação de perdas n ão técnicas através de m étodo de classificação e otimização de hiperparâmetros, baseado em dados endógenos e exógenos. Simpósio Brasileiro de Sistemas Elétricos-SBSE, v. 2, n. 1, 2022.

HE, K. et al. Identity mappings in deep residual networks. In: SPRINGER. Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14. [S.l.], 2016. p. 630–645.

HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

JUNIOR, F. A. C.; BORGES-ANDRADE, J. E. Uso do conceito de aprendizagem em estudos relacionados ao trabalho e organizações. Paidéia (Ribeirão Preto), SciELO Brasil, v. 18, p. 221–234, 2008.

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

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. Neural computation, MIT Press, v. 1, n. 4, p. 541–551, 1989.

MARCOVALDI, M. A.; MARCOVALDI, G. G. dei. Marine turtles of brazil: the history and structure of projeto tamar-ibama. Elsevier, p. 35–41, 1999.

MELLO, A. C. Detecc¸ão de desbalanceamento de massa no rotor de turbinas e ólicas utilizando algoritmos de aprendizado profundo. Universidade Federal de Santa Maria, 2022.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on International Conference on Machine Learning. Madison, WI, USA: Omnipress, 2010. (ICML'10), p. 807–814. ISBN 9781605589077.

OLIVEIRA, M. C.; PONTE, V. M. R.; BARBOSA, J. V. B. Metodologias de pesquisa adotadas nos estudos sobre balanced scorecard. In: Anais do Congresso Brasileiro de Custos-ABC. [S.l.: s.n.], 2006.

OLIVEIRA, P. D. S. d. Uso de aprendizagem de máquina e redes neurais convolucionais profundas para a classificação de áreas queimadas em imagens de alta resolução espacial. 2019.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, v. 12, p. 2825–2830, 2011.

RANZATO, M. et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition. [S.l.: s.n.], 2007. p. 1–8.

REMUS, J. C. Uso de redes neurais de convoluc¸ão para classificac¸ão de anomalias em arames metálicos a partir de imagens de inspeção por partícula magnética. 2023.

RODRIGUES, G. A. Reconhecimento de emoc¸ões utilizando redes neurais convolucionais para auxiliar no tratamento de crianc¸as com autismo. 2020.

ROSAS, C. L. G. Análise da morfometria dos agregados do solo utilizando imagens digitais por meio de redes neurais. Universidade Estadual Paulista (Unesp), 2021.

RUBACK, L.; AVILA, S.; CANTERO, L. Vieses no aprendizado de máquina e suas implicac¸oes sociais: Um estudo de caso no reconhecimento facial. In: SBC. Anais do II Workshop sobre as Implicações da Computação na Sociedade. [S.l.], 2021. p. 90–101.

SZEGEDY, C. et al. Rethinking the Inception Architecture for Computer Vision. 2015.