Artigo recebido em 21/11/2024
Aceito em 24/11/2024

Revista SODEBRAS – Volume 20
N° 223 – JANEIRO/ ABRIL – 2025

AVALIAÇÃO DE DESEMPENHO ENTRE MODELOS NEURAIS CONVOLUCIONAIS PARA CLASSIFICAÇÃO DO OXFORD-IIIT PET DATASET

PERFORMANCE EVALUATION OF CONVOLUTIONAL NEURAL MODELS FOR CLASSIFYING THE OXFORD-IIIT PET DATASET

Héctor Dorrighello Giacon[1]
Rodrigo Henrique Cunha Palácios[1]
Márcio Mendonça[2]
Mariana Gasparotto Palácios[3]
André Roberto Ortoncelli[1]

Resumo – Este estudo investigou o desempenho de redes neurais convolucionais na classificação de imagens para categorizar uma base de dados em duas classes: gato e cachorro. Com diversas abordagens computacionais disponíveis para a comunidade, optou-se por treinar três modelos — MobileNet, ResNet e EfficientNet — utilizando essa base de imagens. Os modelos foram então submetidos a uma avaliação de desempenho final, comparando as probabilidades e os gradientes sobrepostos às imagens, a fim de determinar qual modelo se mostrou mais eficiente. Este artigo aborda conceitos de visão computacional e ilustra como diferentes modelos de redes neurais convolucionais processam os dados, oferecendo um guia para novos estudantes interessados na área de ciência de dados. Facilitando a compreensão dos principais objetivos e contribuições desta investigação cientifica. E se encerra com uma conclusão e endereça futuros trabalhos.

Palavras-chave: Visão Computacional, Classificação, Convolução, Desempenho.

Abstract – This study investigated the performance of image classification to categorize the database into cats or dogs by training convolutional neural networks since several computational approaches are implemented for use by the community. In this way, three models were chosen, MobileNet, ResNet, and EfficientNet, to be trained from the image database to undergo a final performance evaluation between the probabilities and gradient superimposed on the image to determine which model was the most efficient. In this way, this article presents concepts with computer vision. It illustrates how different models of convolution-based neural networks act on data, showing a path to new students in the data community. It ends with a conclusion and addresses future works.

Keywords: Computer Vision, Classification, Convolution, Performance.

[1] Programa de Pós Graduação em Informática (PPGI), Universidade Tecnológica Federal do Paraná, Cornélio Procópio (UTFPR-CP). Contato: hectordorrighello@gmail.com, rodrigopalacios@utfpr.edu.br e ortoncelli@utfpr.edu.br.
[2] Programa de Pós-Graduação em Engenharia Mecânica (PPGEM), Universidade Tecnológica Federal do Paraná, Cornélio Procópio (UTFPR-CP). Contato: mendonca@utfpr.edu.br.
[3] Bacharelado em Engenharia de Software, Universidade Tecnológica Federal do Paraná, Cornélio Procópio (UTFPR-CP). Contato: marianagp294@gmail.com.

I. INTRODUÇÃO

É evidente que a inteligência artificial está cada vez mais presente no cotidiano das pessoas e das empresas em geral, principalmente por meio de *chatbots* (DUTT; SASUBILLI; YERRAPATI; 2020). Entretanto, o uso de modelos de I.A. não está limitado somente a este tipo de aplicação (SICILIANO; KHATIB, 2008).

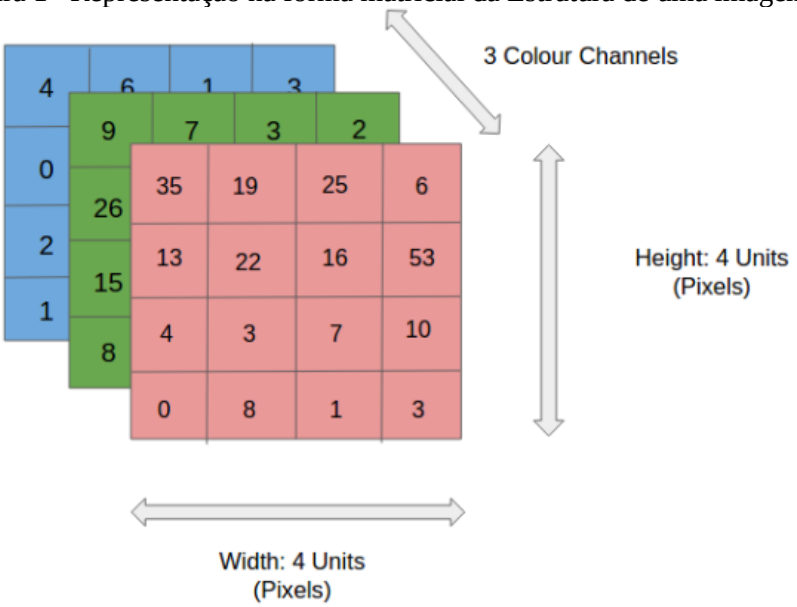
A inteligência artificial, em especial sistemas computacionais inteligentes podem ser classificados em quatro grandes áreas: Aprendizado de Máquina (HAYKIN, 2009), Lógica Fuzzy (PASSINO; YOURKOVICH, 1997), sistemas evolutivos e agentes inteligentes, como por exemplo robótica em grupo (MENDONÇA, *et al*, 2019).

Dentre as várias áreas presentes na própria área de dados, uma que deve ser destacada é a visão computacional. Processar e analisar imagens a ponto de realizar classificações ou mesmo identificar elementos faz deste campo um dos mais estudados e requisitados atualmente. Como exemplo de aplicações na área de visão computacional pode-se citar análise de imagens de satélite e/ou drones sobre plantações agrícolas ou mesmo identificar se lotes de grãos estão conforme os padrões de qualidade. Ademais, áreas como a medicina também tem se beneficiado com o uso de visão computacional. De acordo com Silva, I. R. R. *et al* (2018), foi proposta a utilização de redes neurais convolucionais (do inglês, *convolutional neural networks*, CNN) para a classificação e diagnóstico da doença de Alzheimer, visto que essas redes possuem uma significativa capacidade de aprendizagem em bases de imagens utilizadas para aprendizado.

Como processar imagens tende a ser algo custoso, este processo tende a tornar um gargalo para modelos treinados utilizando figuras, dependendo do tamanho, resolução, qualidade e quantidade, visto que quanto melhor a resolução e quantidade de imagens, mais poder de processamento é necessário. Para isso, uma alternativa utilizada atualmente são as Unidades de Processamento Gráfico, também chamadas de GPUs, no qual pesquisadores que já utilizaram esse tipo de hardware conseguiram otimizar o desempenho de seus projetos em até 300 vezes HWU (WEN-MEI W., 2011). Ressalta-se que o tempo é um recurso mais significativo. Posto isso, propor e desenvolver modelos capazes de serem treinados em tempo hábil e apresentarem uma taxa de aprendizado satisfatória motiva o surgimento de novos métodos estatísticos e computacionais cada vez mais robustos. Outro ponto a ser considerado são os dados em si, seja de imagens ou não, pois dados de baixa qualidade acarretarão treinos de modelos com baixa qualidade, resultando em análises e previsões imprecisas e incorretas. Portanto, vale ressaltar que um resultado de qualidade depende de maneira significativa de dados de qualidade como alimentador do modelo.

As CNNs conseguem, a partir de uma imagem de entrada, atribuir um peso ou importância e diferenciar uma figura de outras. Como os *pixels* de uma imagem são representados nas células de uma matriz e cada canal de cor possui a sua matriz, como mostra a Figura 1, deve-se considerar que o processamento pode se tornar custoso utilizando métodos convencionais. Para melhorar esse gargalo, uma rede neural convolucional pode aplicar filtros e transformações sobre uma imagem a ponto de reduzir e capturar aspectos para realizar uma boa classificação em tempo hábil.

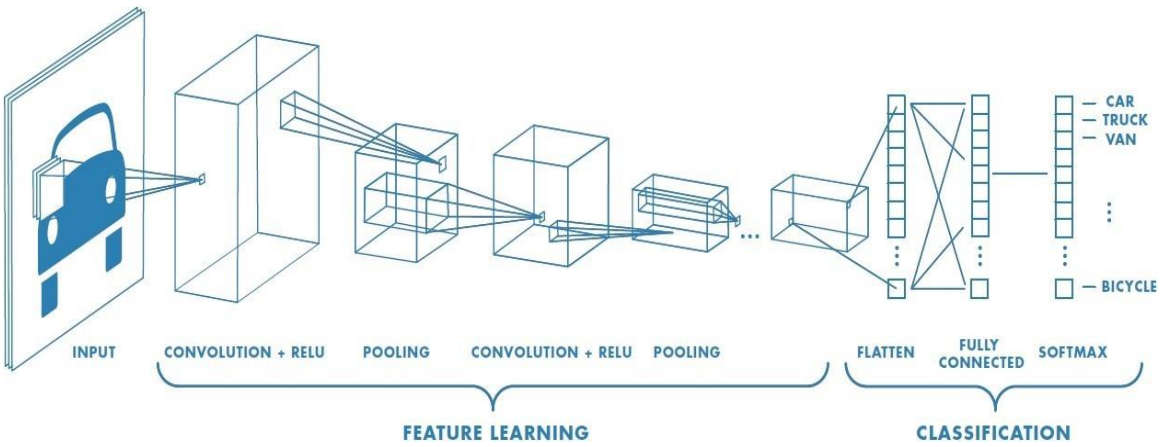
Figura 1 –Representação na forma matricial da Estrutura de uma imagem. .



Fonte: DATA SCIENCE ACADEMY, 2024.

Internamente uma rede neural convolucional possui algumas camadas principais, como apresentado na Figura 2. A primeira a ser destacada é a de entrada, cuja função é fornecer os dados para a rede com o formato desejado. Em seguida tem-se a camada convolucional, no qual é incumbida de extrair características das imagens, bem como detectar os padrões presentes nelas. Após a convolução, é necessária uma camada que aplique uma função de ativação, como a *Rectified Linear Activation* ou *ReLU*, por exemplo, a fim de reduzir a não linearidade para que assim a rede possa aprender mais relações presentes nos dados. A próxima camada, chamada de *pooling*, visa reduzir a dimensionalidade das características obtidas nas camadas anteriores a fim de otimizar o desempenho e tempo de execução da rede de maneira geral. A penúltima camada a ser abordada é a densa. Esta por sua vez tem os seus neurônios conectados a todos os outros, chamada de camada totalmente conectada. Nela os dados recebidos da camada de *pooling* são transformados para poderem ser classificados adequadamente. E, por fim, a última camada corresponde a saída do modelo.

Figura 2 – Estrutura das camadas de uma CNN para classificação de veículos.



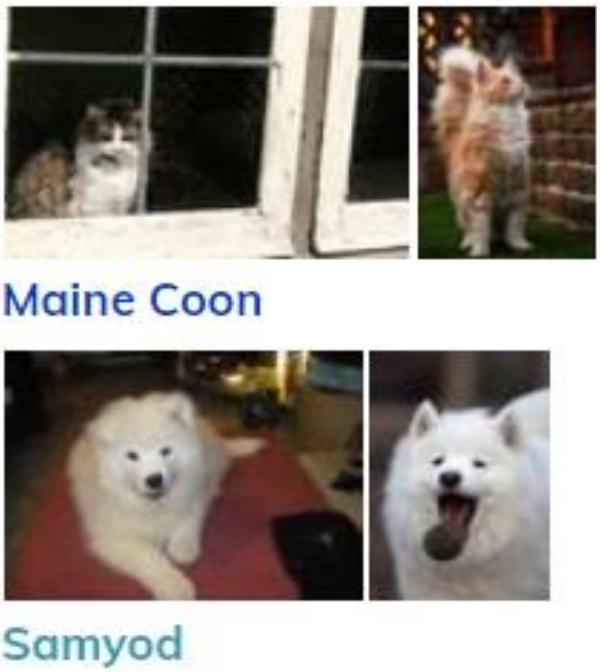
Fonte: DATA SCIENCE ACADEMY, 2024.

Em função disso, o objetivo deste artigo é apresentar uma análise de três modelos de redes neurais baseadas em convolução, de modo a estudar o seu funcionamento e desempenho em classificar imagens de animais, mais especificamente gatos e cachorros. Para isso, são investigados os modelos *MobileNet, EfficientNet* e *ResNet*. Após a implementação e testes, são mostrados os resultados dos modelos treinados com as probabilidades e imagens a fim de chegar a uma conclusão comparativa. Dessa forma, o artigo está estruturado da seguinte forma: Na Seção II é apresentada a metodologia aplicada. Na Seção III é mostrada a solução aplicada para o problema em questão, analisando os dados e modelos escolhidos em termos de métricas e aspectos gráficos visuais. E, por fim, na Seção IV as conclusões do projeto são expressas.

II. METODOLOGIA

Este trabalho tem como objetivo classificar as imagens da base de dados Oxford-IIIT Pet (PARKHI, *et al*, 2012), sob a licença *Creative Commons Attribution-ShareAlike 4.0 International*, em gato ou cachorro, avaliando o desempenho de cada modelo em relação ao treino, validação e teste. Na Figura 3 é apresentado como são definidas as amostras para cada tipo. Como a base de dados possui gatos e cachorros de raças diferentes, esta classificação em específico não será feita, mas pode ser realizada como uma continuação deste trabalho, integrando mais modelos para realizar sub-classificações.

Figura 3 – Amostra da base de dados.



Fonte: DATA SCIENCE ACADEMY, 2024.

Outro ponto a ser destacado é o fato de um modelo de classificação de imagens necessitar não só de figuras, mas de dados que definam cada imagem, como o seu caminho no diretório da máquina (*path*), o seu nome e a qual classe pertence. Esses dados podem estar presentes com a base de imagens, como um arquivo de tabela ou podem ser gerados em tempo de execução. Neste caso, essas informações foram extraídas a partir do nome do arquivo de cada imagem presente na base de dados.

Dessa forma, uma base de dados precisou ser gerada para descrever o que cada imagem representa, bem como o seu caminho (*path*), nome de raça e espécie, no qual representa a classe da imagem. Para extrair tais dados foi utilizado o nome dos arquivos, seguindo o seguinte padrão: *nome_da_raça_id.jpg*, e nisso percebeu-se que raças de gato começavam com a letra maiúscula e assim foram definidas a qual classe cada imagem pertencia. O nome da raça foi extraído do nome dos arquivos utilizando método de separação por caracteres (*split*).

A partir dos dados extraídos e representados juntos como uma tabela, foi necessário analisar como estava a sua distribuição em relação ao número de raças e o número de classes (espécie). Para isso, foram analisadas a quantidade de raças de cada espécie, a quantidade de imagens de cada espécie e a quantidade de imagens de cada raça das duas espécies, ou seja, a distribuição de frequência das raças de gato e cachorro.

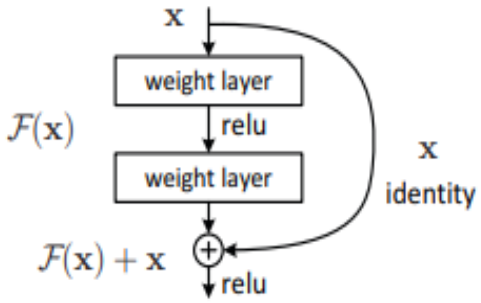
Dessa forma, para um total de 7390 imagens, na classe gato há 12 raças com 2400 imagens no total e com 200 imagens para cada raça. Para a classe cachorro há 25 raças, com 4990 imagens no total e com uma média de 200 imagens por raça. Como há mais dados de cachorro do que gato, uma simples divisão dos dados baseado em porcentagem tende a não proporcionar resultados efetivos, uma vez que algum tipo de viés para a classe dominante em quantidade pode ocorrer. Com a finalidade de evitar desbalanceamento dos dados, a proporcionalidade em relação à raça foi levada em consideração. Para isso, as bases de treino, teste e validação possuem em média a mesma quantidade de imagens em relação às raças de gato e cachorro, fazendo com que o fato da quantidade de raças não seja um fator relevante.

A partir dos dados devidamente separados, necessitou-se escolher modelos para realizar o treino e a classificação das imagens. Os modelos escolhidos para serem

comparados nesta tarefa foram: *ResNet50*, *EfficientNetB3* e *MobileNetV2*. Todas são construções feitas a partir de redes neurais convolucionais, ou seja, são semelhantes, mas diferem em sua arquitetura e finalidade, mas também são eficientes em dados em grade.

O primeiro modelo a ser apresentado é *ResNet50*, no qual foi proposto por He *et al*. (2016). Possui como ideia central o conceito de conexões residuais, ou seja, ligações que saltam entre uma ou mais camadas, uma vez que a rede tenta aprender a diferença entre a camada de entrada e saída em vez de fazer esse mapeamento diretamente em cada camada. Isso ajuda a reduzir o problema de desempenho, uma vez que redes profundas possuem muitas camadas e acarretam diminuição do gradiente no treinamento. Arquiteturalmente, estas conexões são feitas pelos blocos residuais, os quais são estruturas compostas por duas ou mais camadas convolucionais, como apresentado na Figura 4.

Figura 4 – Representação de um bloco residual.



Fonte: DATA SCIENCE ACADEMY, 2024**.**

O modelo *EfficientNetB3* foi proposto por Tan e Le (2019) e consiste em utilizar um conceito chamado escalonamento composto. Essa abordagem tem como ideia central ajustar simultaneamente a profundidade, largura e resolução da rede, causando uma maior adaptabilidade para diferentes tamanhos de entrada. Este modelo é da família *EfficientNet*, que consiste em modelos que prezam manter um equilíbrio entre acurácia, tempo de treinamento e resolução, ou seja, podem apresentar resultados melhores em tempo de treinamento hábil. A fim de evitar o sobre-treinamento (*overfitting*), o modelo utiliza camadas de *Batch Normalization*, que consiste na normalização das ativações das camadas da rede, acelerando o processo de treino, e *Dropout*. A nomenclatura B3 indica o tamanho do modelo, ou seja, modelos maiores possuem mais parâmetros, por consequência possuem um desempenho melhor e podem ser úteis em tarefas mais complexas, mas são computacionalmente mais complexos e exigem mais recursos computacionais.

E, por fim, o modelo *MobileNetV2* foi proposto por Sandler *et al*. (2018). Este modelo tem como finalidade a aplicação em ambientes com recursos limitados, como smartphones ou mesmo dispositivos embarcados, uma vez que visa ser eficiente em termos de custo computacional e memória, bem como sendo uma rede neural menor e com menos parâmetros. Como o foco é eficiência, este modelo utiliza um conceito chamado Convolução Separável. Outra estrutura utilizada em sua arquitetura são os blocos residuais que, assim como os modelos da família *ResNet*, facilitam o uso de mais camadas e assim aprofundando a rede na totalidade. Uma implementação de otimização deste modelo é a técnica *Bottleneck*, no qual reduz a quantidade de cálculos em algumas camadas da rede, minimizando a dimensionalidade dos dados conforme a rede for se aprofundando.

III. RESULTADOS

O treinamento dos modelos escolhidos tende a ser custoso computacionalmente, visto que utilizam dados de imagem. Dessa forma, são necessárias técnicas para otimizar o processo de treinamento, como por exemplo o *Transfer-learning*, na qual aproveita o conhecimento adquirido por um modelo já treinado e, no caso deste projeto, a *ImageNet*, fazendo com que os pesos e conexões aprendidos durante o treino sejam aproveitados pelos modelos escolhidos. Outro ponto a ser destacado é como as redes foram definidas após a instanciação dos modelos, pois ao invés de adicionar as camadas utilizando *Sequential*, utilizou-se a técnica *Functional*. Essa escolha foi necessária para se ter mais controle sobre qual informação está sendo passada entre as camadas, bem como poder ter acesso direto a cada uma das camadas de maneira direta.

A partir dos dados trabalhados e do modelo escolhido, ficaram definidas as constantes para a aplicação dos modelos, portanto, para o tamanho padrão da imagem foi escolhido o valor 300, para o tamanho do lote (*batch*) foi escolhido o valor 8 e para o número de épocas o valor 10.

A partir destes conceitos estabelecidos, as Tabelas 1 e 2 mostram quais foram os hiperparâmetros escolhidos para a criação do *ImageDataGenerator* para os dados de treino. Essa estrutura do framework *TensorFlow* tem como finalidade preparar dados de imagem para modelos convolucionais, fornecendo atributos e métodos úteis para estes casos, como transformações nas imagens, separação por *batch* ou mesmo o tipo de classe a ser predita. Como apresentado na Tabela 1, há um hiperparâmetro a ser destacado: *preprocessing_function*. Essa função prepara os dados conforme o modelo atual, dessa forma, cada um dos 3 modelos possui o seu *preprocessing_function* correspondente. Outro ponto a ser destacado sobre esta tabela é o fato de adicionar *augmentation* sobre a base de treino. Essa técnica consiste em ampliar a quantidade de dados tendo como base informações já existentes, aplicando filtros e transformações sobre elas, para assim aumentar a variabilidade dos dados de treino em relação aos outros.

Tabela 1 – Hiperparâmetros para o *ImageDataGenerator* de treino.

Hiperparâmetro	**Valor**
rotation_range	90
width_shift_range	0.2
height_shift_range	0.2
shear_range	0.2
zoom_range	0.2
horizontal_flip	*True*
fill_mode	*“nearest”*
preprocessing_function	<depende do modelo>

Vale ressaltar que as tabelas correspondentes de teste e validação para a Tabela 1 possuem apenas o parâmetro *preprocessing_function* e, por isso, não foram apresentadas. Na Tabela 2 vale para as 3 bases de dados, uma vez que utilizam os mesmos hiperparâmetros**.**

Tabela 2 – Hiperparâmetros para *flow_from_dataframe*.

Hiperparâmetro	**Valor**
dataframe	*train, test or valid*
x_col	*‘image_path’*

y_col	*'specie'*
target_size	(300, 300)
batch_size	8
class_mode	*'categorical'*

Na Tabela 3 são apresentados os hiper parâmetros correspondentes à instanciação de um modelo. Neste caso dois são destacados: *weights* e *model.trainable*. O primeiro consiste em utilizar os pesos da *ImageNet*, ou seja, utilizar os pesos de uma rede pré-treinada com uma densa base de dados para de otimizar o treino do modelo atual. Já para *model.trainable*, como neste projeto está sendo utilizado o conceito de *Transfer-learning*, definir este parâmetro como *False* indica que os pesos do modelo não serão atualizados durante o treino, uma vez que os pesos da *ImageNet* foram escolhidos.

Tabela 3 – Definição do modelo.

Hiperparâmetro	**Valor**
weights	*'imagenet'*
include_top	*False*
input_tensor	*Input (shape = (*300, 300, 3*))*
model.trainable	*False*

Por outro lado, na Tabela 4 estão definidos os *callbacks* utilizados durante os treinos. Um *callback* é uma função que pode ser executada em alguns pontos durante o treinamento de um modelo, como fim de uma época de treinamento, monitoramento de métrica ou mesmo visualização de camadas intermediárias. No caso deste projeto foram escolhidos dois: *EarlyStopping* e *ReduceLROnPlateau*. O primeiro consiste em evitar o sobre ajuste, ou seja, o modelo se ajusta demais aos dados de treinamento e acaba perdendo a sua capacidade de generalização, resultando em classificações imprecisas. Outro ponto sobre este *callback* é a restauração do melhor modelo, ou seja, quando o treinamento é interrompido. Neste caso é restaurado o modelo no ponto em que a métrica monitorada se apresentou melhor, evitando treino e gasto computacional desnecessário. *ReduceLROnPlateau* é utilizado para reduzir a taxa de aprendizagem do modelo quando a métrica monitorada para de melhorar, ou seja, chegou a um ponto máximo (*Plateau*). Neste caso, este *callback* ajusta a taxa de aprendizagem durante o treino no momento de estagnação para que a precisão e generalização do modelo não sejam prejudicadas.

Tabela 4 – Definição dos *callbacks*.

Hiperparâmetro
EarlyStopping(patience = 4, *min_delta* = 0.01*)*
ReduceLROnPlateau(monitor = 'val_acc', factor = 0.2, *patience* = 4*)*

Na Tabela 5 são apresentados os hiperparâmetros utilizados para a compilação do modelo antes do início do treinamento. Destacam-se dois deles: *loss* e *metrics*. O primeiro, *loss*, refere-se à função de perda, utilizada pelo otimizador para avaliar o quão bem o modelo está performando durante o treinamento. Para este projeto, foi escolhida a função *categorical cross-entropy*, comumente empregada em problemas de classificação onde os dados são rotulados de forma categórica e cada amostra pertence a apenas uma classe, como "gato" (*Cat*) ou "cachorro" (*Dog*).

O segundo hiperparâmetro, *metrics*, corresponde às métricas utilizadas para monitorar o desempenho do modelo durante o processo de treinamento. Neste projeto, a métrica selecionada foi a acurácia (*acc*), que representa a proporção de classificações corretas realizadas pelo modelo em relação ao total de predições.

Tabela 5 – Hiper parâmetros de compilação.

Hiperparâmetro	**Valor**
loss	*"categorical_crossentropy"*
optimizer	*"adam"*
metrics	*["acc"]*

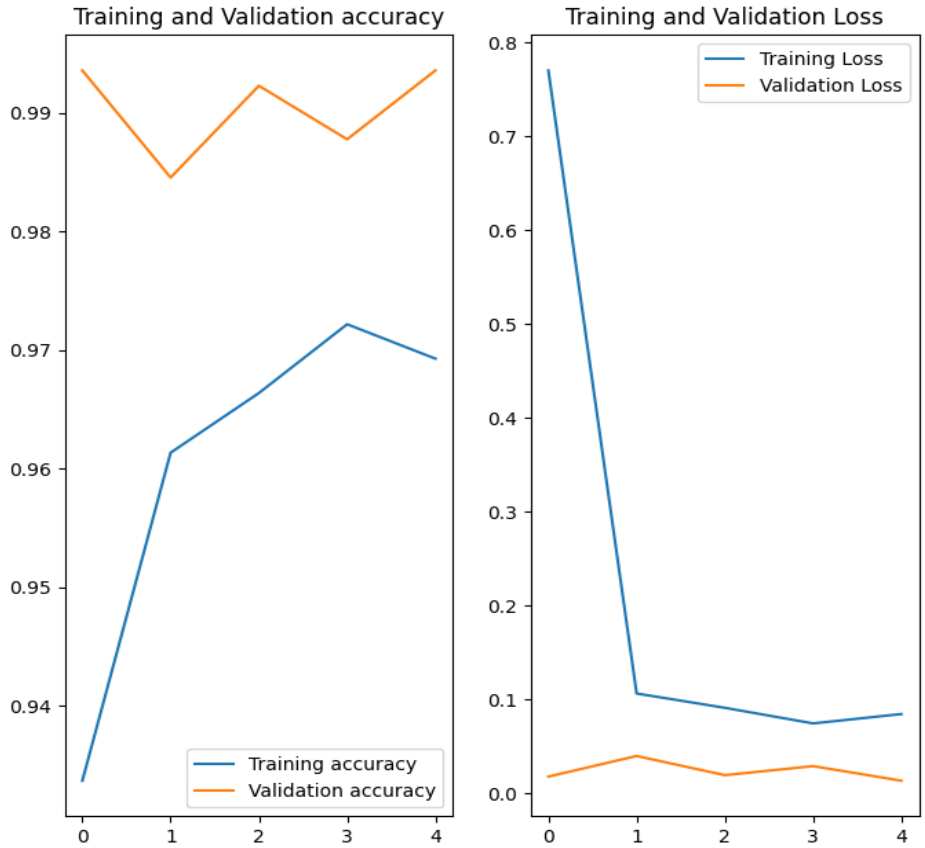
3.1 – Análises dos Resultados

Nesta seção, são apresentados e analisados os resultados dos três modelos - *ResNet*, *EfficientNet* e *MobileNet* - incluindo imagens, acurácia geral e a probabilidade resultantes de cada classificação de imagem. Para cada modelo testado, estão disponíveis o gráfico de desempenho do treinamento e as imagens classificadas, exibidas utilizando o *GRAD-CAM* (ALAM, M. U.; BALDVINSSON, J. R.; WANG, Y., 2022), juntamente com as probabilidades de a imagem pertencer à classe "gato" ou "cachorro". As imagens originais e as saídas divididas por *argmax* e *argmin* encontram-se no repositório do projeto, acessível em (YU, J.; CAO, J.; HE, R., 2022). É importante mencionar que o repositório contém um *Dockerfile* para a criação do ambiente virtual a partir do arquivo *requirements.txt*, além de um arquivo *README.md* para auxílio. Um último detalhe sobre a pasta do projeto é que as figuras escolhidas para testar e avaliar o gradiente são de uso livre, ou seja, não possuem direitos autorais.

Abordando mais especificamente o conceito do *GRAD-CAM*, este consiste em verificar como as camadas de convolução são ativadas, permitindo avaliar visualmente se o modelo aprendeu as características corretas. Em outras palavras, a ideia é extrair um mapa de ativação das *features* na última camada de convolução. Por esse motivo, foi necessário utilizar a técnica *Functional* para as camadas dos modelos neste projeto, a fim de identificar onde na imagem os modelos aprenderam as características. Para efeito visual, o mapa de ativação é representado por um mapa de calor (*heatmap*) e sobreposto à imagem de teste, destacando as características consideradas para a classificação pelos modelos. Essa abordagem também pode ser útil para evidenciar falsas taxas de acerto, já que as imagens podem conter marcas d'água e elementos que podem dificultar o trabalho de classificação.

Em relação aos modelos, o primeiro a ser analisado é *ResNet50*, no qual foi possível obter a segunda maior acurácia para a base de teste com o valor de 0.9969. De uma maneira geral, os resultados se mostraram promissores. A Figura 5 mostra a relação entre acurácia e taxa de aprendizagem para as bases de treino e validação no qual, para este caso, a acurácia do treinamento se mostrou inferior. Algumas possibilidades para este caso é o fato de utilizar a camada de *Dropout*, uma vez que sendo ativada somente na etapa de treino, pode causar este fenômeno. Outro ponto para este caso é o fato que a base de treino ser superior em número de dados, ou seja, a sua representatividade perante os dados é maior.

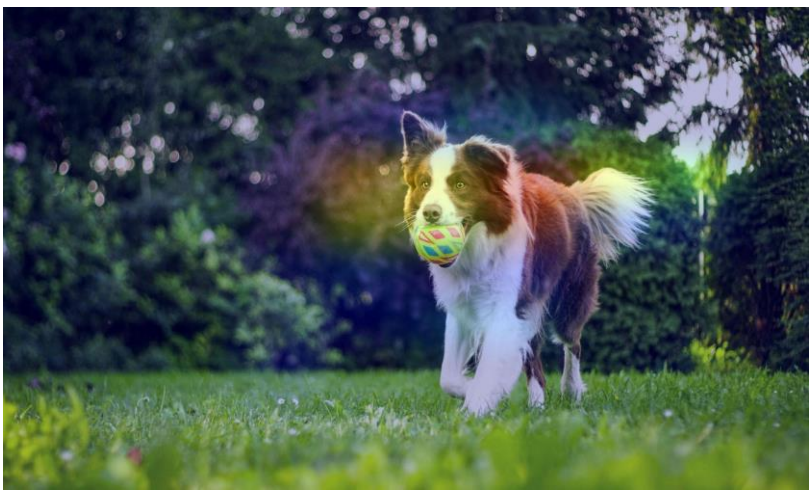
Figura 5 – Acurácia e taxa de aprendizado para *ResNet50* em relação a treino e validação.



Fonte: O próprio autor, 2024.

Na Figura 6, representando um cachorro, a classe predita foi *Dog* com probabilidade de 99.9995% e ser da classe *Cat* com 0.0005%, correspondendo corretamente à classe da imagem. O fato de haver uma bola na boca do cachorro não foi um impedimento para a correta classificação.

Figura 6 – Saída do modelo *ResNet50* para imagem com cachorro.



Fonte: O próprio autor, 2024.

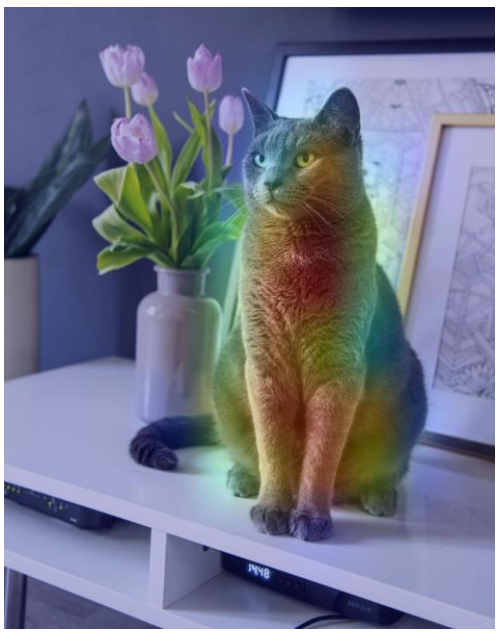
Para a Figura 7, representando dois cachorros, a classe predita foi *Dog* com probabilidade de 99.9999% e ser da classe *Cat* com 0.0001%, correspondendo corretamente à classe da imagem. Um destaque para o gradiente, mostrado por meio do *heatmap* sobre a figura, é o fato de estar destacado sobre o cachorro à direita. Neste caso, uma possibilidade é o fato deste cachorro ocupar um tamanho relativamente maior em relação ao outro na imagem.

Figura 7 – Saída do modelo *ResNet50* para imagem com dois cachorros.

Fonte: O próprio autor, 2024.

A Figura 8, representando um gato, a classe predita foi *Cat* com probabilidade de 100% e probabilidade de ser da classe *Dog* de 4.0676e07%, ou seja, muito próximo de 0%, correspondendo corretamente à classe da imagem. Vale destacar que o gradiente mostrado na figura se mostrou em cores mais intensas principalmente sobre o corpo do gato, mostrando o porquê a probabilidade ter sido máxima.

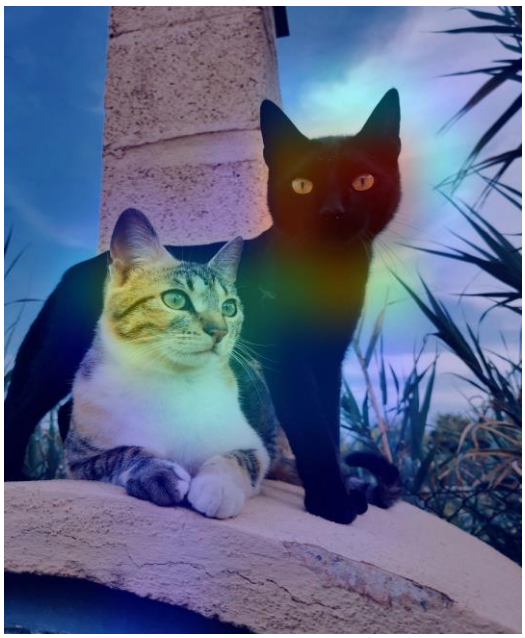
Figura 8 – Saída do modelo ResNet50 para imagem com gato.



Fonte: O próprio autor, 2024.

Por outro lado, na Figura 9 representando dois gatos, a classe predita foi *Cat* com probabilidade de 100% e probabilidade de ser da classe *Dog* de 3.5885e-06%, ou seja, muito próximo de 0%, correspondendo corretamente à classe da imagem. Neste caso, o gradiente mostrado pelo *heatmap* não foi muito intenso, provavelmente por haver dois gatos.

Figura 9 – Saída do modelo *ResNet50* para imagem com dois gatos.



Fonte: O próprio autor, 2024.

Complementarmente, na Figura 10 é apresentado o processamento de uma imagem de gato juntamente com cachorro. A classe predita foi *Cat* com probabilidade de 99.9529% e probabilidade de ser da classe *Dog* de 0.0470%. Para este modelo, não houve indecisão para este caso, como mostra o gradiente sobre a imagem, no qual foi intenso sobre o gato e ignorou o cachorro. Isso indica um ponto de melhoria para este modelo, uma vez que a *ResNet* se ajustou mais para a classe gato, diferentemente da *EfficientNet* e *MobileNet*, como são mostrados nos testes seguintes.

Figura 10 – Saída do modelo *ResNet50* para imagem com um gato e um cachorro.

Fonte: O próprio autor, 2024.

Em relação aos testes com o modelo *EfficientNetB3*, obteve-se os resultados mais promissores neste trabalho. Na Figura 11 é apresentada uma avaliação para treinamento e validação, no qual a acurácia se mostrou superior aos testes com os outros modelos. Especificamente, os comportamentos contrastantes nos gráficos da esquerda e direita podem representar diferentes métricas de desempenho ou saídas experimentais sob condições variadas. O gráfico à esquerda ilustra uma resposta controlada com uma variável se estabilizando após mudanças iniciais, enquanto o gráfico à direita sugere uma variável que decresce, indicando uma perda de sinal ou diminuição de eficácia. A estabilidade da linha laranja em ambos os gráficos indica uma medida de linha de base ou controle, fornecendo um ponto de comparação.

Figura 11 – Acurácia e taxa de aprendizado para *EfficientNetB3* em relação a treino e validação.

Fonte: O próprio autor, 2024.

Na Figura 12, representando um cachorro, é evidente que a classe predita foi *Dog* com probabilidade de 100%, correspondendo corretamente a classificação desejada. Isso mostra que o fato de o cachorro ter uma bola na boca não foi um impedimento para a sua classificação correta.

Figura 12 – Saída do modelo *EfficientNetB3* para imagem com cachorro.

Fonte: O próprio autor, 2024.

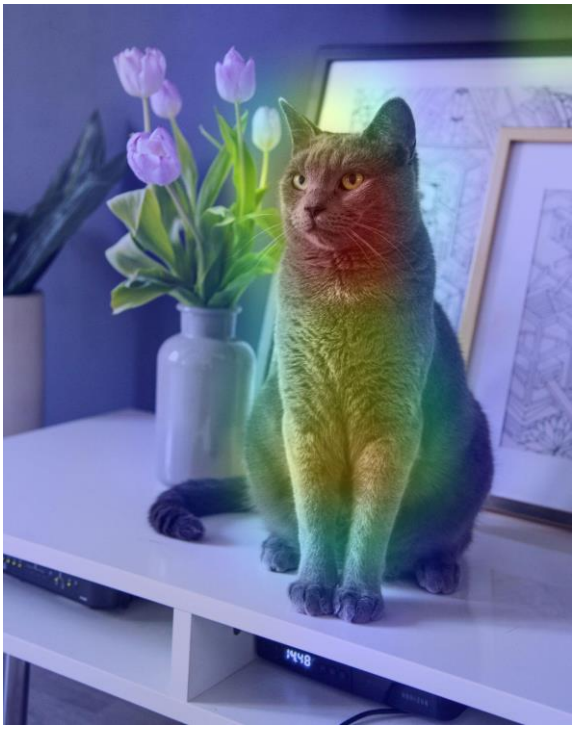
Em outro teste apresentado na Figura 13, representando dois cachorros, a classe predita foi *Dog* com probabilidade de ser da classe *Cat* de 0.0010% e probabilidade de ser da classe cachorro de 99.9989%, correspondendo corretamente a classe desejada. Como mostra o *heatmap* sobre a imagem, o gradiente foi marcado no cachorro à esquerda, levando a entender pelo motivo da variabilidade das imagens.

Figura 13 – Saída do modelo *EfficientNetB3* para imagem com dois cachorros.

Fonte: O próprio autor, 2024.

Na Figura 14, na qual é representa um gato, a classe predita foi *Cat* com probabilidade de 99.9531% e probabilidade de ser da classe *Dog* de 0.0469%, correspondendo corretamente à classe da imagem. Analisando o gradiente marcado pelo *heatmap*, nota-se que neste modelo o aprendizado em relação a gatos foi bem assertivo, visto que as cores se mostraram mais intensas.

Figura 14 – Saída do modelo *EfficientNetB3* para imagem com gato.



Fonte: O próprio autor, 2024.

A Figura 15, representando dois gatos, a classe predita foi *Cat* com probabilidade de 99.9987% e probabilidade de ser da classe *Dog* de 0.0013%, correspondendo corretamente à classe da imagem. Aqui o gradiente apresentou cores mais intensas, com destaque aos rostos dos gatos.

Figura 15 – Saída do modelo *EfficientNetB3* para imagem com dois gatos.

Fonte: O próprio autor, 2024.

Analisando a Figura 16, em que são apresentados um gato e um cachorro, a classe predita foi *Dog* com probabilidade de 69.8008% e probabilidade de ser classe *Cat* de 30.1992%.

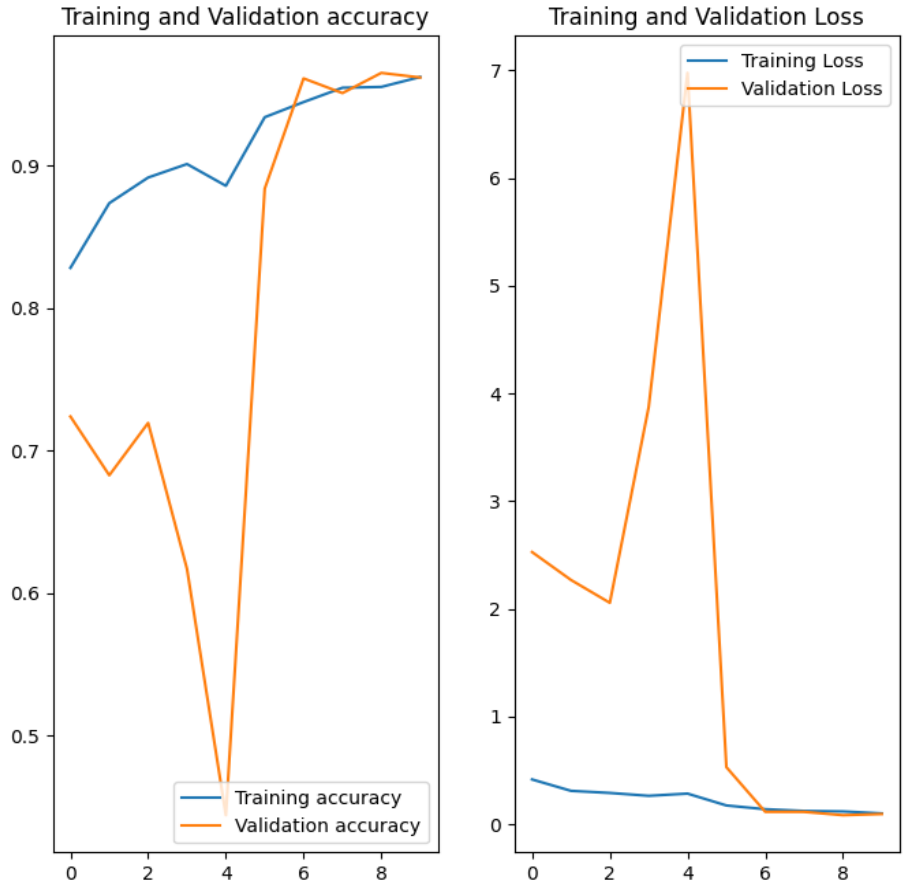
Figura 16 – Saída do modelo *EfficientNetB3* para imagem com um gato e um cachorro.

Fonte: O próprio autor, 2024.

E, por fim, o último modelo avaliado foi *MobileNetV2*, no qual se obteve a menor acurácia para a base de teste dentre os 3 modelos testados. Contudo, o modelo se mostrou robusto e assertivo, visto que a premissa é a de ser uma rede reduzida para ser executada em dispositivos móveis ou mesmo sistemas embarcados.

Na Figura 17 é apresentada a relação de acurácia e taxa de aprendizagem. Embora se tenha obtido uma queda abrupta na taxa de aprendizado para a base de validação, o modelo ao final apresentou acurácias para as duas bases muito próximas. De modo específico, analisando os gráficos apresentados é possível constatar a evolução da acurácia e perda (*loss*) durante o treinamento e validação. O gráfico à esquerda exibe a acurácia de treinamento e validação ao longo das épocas, enquanto o gráfico à direita representa a perda para ambos os conjuntos de dados. No gráfico de acurácia, observa-se que a acurácia de treinamento aumenta consistentemente, enquanto a acurácia de validação exibe uma variação maior nas primeiras épocas, possivelmente indicando uma instabilidade inicial do modelo**.** Contudo, a partir de uma certa época, a acurácia de validação estabiliza e acompanha a acurácia de treinamento, o que sugere que o modelo está aprendendo a generalizar melhor.

Figura 17 – Acurácia e taxa de aprendizado para *MobileNetV2* em relação a treino e validação**.**



Fonte: O próprio autor, 2024.

A Figura 18, representando um cachorro, a classe predita foi *Dog* com probabilidade de 99.8534% e de ser da classe *Cat* de 0.1466%, correspondendo corretamente à classe da imagem. Assim como os modelos anteriores, o fato de ter uma bola na boca do cachorro não interferiu na correta classificação.

Figura 18 – Saída do modelo *MobileNetV2* para imagem com cachorro.

Fonte: O próprio autor, 2024.

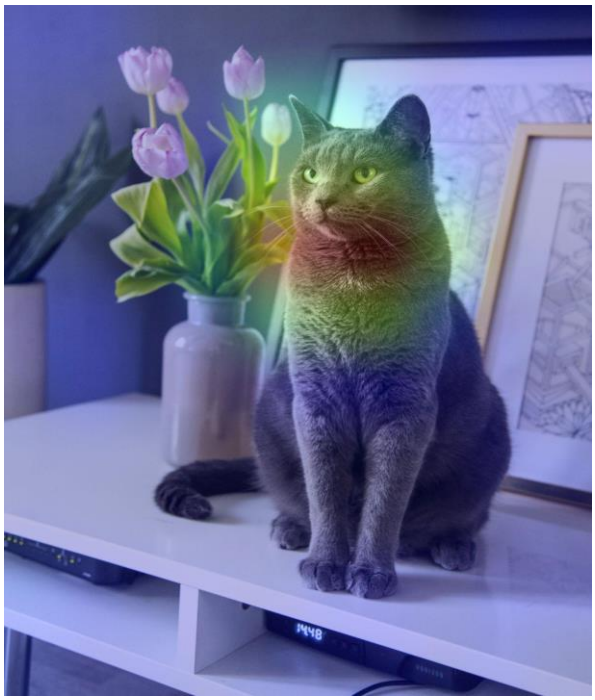
A Figura 19, representando dois cachorros, a classe predita foi *Dog* com probabilidade de 99.9976% e classe *Cat* de 0.0024%, correspondendo corretamente à classe da imagem. Um ponto a ser considerado aqui é o fato do gradiente mostrado pelo *heatmap* estar com cores mais intensas em ambos os cachorros.

Figura 19 – Saída do modelo *MobileNetV2* para imagem com dois cachorros.

Fonte: O próprio autor, 2024.

A Figura 20, representando um gato, a classe predita foi *Cat* com probabilidade de 99.9984% e probabilidade de ser da classe *Dog* de 0.0016%, correspondendo corretamente à classificação desejada.

Figura 20 – Saída do modelo *MobileNetV2* para imagem com gato.



Fonte: O próprio autor, 2024.

Na Figura 21, que é representada por dois gatos, a classe predita foi *Cat* com probabilidade de 99.9999% e probabilidade de ser da classe *Dog* de 2.3567e-05%, ou seja, muito próximo de 0%, estando de acordo com a resposta desejada. Neste caso, mesmo que o gradiente na imagem tenha focado o gato de cores mais claras, a probabilidade de ser da classe cachorro foi extremamente baixa, mostrando que o *MobileNet* se ajustou a classe *Cat*.

Figura 21 – Saída do modelo *MobileNetV2* para imagem com dois gatos.

Fonte: O próprio autor, 2024.

E, por fim, a Figura 22 representa a imagem de gato e cachorro, no qual a classe predita foi *Cat* com probabilidade de 60.5787% e probabilidade de ser da classe *Dog* de 39.4213%. Este caso mostra que, apesar de ter escolhido a classe gato, a diferença entre as probabilidades foi a menor comparando os 3 modelos testados, induzindo a concluir que o *MobileNet* conseguiu identificar as duas classes nesta figura.

Figura 22 – Saída do modelo *MobileNetV2* para imagem com um gato e um cachorro.

Fonte: O próprio autor, 2024**.**

IV. CONCLUSÃO

O objetivo deste trabalho foi comparar diferentes modelos convolucionais na tarefa de classificar imagens como sendo de gatos ou cães. Escolher o melhor modelo ou as configurações de camadas mais adequadas para cada caso apresenta desafios, pois envolve considerar o problema específico a ser resolvido, a qualidade dos dados de entrada, o poder computacional disponível e a análise dos resultados obtidos.

Os modelos testados apresentaram resultados promissores com um substancial taxa de acerto nos testes realizados, indicando o potencial dessas arquiteturas para a classificação de padrões com imagens. A técnica *GRAD-CAM* também possibilitou uma análise visual direta da eficácia do modelo em relação às probabilidades geradas.

Em estudos futuros dessa investigação cientifica pretende-se incorporar variações nos modelos testados, explorar o uso de camadas de *Dropout*, testar variações nas camadas de *MaxPooling*, ajustar hiperparâmetros, utilizar diferentes call-backs, variar número de épocas e o tamanho do lote. Outra possível melhoria envolve a base de dados, visando uma maior uniformidade dos padrões e características para otimizar a performance dos modelos.

V. REFERÊNCIAS

ALAM, M. U.; BALDVINSSON, J. R.; WANG, Y. **Exploring LRP and Grad-CAM visualization to interpret multi-label-multi-class pathology prediction using chest radiography**. In: IEEE International Symposium on Computer-Based Medical Systems, Shenzhen, China, p. 258-263, 2022.

DATA SCIENCE ACADEMY. **Deep Learning Book**. Disponível em: <https://www.deeplearningbook.com.br/>. Acesso em: 2024.

DUTT, V.; SASUBILLI, S. M.; YERRAPATI, A. E**. Dynamic Information Retrieval with Chatbots: A Review of Artificial Intelligence Methodology**. In: International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, p. 1299–1303, 2022.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. Mass: Addison-Wesley, 1989.

HAYKIN, Simon. **Neural Networks and Learning Machines**. 3 ª. ed. Upper Saddle River, NJ: Prentice Hall, 2009.

HE, K., ZHANG, X., REN, S., SUN, J. **Deep Residual Learning for Image Recognition**, Microsoft Research, 2015.

HWU, Wen-mei W. **GPU Computing Gems Jade Edition**, Burlington, MA: Morgan Kaufmann, 1ª ed, 2011.

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. Ann Arbor: University of Michigan Press, 1975.

MENDONÇA, M., KONDO, H. S., BOTONI de SOUZA, L., PALÁCIOS, R. H. C., SILVA de ALMEIDA, J. P. L. **Semi-Unknown Environments Exploration Inspired by Swarm Robotics using Fuzzy Cognitive Maps**, In: IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), New Orleans, LA, USA, pp. 1-8, 2019.

PARKHI, O. M., VEDALDI, A., ZISSERMAN, A., JAWAHAR, C. V. **Cats and Dogs**, em IEEE Conference on Computer Vision and Pattern Recognition, 2012.

PASSINO, M. K., YOURKOVICH, S. **Fuzzy Control**. Menlo Park: Addison-Wesley, 1997.

SANDLER, M.; HOWARD, A.; ZHU, M; ZHMOGINOV, A; CHEN, L-C. **MobileNetV2: Inverted Residuals and Linear Bottlenecks**. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4510-4520, 2018.

SELVARAJU, R. R., COGSWELL, M., Das, A., VEDANTAM, R., PARIKH, D., BATRA, D. **Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization**, International Journal of Computer Vision, 2019.

SICILIANO, B.; KHATIB, O. (EDS.). **Springer Handbook of Robotics**. 2. ed. Heidelberg: Springer-Verlag Berlin Heidelberg, 2016.

SILVA, I. R. R.; SOUZA, R. G.; SILVA, G. S. L.; OLIVEIRA, C. S.; CAVALCANTI, L. H.; BEZERRA, R. S., *et al*. **Utilização de Redes Convolucionais Para Classificação e Diagnóstico da Doença de Alzheimer**, In: II Simpósio de Inovação em Engenharia Biomédica, 2018.

TAN, M., Le, Q. V. **EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks**, In: International Conference on Machine Learning, 2019.

YU, J.; CAO, J.; HE, R. **Improving Subgraph Recognition with Variational Graph Information Bottleneck**. In: IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), New Orleans, LA, USA, p. 19374-19383, 2022.