UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
ENGENHARIA DE COMPUTAÇÃO

GUILHERME DE OLIVEIRA MONTEIRO PEIXOTO

TREINAMENTO E AVALIAÇÃO DE REDES NEURAIS CONVOLUCIONAIS (CNNs) PARA A IDENTIFICAÇÃO DE COMPORTAMENTO CANINO EM IMAGENS

Maceió / AL
2024

GUILHERME DE OLIVEIRA MONTEIRO PEIXOTO

TREINAMENTO E AVALIAÇÃO DE REDES NEURAIS CONVOLUCIONAIS (CNNs) PARA A IDENTIFICAÇÃO DE COMPORTAMENTO CANINO EM IMAGENS

Trabalho de Conclusão de Curso submetido ao curso de Engenharia de Computação do Instituto de Computação da Universidade Federal de Alagoas como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador(a): Prof. Dr. Thales Miranda de Almeida Vieira

Maceió / AL

2024

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

P379t Peixoto, Guilherme de Oliveira Monteiro.
Treinamento e avaliação de redes neurais convolucionais (CNNs) para a identificação de comportamento canino em imagens / Guilherme de Oliveira Monteiro Peixoto. – 2024.
60 f. : il.

Orientador: Thales Miranda de Almeida.
Monografia (Trabalho de conclusão de curso em Engenharia de Computação) - Universidade Federal de Alagoas, Instituto de Computação. Maceió, 2024.

Bibliografia: f. 56-60.

1. Redes neurais convolucionais (Computação). 2. Classificação de ações em vídeos. 3. Saúde animal - Monitoramento. 4. Aprendizado do computador. I. Título.

CDU: 004.383.8.032.26

,

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL

Instituto de Computação – IC

Campus A. C. Simões - Av. Lourival de Melo Mota, BL 12
Tabuleiro do Martins, Maceió/AL - CEP: 57.072-970
Telefone: (082) 3214-1401

Trabalho de Conclusão de Curso - TCC

Formulário de Avaliação

Nome do Aluno

Guilherme De Oliveira Monteiro Peixoto

Nº de Matrícula

16113149

Título do TCC (Tema)

TREINAMENTO E AVALIAÇÃO DE REDES NEURAIS CONVOLUCIONAIS (CNNs) PARA A IDENTIFICAÇÃO DE COMPORTAMENTO CANINO EM IMAGENS

Banca Examinadora

THALES MIRANDA DE ALMEIDA VIEIRA
Nome do Orientador
_____
Assinatura

ADRIANO OLIVEIRA BARBOSA
Nome do Professor
_____
Assinatura

XU YANG
Nome do Professor
_____
Assinatura

Data da Defesa

06/12/2024

Nota Obtida

9,5 (nove inteiros e cinco décimos)

Observações

Nada a declarar.

Coordenador do Curso
De Acordo

_____
Assinatura

Agradeço aos meus pais, Alberto Monteiro Peixoto e Divaneide da Silva Oliveira, pelo incentivo constante. Dedico este trabalho, com profundo carinho e admiração, ao meu avô, Antonio Brasiliano Peixoto, que foi, para mim, o maior exemplo de integridade a ser seguido.

RESUMO

O reconhecimento automático de ações em imagens pode transformar profundamente o modo como interagimos e cuidamos dos pets. Quando aplicado ao monitoramento da saúde canina, essa tecnologia tem o potencial de detectar sinais precoces de doenças ou desconforto, permitindo intervenções rápidas e eficazes. Este trabalho tem como objetivo desenvolver e validar um modelo de Rede Neural Convolucional (CNN) capaz de classificar, a partir de uma única imagem, se um cão está se coçando ou não. A ação de coçar, quando realizada excessivamente, pode ser um sinal de alergias alimentares ou ambientais, pulgas, carrapatos ou outros parasitas, além de condições de pele como dermatite e infecções. O modelo CNN-DATAUG-RELU-L2-DROPOUT foi projetado com uma arquitetura que utiliza camadas convolucionais com funções de ativação ReLU, regularização L2 e *dropout*. O modelo foi treinado e avaliado com um conjunto de dados coletado por uma empresa multinacional da França, que foi posteriormente anotado ao longo deste trabalho, alcançando uma acurácia de 87,83%, precisão de 90,23% e cobertura de 89,31%. Esses resultados indicam um bom desempenho em condições controladas. Embora modelos de reconhecimento de ações frequentemente utilizem sequências de *frames*, este estudo demonstra que a análise de *frames* individuais pode ser uma abordagem simples e eficaz.

Palavras-chave: redes neurais convolucionais; classificação de ações; monitoramento de saúde animal; deep learning.

ABSTRACT

Automatic action recognition can profoundly transform the way we interact with and care for pets. When applied to monitoring canine health, this technology has the potential to detect early signs of disease or discomfort, allowing for quick and effective interventions. This work aims to develop and validate a Convolutional Neural Network (CNN) model capable of classifying, from a single image, whether a dog is scratching or not. The scratching action, when performed excessively, can be a sign of food or environmental allergies, fleas, ticks, or other parasites, as well as skin conditions such as dermatitis and infections. The CNN-DATAUG-RELU-L2-DROPOUT model was designed with an architecture that utilizes convolutional layers with ReLU activation functions, L2 regularization, and dropout. The model was trained and evaluated on a specific dataset, achieving an accuracy of 87.83%, a precision of 90.23%, and a cobertura of 89.31%. These results indicate good performance under controlled conditions. Although action recognition models often use sequences of frames, this study demonstrates that analyzing individual frames can be a simple and effective approach.

Keywords: convolutional neural networks; action classification; animal health monitoring; Convolutional Neural Network; deep learning.

SUMÁRIO

1. INTRODUÇÃO

A detecção precoce de comportamentos caninos relacionados à saúde, como prurido, lambedura excessiva, tremores e letargia, é crucial para a identificação de doenças dermatológicas, endócrinas, ortopédicas e neurológicas em cães. Esses sinais podem indicar a presença de doenças que, quando reconhecidas rapidamente, permitem intervenções mais eficazes, melhorando significativamente o prognóstico e reduzindo os custos do tratamento (VIRGA, 2003).

Por exemplo, a dermatite alérgica à picada de pulga (VIRGA, 2003) é uma reação alérgica à saliva das pulgas que causa prurido intenso. A coceira costuma se concentrar na base da cauda, abdômen e regiões lombares, levando o cão a se lamber excessivamente e pode resultar em lesões na pele e queda de pelos. Se não tratada prontamente, essa condição pode evoluir para infecções secundárias graves, devido à irritação constante e ao comprometimento da barreira cutânea.

Outro exemplo é a piodermite (VIRGA, 2003), uma infecção bacteriana da pele que frequentemente surge como complicação de outras condições que causam prurido, como alergias ou infestação por ectoparasitas. Os sintomas incluem coceira, erupções cutâneas, crostas e perda de pelos.

Comportamentos como o prurido podem, portanto, ser sinais iniciais de uma ampla gama de problemas de saúde. Detectá-los cedo é essencial para que as intervenções médicas sejam rápidas e precisas, prevenindo complicações e proporcionando um melhor cuidado para o animal.

No entanto, a identificação desse comportamento por meio de observação humana é limitada pela sua subjetividade e principalmente pela dificuldade de realizar um monitoramento contínuo.

O uso de sensores e coleiras inteligentes tem se mostrado uma alternativa para o monitoramento automatizado, mas enfrenta desafios como o custo elevado, a precisão limitada na detecção de certos comportamentos e o desconforto causado pelo uso prolongado dos dispositivos. Isso levanta a necessidade de soluções que possam realizar o monitoramento sem interferir no comportamento natural do animal

(STABACH et al., 2020). Nesse contexto, a visão computacional emerge como uma abordagem promissora.

Enquanto o reconhecimento de ações humanas com visão computacional é um campo amplamente explorado (MARR; VAINA, 1982), com aplicações consolidadas em segurança (RIBEIRO et al., 2009), saúde e acessibilidade, o reconhecimento de ações em animais, especialmente cães, é uma área pouco estudada.

Classificar comportamentos caninos, como o prurido, usando visão computacional apresenta diversas barreiras técnicas e práticas que tornam essa tarefa desafiadora. Primeiramente, existe uma grande variabilidade comportamental entre diferentes cães. Fatores como raça, tamanho, idade e nível de atividade influenciam a forma como os comportamentos são manifestados, o que dificulta a criação de modelos generalizáveis que funcionem bem para uma ampla variedade de cães.

Outro desafio significativo é a escassez de dados anotados. Para realizar o treinamento eficiente, são necessários grandes e bem rotulados conjuntos de dados. A complexidade do cenário em que o comportamento ocorre também afeta a precisão do modelo. Ambientes com fundos visualmente complexos, variação de iluminação ou a presença de outros animais e objetos em movimento dificultam a detecção precisa dos movimentos relacionados ao prurido. Além disso, a necessidade de realizar a detecção em tempo real representa uma barreira, pois os modelos devem equilibrar a precisão com a eficiência de processamento para fornecer respostas rápidas.

Essas barreiras destacam a complexidade do problema e diante desse cenário, este trabalho propõe o desenvolvimento de um modelo baseado em redes neurais convolucionais para a classificação do comportamento de prurido em cães. O objetivo é criar uma metodologia que seja capaz de identificar com precisão esse comportamento, utilizando como parâmetros de validação a acurácia do modelo em dados de teste coletados por uma multinacional francesa.

Panorama do reconhecimento de ações em visão computacional

O reconhecimento de ações na área de visão computacional começou a se destacar a partir de 1980, juntamente com o desenvolvimento de técnicas para analisar movimentos de objetos em vídeos (HESTER; CASASENT, 1980; MARR; VAINA, 1982). No início, essas técnicas dependiam de métodos baseados em regras e reconhecimento de padrões simples, como técnicas que analisam o padrão de movimento dos píxeis entre *frames* consecutivos de um vídeo, ou técnicas que utilizam padrões predefinidos para identificar ações específicas comparando as imagens capturadas com esses padrões (SHAH; JAIN, 1984; SETHI; SALARI; VEMURI, 1988; CÉDRAS; SHAH, 1995).

Com a chegada das redes neurais artificiais e o ressurgimento da Inteligência Artificial nos anos 2000, impulsionados pelo aumento da capacidade de computação e pela disponibilidade de grandes conjuntos de dados (HUNT, 2014), o campo da visão computacional, especialmente o reconhecimento de ações, experimentou transformações no modo como os algoritmos processam imagens e vídeos (WANG; HU; TAN, 2003; BOBICK; DAVIS, 2001).

Além disso, a revolução do Aprendizado Profundo a partir de 2010 transformou a visão computacional (SEJNOWSKI, 2018), com redes neurais convolucionais (LECUN et al., 1998) sendo muito usadas em tarefas como classificação de imagens, detecção de objetos e, especialmente, reconhecimento de ações (JI et al., 2012).

Principais aplicações do reconhecimento de ações em visão computacional

Na área de segurança, a tecnologia de reconhecimento de ações usando aprendizado profundo é fundamental para monitorar comportamentos suspeitos automaticamente (NGUYEN; MEUNIER, 2019). Na saúde, essa mesma tecnologia é empregada para o monitoramento contínuo de pacientes. Ela ajuda a prevenir quedas e a otimizar os cuidados oferecidos, monitorando movimentos que possam indicar riscos ou necessidades médicas (RAJALAXMI et al., 2022). No campo da acessibilidade, facilita a comunicação com pessoas surdas ao traduzir a linguagem de sinais em tempo real, eliminando barreiras na comunicação (GARCIA; VIESCA, 2016). Além disso, em veículos autônomos, a tecnologia antecipa ações de

pedestres e outros veículos, aprimorando as decisões de condução e aumentando a segurança no trânsito (ZHANG; BERGER, 2023). Essas aplicações destacam a ampla utilidade e o impacto do reconhecimento de ações em setores como segurança, saúde, acessibilidade e transporte.

Relevância do reconhecimento de ações animais

Os estudos nessa área têm se concentrado predominantemente em humanos (WU et al., 2017). Uma exploração mais aprofundada das tendências gerais em pesquisas de reconhecimento de ações confirma essa disparidade (WU et al., 2017; FENG et al., 2021). Enquanto houve um desenvolvimento extensivo e aplicação de tecnologias de reconhecimento de ações humanas (NGUYEN; MEUNIER, 2019; RAJALAXMI et al., 2022), avanços similares no reconhecimento de ações de animais, especialmente além de animais domésticos ou de laboratório, não são comuns.

Entretanto, o interesse para estudar comportamentos animais usando visão computacional vem crescendo nos últimos anos (DEAKIN et al., 2019; SCHMIDT et al., 2022). Um exemplo disso é o estudo sobre o monitoramento automatizado de peixes-zebra que, após procedimentos invasivos, utilizou tecnologias de visão computacional para analisar mudanças no comportamento do animal em um ambiente controlado (DEAKIN et al., 2019).

Conforme mostrado, o reconhecimento automático de comportamentos permite a monitoração contínua de animais em zoológicos, reservas, ou até em ambientes domésticos, sem a necessidade de intervenção humana constante (KOGER et al., 2023). Isso reduz o estresse causado pela presença de humanos e aumenta a precisão na detecção de comportamentos anormais ou sinais de doença, permitindo intervenções rápidas e precisas que melhoram o bem-estar e a saúde dos animais (RAVBAR; BRANSON; SIMPSON, 2019).

Tratando-se especificamente de cães, o reconhecimento automático de ações pode transformar o modo como interagimos e cuidamos deles. Esta tecnologia é capaz de oferecer uma variedade de aplicações práticas que beneficiam tanto os animais quanto seus donos (JUKAN; MASIP-BRUIN; AMLA, 2017). Por exemplo, quando aplicada ao monitoramento da saúde canina, essa tecnologia pode detectar

sinais precoces de doenças ou desconforto. Esse recurso é especialmente valioso em ambientes urbanos densos, em que os cães podem enfrentar maiores níveis de estresse, ou em abrigos, em que ajuda a ajustar o ambiente para melhor satisfazer as necessidades dos animais.

Além disso, ao integrar essa tecnologia com outras soluções de casas inteligentes, como alimentadores automáticos e portas que se abrem por reconhecimento, isso garante que, mesmo quando os cuidadores não estão presentes, os cães permaneçam seguros, saudáveis e contentes.

1.1 Objetivos e justificativa

1.1.1 Objetivo geral

O objetivo principal deste Trabalho de Conclusão de Curso (TCC) é desenvolver e validar um modelo de rede neural convolucional capaz de classificar, a partir de uma única imagem, se um cão está se coçando ou não.

1.1.2 Objetivos específicos

- Preparar o conjunto de dados a partir de vídeos de cães exibindo uma variedade de ações, com foco na ação de se coçar;
- Implementar técnicas de pré-processamento e aumento de dados para garantir a qualidade e diversidade do conjunto de dados;
- Projetar uma arquitetura de CNN adequada para identificar a ação;
- Treinar o modelo usando o conjunto de dados;
- Avaliar a acurácia e consistência do modelo de CNN;
- Discutir a integração do modelo de CNN desenvolvido em sistemas de monitoramento animal futuros.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Introdução à visão computacional

A visão computacional é uma subárea da Inteligência Artificial que se concentra no desenvolvimento de técnicas para interpretar e extrair informações úteis de imagens e vídeos.

Conforme comentado no capítulo anterior, as primeiras investigações em visão computacional começaram na década de 1960 com o desenvolvimento de técnicas para extração de características tridimensionais de objetos a partir de imagens bidimensionais (HUNT, 2014). Durante as décadas de 1970 e 1980, o campo expandiu-se com o aumento da capacidade de processamento dos computadores, permitindo o desenvolvimento de algoritmos para análise de texturas, detecção de bordas e reconhecimento de objetos (HESTER; CASASENT, 1980; MARR; VAINA, 1982). Este período também viu a aplicação de sistemas de visão computacional em contextos industriais, impulsionando a automação de processos de inspeção e montagem (CÉDRAS; SHAH, 1995; HUNT, 2014).

No início do século 21, o campo da visão computacional transformou-se com a introdução do aprendizado profundo e das redes neurais convolucionais (CNNs). Este período também viu a aplicação de sistemas de visão computacional em contextos industriais, impulsionando a automação de processos de inspeção e montagem (CÉDRAS; SHAH, 1995; HUNT, 2014).

Os campos de estudo na visão computacional incluem o processamento de imagens, que lida com a manipulação direta dos valores de píxeis para realizar operações como ajuste de brilho, contraste, equalização de histograma, normalização de cores e filtragem; A detecção de características, que busca identificar pontos de interesse em uma imagem, como bordas, cantos, texturas e outros elementos distintivos; A segmentação, onde o objetivo é identificar e isolar regiões consistentes numa imagem; A reconstrução 3D, que se dedica a criar modelos tridimensionais de objetos a partir de imagens bidimensionais. Isso é particularmente útil em aplicações que requerem uma compreensão espacial detalhada dos objetos; O reconhecimento de objetos, que envolve rotular

automaticamente os objetos presentes nas imagens; A análise de movimento, que lida com a detecção e acompanhamento de objetos em movimento em vídeos (SZELISKI, 2022).

Essas sub-áreas interagem e se complementam, formando a base para o desenvolvimento de sistemas de Visão Computacional que podem interpretar e entender o mundo visual de maneira semelhante aos humanos.

2.2 Fundamentos de aprendizado de máquina

O aprendizado de máquina é uma subárea da inteligência artificial que se concentra no desenvolvimento de sistemas que podem aprender a partir de dados e fazer previsões ou tomar decisões sem ser explicitamente programados para tal. O processo envolve o treinamento de um modelo estatístico usando um conjunto de dados, em que o modelo aprende padrões e características desse conjunto e, então, utiliza esse aprendizado para prever resultados em novos dados (EL NAQA; MURPHY, 2015). Os conceitos fundamentais incluem características, que são os atributos ou variáveis dos dados; rótulos, que são os resultados ou respostas associados às características; e o modelo, que é o algoritmo que aprende a relação entre características e rótulos.

O aprendizado de máquina é comumente dividido em três categorias principais, cada uma com seus métodos, aplicações e desafios específicos: aprendizado supervisionado, não supervisionado e por reforço (EL NAQA; MURPHY, 2015). Essa categorização ajuda a definir a abordagem e as técnicas usadas no desenvolvimento de modelos de aprendizado de máquina, dependendo da natureza dos dados disponíveis e do tipo de problema que se deseja resolver.

No aprendizado supervisionado, os modelos são treinados usando um conjunto de dados que já contém as respostas desejadas, chamadas de rótulos. Esse tipo de aprendizado é utilizado quando o objetivo é prever um resultado a partir de um conjunto de entradas (ALPAYDIN, 2021).

Por outro lado, o aprendizado não supervisionado é usado quando não há rótulos disponíveis nos dados. Neste caso, o objetivo é identificar padrões ou

estruturas inerentes aos dados. Métodos comuns incluem *clustering* (agrupamento) e redução de dimensionalidade (EL NAQA; MURPHY, 2015; ALPAYDIN, 2021).

O aprendizado por reforço difere significativamente dos dois anteriores, pois se concentra em desenvolver um sistema (agente) que melhora sua performance baseado em *feedback* contínuo de suas ações (EL NAQA; MURPHY, 2015). Este tipo de aprendizado é inspirado pela maneira como os seres humanos aprendem a partir das consequências de suas ações. Em ambientes como jogos ou robótica, o agente aprende a tomar decisões, recebendo recompensas ou penalidades com base no sucesso ou falha de suas ações anteriores. O objetivo é maximizar a soma das recompensas ao longo do tempo, ajustando as estratégias conforme necessário.

Cada um desses tipos de aprendizado utiliza técnicas e algoritmos distintos e é escolhido com base nas características específicas do problema e do conjunto de dados à disposição. Essa diversidade permite que o campo do aprendizado de máquina seja aplicado em diversas situações (SOUTO; MELLO; FURTADO, 2019).

2.2.1 Métricas de avaliação

A avaliação de modelos de aprendizado de máquina é fundamental para determinar a eficácia de um algoritmo em realizar as tarefas para as quais foi designado. As métricas de avaliação variam conforme o tipo de tarefa (ALPAYDIN, 2021). Para tarefas de classificação, métricas como acurácia, precisão, cobertura e pontuação F1 (*F1-score)* são comuns (POWERS, 2020). A precisão mede a proporção de identificações corretas (positivas) entre todas as identificações positivas feitas pelo modelo, enquanto a cobertura avalia a proporção de positivos verdadeiros identificados em relação ao total de casos positivos reais. O *F1-score* é a média harmônica de precisão e cobertura, fornecendo um balanço entre essas duas métricas. Além disso, a área sob a curva de característica de operação do receptor (ROC, na sigla em inglês) é utilizada para medir a capacidade do modelo de distinguir entre as classes em diferentes limiares de decisão (HANLEY; MCNEIL, 1982).

Um dos principais desafios no aprendizado de máquina é o sobreajuste, que ocorre quando um modelo se ajusta tão bem aos dados de treinamento que perde a capacidade de generalizar para novos dados. Isso tende a acontecer quando o

modelo é excessivamente complexo, com muitos parâmetros em comparação ao número de observações disponíveis. Para detectar o sobreajuste, técnicas como a validação cruzada são comumente utilizadas; nesse processo, o conjunto de dados é dividido em várias partes, permitindo que o modelo seja treinado em algumas dessas partes e testado em outras, simulando sua capacidade de generalização em amostras não utilizadas durante o treinamento. Se o desempenho do modelo cair significativamente ao ser testado em dados que não viu antes, é um sinal claro de que ele está sobreajustado aos dados de treinamento.

No entanto, apenas detectar o sobreajuste com validação cruzada não resolve o problema. Para mitigar essa questão, várias técnicas podem ser aplicadas, como a regularização, técnicas de aumento de dados, a redução da complexidade do modelo, a parada precoce (*early stopping*) e o *dropout*.

Finalmente, questões éticas também desempenham um papel crítico no desenvolvimento de sistemas de aprendizado de máquina. Preocupações com viés e a transparência das decisões tomadas pelos modelos são relevantes. Viés nos dados de treinamento pode levar a resultados enviesados, que perpetuam e amplificam desigualdades existentes. A transparência é essencial para construir a confiança do usuário, especialmente em aplicações críticas (CRAWFORD, 2021). Por isso, explicar as decisões dos modelos de forma compreensível para humanos é essencial para a adoção responsável da tecnologia.

2.3 Aprofundamento em aprendizado profundo

O aprendizado profundo é uma subcategoria do aprendizado de máquina, que se refere ao uso de redes neurais artificiais com muitas camadas de neurônios, conhecidas como redes neurais profundas. Essas redes são capazes de aprender representações de dados em múltiplos níveis de abstração, permitindo que o sistema identifique padrões complexos e sutis nos dados (FUKUSHIMA, 1980). O aprendizado profundo se distingue por sua capacidade de processar grandes volumes de dados e aprender características automaticamente, sem a necessidade de intervenção ou programação manual específica para cada característica. Isso é possível graças a uma arquitetura de aprendizado hierárquico, na qual

características de baixo nível são usadas para aprender características de nível mais alto.

Diferenças entre aprendizado profundo e aprendizado de máquina tradicional

Enquanto o aprendizado de máquina tradicional muitas vezes depende da seleção manual de características e da engenharia de atributos para melhorar o desempenho do modelo, o aprendizado profundo automatiza essas etapas (LECUN et al., 1989). As redes neurais profundas são capazes de identificar as características importantes por conta própria durante o processo de treinamento. Esta é uma das principais vantagens do aprendizado profundo, pois reduz a necessidade de intervenção humana e aumenta a capacidade do modelo de trabalhar com dados brutos e complexos, como imagens, áudio e texto não estruturados (LECUN; BENGIO; HINTON, 2015). Além disso, o aprendizado profundo exige um poder computacional consideravelmente maior do que muitas técnicas tradicionais de aprendizado de máquina, especialmente durante as fases de treinamento, devido à complexidade e profundidade das redes neurais. Isso geralmente implica em um maior consumo de tempo e recursos, incluindo o uso de GPUs e *clusters* de computação em nuvem para processar e analisar os dados de maneira eficiente (LECUN; BENGIO; HINTON, 2015).

Enquanto o aprendizado de máquina tradicional pode ser eficaz em problemas em que os padrões são mais facilmente discerníveis e os conjuntos de dados são menores e menos complexos, o aprendizado profundo é particularmente poderoso em cenários em que os padrões são extremamente complexos ou ocultos e os conjuntos de dados são grandes e ricos em informações (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.5 Técnicas de pré-processamento em visão computacional

O pré-processamento de imagens é um passo crucial no fluxo de trabalho de visão computacional. Esta etapa é importante para melhorar o desempenho dos modelos, pois ajuda a reduzir a variabilidade não essencial nas imagens e acentuar os aspectos que são mais importantes para a análise. Ao simplificar os dados de entrada, o pré-processamento permite que os modelos se concentrem nas

características verdadeiramente relevantes, aumentando assim a eficiência e a precisão dos resultados (GUO et al., 2016). Além disso, ao normalizar e padronizar as imagens, reduz-se o risco de sobreajuste e aumenta-se a capacidade do modelo de generalizar a partir de novos dados de entrada, que podem não ser idênticos aos dados vistos durante o treinamento.

A normalização é uma das técnicas de pré-processamento mais comuns e envolve a padronização das escalas de intensidade das imagens (SOUTO; MELLO; FURTADO, 2019). Isso geralmente significa ajustar o brilho e o contraste das imagens ou escalar os valores de pixel para uma faixa comum, como 0 a 1 ou -1 a 1. Este processo ajuda a mitigar as diferenças de iluminação e exposição entre as imagens capturadas em diferentes condições de iluminação e ambientes, tornando o modelo menos sensível a essas variações e mais focado na identificação de padrões relevantes (POWERS, 2020).

O aumento de dados é outra técnica fundamental, especialmente útil quando o conjunto de dados disponível é limitado. Esta técnica envolve a criação artificial de novos dados de treinamento a partir de dados existentes por meio de modificações aleatórias e controladas, como rotação, translação, redimensionamento e alterações de cor. O objetivo é simular a variação natural na categoria de objetos que o modelo tentará reconhecer, ensinando-o a ser robusto a mudanças na orientação, escala e iluminação dos objetos nas imagens (SOUTO; MELLO; FURTADO, 2019).

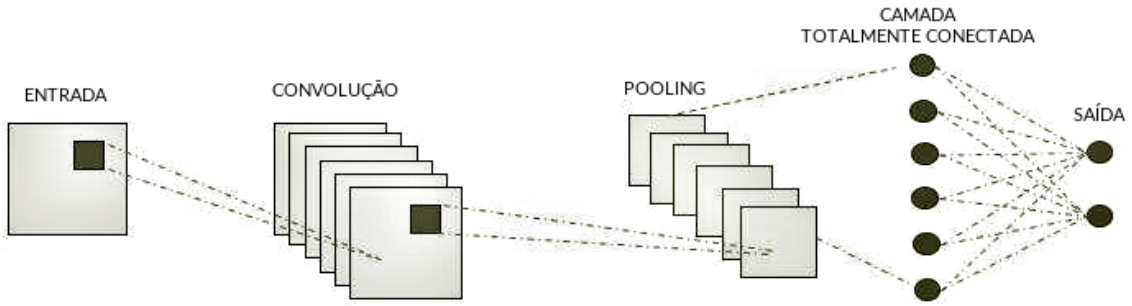
A correção geométrica é crucial para alinhar e corrigir distorções em imagens que podem surgir devido à perspectiva ou ao ângulo de captura. Técnicas como a transformação de perspectiva são usadas para ajustar a imagem de forma que simule uma visão frontal do objeto, o que é particularmente importante em aplicações como a reconhecimento de texto ou análise de imagens de satélite. Essa correção ajuda a padronizar a entrada para que o modelo possa aplicar de maneira consistente os padrões aprendidos, independentemente das variações geométricas nas novas imagens.

2.6 Contextualização do uso de CNNs

2.6.1 Definição de redes neurais convolucionais

Redes neurais convolucionais são uma classe de redes neurais profundas, altamente eficazes na análise visual de imagens. Essas redes são especificamente projetadas para processar dados que têm uma estrutura em grade, como imagens, que são representadas como matrizes de píxeis.

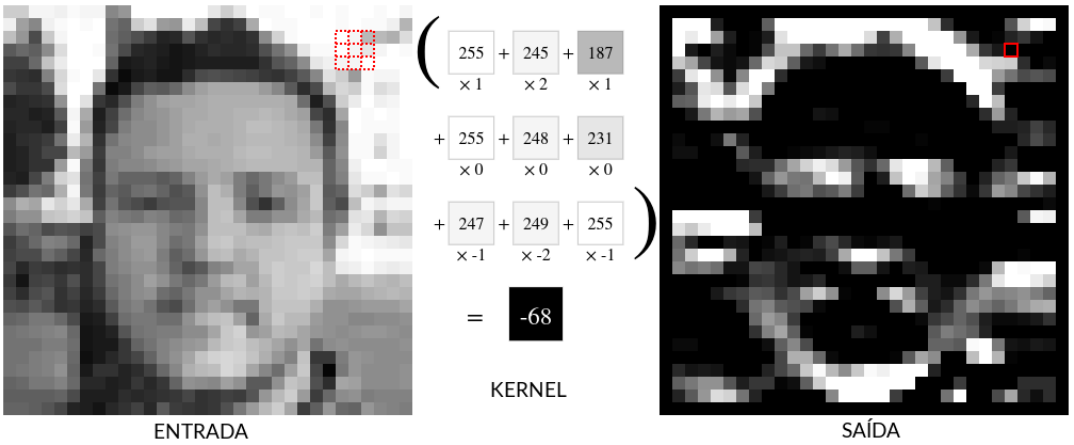
Figura 1: Arquitetura básica de uma CNN.



Fonte: elaborado pelo autor, 2024.

Conforme mostrado na Figura 1, o componente principal de uma CNN é a camada convolucional, na qual um conjunto de filtros (ou *kernels*) é aplicado à imagem de entrada para criar mapas de características. Cada filtro é uma pequena janela que desliza sobre a imagem, realizando uma operação de convolução, que resulta em um mapa de características representando a resposta do filtro a diferentes regiões da imagem. Estes filtros são cruciais porque permitem que a rede aprenda padrões espaciais específicos, tais como bordas, texturas ou formas, que são essenciais para tarefas de reconhecimento de imagem.

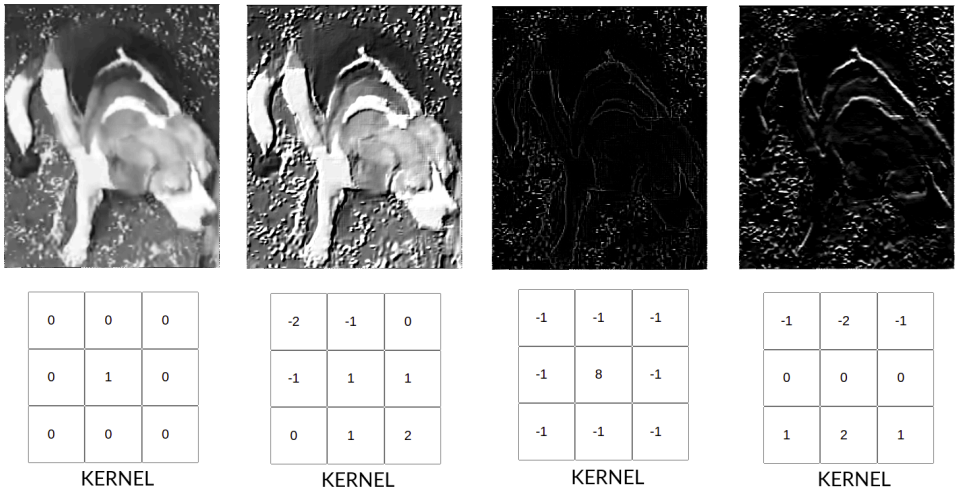
Figura 2: Arquitetura básica de uma CNN.



Fonte: Disponível em: https://setosa.io/ev/image-kernels/. Acesso em 24 out. 2024

Os filtros na camada convolucional são geralmente menores que a dimensão da imagem de entrada, por exemplo, 3x3 ou 5x5 píxeis, como mostrado na Figura 2, mas são aplicados em toda a extensão da imagem. Durante o treinamento, a rede ajusta os valores dos filtros de modo que eles sejam capazes de detectar as características relevantes para a tarefa. Cada filtro gera um mapa de características distinto (Figura 3), que pode ser visto como uma representação da presença do padrão que o filtro foi treinado para detectar em várias posições na imagem.

Figura 3: Aplicação de *Kernels* com diferentes valores.



Fonte: elaborado pelo autor, 2024.

Além disso, as camadas convolucionais geralmente são seguidas por uma operação de não-linearidade, como a função de ativação de unidade linear retificada

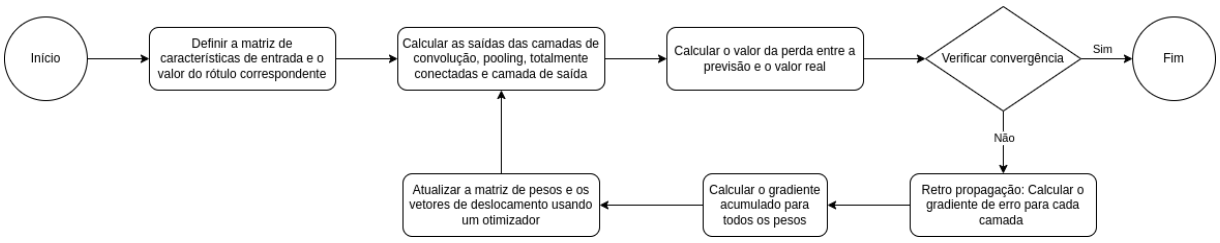
(ReLU), que aplica uma transformação *element-wise*. Isso introduz não-linearidade no modelo, permitindo que ele aprenda representações mais complexas (PRINCE, 2012).

A arquitetura de uma CNN tipicamente inclui várias dessas camadas convolucionais, intercaladas com camadas de agrupamento (*pooling*), que reduzem a dimensionalidade dos mapas de características enquanto preservam as características mais importantes. Esta combinação permite que a rede execute uma forma de extração de características hierárquica: camadas mais profundas na rede podem reconhecer características cada vez mais complexas, construindo sobre as saídas das camadas anteriores. Esse processo é fundamental para lidar com a grande variabilidade de formas e tamanhos que os objetos podem apresentar em imagens visuais.

Além das camadas convolucionais e de agrupamento, as CNNs geralmente incluem camadas de normalização, como as camadas de normalização por lotes (*batch normalization*), e camadas totalmente conectadas (*fully connected layers*) no final da rede (NAIR; HINTON, 2010). A normalização por lotes ajuda a acelerar o treinamento da rede e a reduzir o problema de sobreajuste, estabilizando as distribuições das entradas das camadas ao longo do treinamento. As camadas totalmente conectadas, por outro lado, são usadas para combinar todas as características aprendidas para fazer predições finais, como a classificação de uma imagem em várias categorias (NAIR; HINTON, 2010).

O treinamento de uma rede neural convolucional é realizado através de um processo iterativo que envolve o algoritmo de retropropagação e um otimizador, como o gradiente descendente estocástico.

Figura 4: Fluxograma do Processo de Treinamento de uma CNN.



Fonte: elaborado pelo autor, 2024.

A retropropagação é uma técnica utilizada para treinar redes neurais, ajustando os pesos com base no erro entre as predições da rede $\widehat{y}$ e os rótulos reais y. O processo começa com o cálculo de uma função de perda L, que quantifica a diferença entre a saída prevista pela rede e o valor real. Por exemplo, para um problema de regressão, a função de perda comum é o erro quadrático médio (MSE), descrito pela equação (1):

(1) $$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Em problemas de classificação, a função de perda frequentemente utilizada é a Entropia Cruzada, conforme a equação (2):

(2) $$L = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

Uma vez que a função de perda é calculada, o próximo passo é ajustar os pesos da rede para minimizar essa função. Isso é feito computando o gradiente da função de perda em relação a cada peso ω_{ij}, conforme a equação (3):

(3) $$\frac{\partial L}{\partial w_{ij}}$$

Os pesos são então atualizados para reduzir o erro, seguindo o método de descida do gradiente, como mostrado na equação (4):

(4) $$w_{ij} := w_{ij} - \eta\frac{\partial L}{\partial w_{ij}}$$

em que η é a taxa de aprendizado, que determina o tamanho do passo na direção que minimiza o erro.

A retropropagação ocorre de trás para frente na rede, ou seja, a partir da camada de saída em direção às camadas iniciais. Para a camada de saída, o gradiente do erro é calculado diretamente a partir da função de perda e da saída prevista, seguindo a equação (5):

(5) $$\delta^{(L)} = \frac{\partial L}{\partial \hat{y}} \odot f'(z^{(L)})$$

Sendo $\odot$ o produto de Hadamard, ou seja, a multiplicação elemento a elemento entre dois vetores ou matrizes. Nas camadas ocultas, o gradiente é propagado para trás usando a equação (6):

(6) $$\delta^{(l)} = (\delta^{(l+1)} \cdot W^{(l+1)}) \odot f'(z^{(l)})$$

Aqui, $W^{(l+1)}$ são os pesos da camada seguinte e $z^{(l)}$ é a entrada ponderada na camada atual. Esse processo permite ajustar iterativamente os pesos, melhorando a capacidade da rede de mapear corretamente os dados de entrada para as saídas desejadas, minimizando o erro ao longo das iterações.

2.6.2 Breve histórico do desenvolvimento das CNNs

A primeira implementação prática de uma CNN foi realizada por Kunihiko Fukushima em 1980, com um modelo de rede neural que era capaz de reconhecer padrões visuais como letras escritas à mão (FUKUSHIMA, 1980). Embora o Neocognitron fosse um protótipo inicial e não treinado por aprendizado supervisionado, ele introduziu o conceito de camadas convolucionais e de *pooling* que são fundamentais nas CNNs modernas (FUKUSHIMA, 1980).

O grande avanço no desenvolvimento das CNNs ocorreu em 1989, com o trabalho de Yann LeCun, que aplicou o algoritmo de retropropagação para treinar

uma rede convolucional. Seu modelo, conhecido como LeNet, foi usado com sucesso para o reconhecimento de dígitos manuscritos em cheques bancários. LeNet marcava a transição para redes mais profundas e eficazes e introduziu muitos dos princípios que são usados nas CNNs contemporâneas, como a aplicação de convoluções para extração automática de características e a utilização de múltiplas camadas para aprendizado hierárquico de características.

O interesse por CNNs cresceu significativamente com a introdução de unidades de processamento gráfico no treinamento dessas redes, o que permitiu a experimentação com redes muito mais profundas e complexas. Em 2012, um modelo chamado AlexNet (BOTTOU, 2010), ganhou o desafio ImageNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) usando cinco camadas convolucionais e foi o primeiro a usar técnicas como *dropout* para evitar o sobreajuste, promovendo avanços significativos na eficiência e eficácia das CNNs.

Desde então, as CNNs continuaram a evoluir e se tornaram o padrão para muitas tarefas de visão computacional.

2.6.3 Princípios fundamentais das CNNs

Esta seção detalha os principais componentes da arquitetura de uma CNN, explicando como cada tipo de camada contribui para a capacidade de uma CNN de realizar tarefas complexas de reconhecimento visual.

O componente mais característico das CNNs é a camada convolucional, que como já abordado aplica um conjunto de filtros à imagem de entrada para produzir mapas de características. Cada filtro em uma camada convolucional é pequeno espacialmente (por exemplo, 3x3 ou 5x5 píxeis), mas se estende através da profundidade total da entrada, abrangendo todos os canais de cor (RGB, por exemplo). Conforme já descrito, quando um filtro desliza, realizando a convolução, sobre a imagem, ele produz um mapa bidimensional de respostas de ativação de neurônios, conhecido como mapa de características, que capta informações como bordas, texturas ou outros padrões visuais (LECUN; BENGIO; HINTON, 2015).

Matematicamente, a operação de convolução aplicada por um filtro (ou kernel) K de tamanho $k \times k$ à imagem de entrada I de tamanho $H \times W$ pode ser descrita (LECUN; BENGIO; HINTON, 2015) pela equação (7):

(7) $$(f * g)(i, j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} f(m, n) \cdot g(i - m, j - n)$$

em que f é o filtro, g é a imagem de entrada, e (i, j) são as coordenadas do pixel na imagem de saída. Na prática, o filtro é uma pequena matriz que desliza sobre a imagem de entrada, multiplicando e somando os valores correspondentes para produzir um valor no mapa de características de saída (LECUN; BENGIO; HINTON, 2015).

Além dos filtros, as camadas convolucionais possuem características como deslocamento (*stride*) e preenchimento (*padding*) (LECUN et al., 1989). O deslocamento S define o passo com que o filtro se move pela imagem. Se o deslocamento for 1, o filtro se move um pixel de cada vez, resultando em um mapa de características detalhadas. Se o deslocamento for maior que 1, o filtro pula píxeis, reduzindo a resolução do mapa de características. Formalmente, se o deslocamento S for considerado, a posição (i, j) na saída é calculada pela equação (8):

(8) $$(K * I)(i, j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} K(m, n) \cdot I(i \cdot S + m, j \cdot S + n)$$

O preenchimento adiciona bordas à imagem de entrada para controlar o tamanho da saída. No preenchimento '*valid*', a imagem de entrada é reduzida após cada operação de convolução, enquanto no preenchimento '*same*', é adicionado de forma que a saída tenha o mesmo tamanho que a entrada. O preenchimento P necessário para cada lado da entrada é calculado pela equação (9):

(9) $$P = \left\lfloor \frac{k - 1}{2} \right\rfloor$$

em que k é o tamanho do kernel.

Portanto, a convolução permite que a rede aprenda invariâncias locais, o que significa que ela pode reconhecer características independentemente de sua posição na imagem. Após a camada convolucional, geralmente segue-se uma camada de *pooling* (também chamada de *subsampling* ou *downsampling*), que reduz a dimensão espacial dos mapas de características, resultando em uma redução da quantidade de dados e de parâmetros a serem processados (LECUN; BENGIO; HINTON, 2015). O *pooling* ajuda a tornar as representações obtidas invariantes a pequenas translações e distorções na imagem de entrada. O tipo mais comum de *pooling* é o *max pooling*, que reduz o tamanho do mapa pegando o valor máximo de uma janela pequena (por exemplo, 2x2 píxeis) e descartando os outros valores.

Além das camadas convolucionais e de *pooling*, as CNNs frequentemente incorporam camadas de normalização, como a normalização por lotes (*batch normalization*). Essa técnica ajusta a ativação em uma camada para ter uma média zero e uma variação unitária. Isso ajuda a combater o problema conhecido como deslocamento de covariância interna (NAIR; HINTON, 2010).

Ioffe e Szegedy (2015) definiram o deslocamento de covariância interna como as mudanças nas distribuições das ativações das camadas internas durante o treinamento, um problema que pode dificultar a convergência dos modelos de redes neurais profundas. Para mitigar este problema, eles introduziram a técnica de normalização por lotes, que normaliza as ativações das camadas em mini-lotes de dados, estabilizando a distribuição das entradas em cada camada e, assim, acelerando o treinamento. A normalização por lotes permite usar taxas de aprendizado maiores e ser menos cuidadoso com a inicialização dos parâmetros, além de atuar como um regularizador, reduzindo levemente o sobreajuste.

Outra técnica comum para prevenir o sobreajuste é a inclusão de camadas de *dropout* nas CNNs (SRIVASTAVA et al., 2014). Durante o treinamento, a camada de *dropout* desliga aleatoriamente uma porção dos neurônios em uma camada, ou seja, suas ativações são setadas para zero. Isso força a rede a não depender excessivamente de qualquer neurônio individual e a aprender representações mais robustas que são úteis em conjunto com muitos diferentes subconjuntos aleatórios de outros neurônios (SRIVASTAVA et al., 2014). No teste, todos os neurônios são usados, mas suas saídas são ajustadas para compensar o desligamento aleatório.

As camadas de ativação são fundamentais em qualquer CNN, pois introduzem não-linearidades no modelo. A função de ativação mais comum nas CNNs modernas é a ReLU (Unidade Linear Retificada) (PRINCE, 2012), que é computacionalmente eficiente e reduz o problema do desaparecimento do gradiente (BENGIO; SIMARD; FRASCONI, 1994), permitindo que redes mais profundas sejam treinadas de forma mais eficaz.

Após várias camadas convolucionais e de *pooling*, é comum encontrar uma ou mais camadas densas (ou completamente conectadas). Essas camadas são projetadas para condensar a informação extraída pelas camadas anteriores em previsões finais (LECUN; BENGIO; HINTON, 2015).

Ainda existem outras camadas, menos faladas como as camadas de concatenação que são usadas em arquiteturas mais novas de CNN, como nas redes *inception*. Essas camadas combinam saídas de várias camadas anteriores, permitindo que a rede aprenda a combinar características em vários níveis de abstração. As camadas de convolução transpostas (RADFORD; METZ; CHINTALA, 2015) que são frequentemente usadas em modelos generativos, como as redes generativas adversárias (GANs) e nas redes de segmentação semântica. A camada de entrada que define como os dados de entrada são formatados e preparados para serem processados pelas camadas subsequentes. Algumas CNNs incluem camadas de *pooling* adaptativo (HE et al., 2015), que permitem que a rede ajuste dinamicamente a área de *pooling* para produzir saídas de tamanho fixo, independentemente do tamanho da entrada. Isso é especialmente útil em situações em que as imagens de entrada podem variar significativamente em tamanho ou quando é necessário conectar camadas convolucionais a camadas densas.

2.6.5 Vantagens das CNNs

Capacidade das CNNs de escalar com grandes quantidades de dados

Uma das vantagens das CNNs é sua habilidade de escalar com a quantidade de dados (LECUN; BENGIO; HINTON, 2015). Primeiro, os filtros (*kernels*) aplicados nas camadas convolucionais são compartilhados por toda a imagem, o que reduz significativamente o número de parâmetros que precisam ser aprendidos. A aplicação dos mesmos filtros em diferentes partes da imagem permite

que a rede aprenda a detectar características em qualquer posição, aumentando a robustez e a capacidade de generalização da rede. Isso permite que a rede seja treinada eficientemente, mesmo com grandes conjuntos de dados.

Além disso, o uso de Unidades de Processamento Gráfico (GPUs) é altamente eficiente para operações de matriz. Técnicas de paralelização, como dividir o treinamento entre múltiplas GPUs ou utilizar *clusters* de computadores, permitem que grandes volumes de dados sejam processados de forma eficiente.

Por fim, o Gradiente Estocástico com Mini-batches (SGD) (LECUN; BENGIO; HINTON, 2015) é uma técnica que permite que a rede aprenda de grandes volumes de dados em partes menores, tornando o treinamento mais eficiente e menos propenso a ficar preso em mínimos locais. Essas características combinadas explicam por que as CNNs são capazes de escalar tão bem com a quantidade de dados.

Extração automática e hierárquica de características

As CNNs aprendem automaticamente hierarquias de características diretamente dos dados (LECUN; BENGIO; HINTON, 2015), uma propriedade que outros métodos, baseados em engenharia de características (BAY; TUYTELAARS; VAN GOOL, 2006) ou métodos de classificação tradicionais, não possuem. Essa habilidade surge de várias características fundamentais das CNNs. Em primeiro lugar, os filtros convolucionais são aprendidos durante o processo de treinamento, permitindo que a rede descubra automaticamente as características mais relevantes dos dados de entrada (BOTTOU, 2010). As CNNs consistem em múltiplas camadas convolucionais, onde cada camada aprende características de nível superior a partir das características aprendidas pela camada anterior (LECUN; BENGIO; HINTON, 2015). Além disso, as camadas de pooling reduzem a dimensionalidade dos dados ao resumir características locais (LECUN; BENGIO; HINTON, 2015), permitindo que a rede mantenha informações importantes enquanto descarta detalhes menos relevantes. Durante o treinamento, a rede ajusta automaticamente os pesos dos filtros convolucionais através do algoritmo de retropropagação (*backpropagation)*, otimizando diretamente o desempenho na tarefa específica, como a classificação de imagens.

Adaptação e versatilidade

A adaptabilidade das CNNs a diferentes domínios é uma de suas características mais valiosas. Com ajustes na arquitetura, as CNNs têm sido aplicadas com sucesso em uma variedade de áreas. A capacidade de transferir aprendizado entre diferentes tarefas e domínios usando técnicas como transferência de aprendizado fortalece ainda mais sua utilidade e eficácia, diminuindo a necessidade de grandes conjuntos de dados rotulados para novas aplicações (LECUN; BENGIO; HINTON, 2015).

2.6.6 Desafios e limitações no uso de CNNs

Apesar das vantagens significativas das redes neurais convolucionais na visão computacional e outras áreas, elas também apresentam uma série de desafios e limitações que podem impactar sua aplicabilidade e eficácia em certos contextos.

Necessidade de grandes quantidades de dados

Um dos principais desafios no uso de CNNs é sua dependência de grandes quantidades de dados rotulados para treinamento. Enquanto as CNNs são excepcionais em aprender a partir de vastos conjuntos de dados, a coleta e a rotulação desses dados podem ser extremamente custosas e demoradas. Esse requisito de grandes *datasets* rotulados limita a aplicabilidade das CNNs em domínios em que os dados são escassos ou em que a rotulação é inviável devido a restrições de tempo ou de custo.

Sobrecarga computacional

As CNNs exigem uma quantidade significativa de recursos computacionais para treinamento e inferência, especialmente à medida que a profundidade e a complexidade da rede aumentam. Esse alto custo computacional pode ser proibitivo para aplicações que exigem resposta em tempo real em dispositivos com capacidade de processamento limitada, como dispositivos móveis ou embarcados.

Risco de sobreajuste

Devido à sua capacidade de modelar complexidades extremamente altas, as CNNs estão especialmente suscetíveis ao sobreajuste, especialmente quando

treinadas com conjunto de dados relativamente pequenos em relação ao número de parâmetros na rede. O sobreajuste ocorre quando uma rede aprende padrões específicos dos dados de treinamento que não são generalizáveis para novos dados, resultando em um desempenho pobre em conjuntos de dados não vistos anteriormente.

Transparência e interpretabilidade

Outra limitação importante das CNNs é a falta de transparência e interpretabilidade de suas operações internas. Devido à natureza complexa e multicamada, é muitas vezes difícil entender como exatamente as decisões são tomadas dentro da rede.

2.9 Algoritmos de otimização em aprendizado profundo

Papel dos algoritmos de otimização na aprendizagem de redes neurais

Conforme discutido anteriormente, algoritmos de otimização ajustam os pesos das conexões na rede para minimizar uma função de perda, que mede o quão bem a rede está performando em relação ao seu objetivo de treinamento (LECUN; BENGIO; HINTON, 2015).

O processo de otimização envolve a atualização contínua dos pesos da rede com base em como eles afetam a função de perda, refinando-os ao longo de várias iterações. Essa abordagem iterativa permite que a rede aprenda progressivamente a mapear os dados de entrada para as saídas corretas. A forma como os pesos são ajustados influencia a rapidez com que o modelo converge para uma solução e a qualidade final do modelo treinado.

Além de métodos básicos de otimização, existem variações que buscam melhorar a eficiência do treinamento. Essas variações incluem algoritmos que incorporam ajustes adicionais, como *momentum* e *Nesterov Accelerated Gradient*, que ajudam a suavizar as atualizações e acelerar a convergência. Otimizadores adaptativos, como *Adam*, *RMSprop* e *Adagrad*, introduzem a capacidade de ajustar a taxa de aprendizado individualmente para cada peso, melhorando o desempenho em cenários complexos.

Além das variações mencionadas, algumas abordagens combinam diferentes métodos para maximizar os benefícios de cada técnica, aplicando termos de regularização para evitar sobreajuste ou ajustando dinamicamente os parâmetros de treinamento.

Ajuste de hiperparâmetros e sua influência no treinamento de modelos

O ajuste de hiperparâmetros é outro aspecto crítico na otimização de redes neurais. Hiperparâmetros, como a taxa de aprendizado, o número de camadas na rede, o número de unidades por camada e o tipo de função de ativação, têm um impacto significativo na performance do modelo. A escolha adequada desses parâmetros pode significar a diferença entre um modelo que aprende eficientemente e um que falha em convergir ou generalizar a partir de dados novos.

Técnicas de ajuste de hiperparâmetros, como a busca em grade, busca aleatória (BERGSTRA; BENGIO, 2012) e otimização bayesiana (SNOEK; LAROCHELLE; ADAMS, 2012), são utilizadas para explorar o espaço de hiperparâmetros de forma sistemática e identificar as configurações que produzem os melhores resultados no conjunto de validação.

Os algoritmos de otimização e o ajuste de hiperparâmetros são, portanto, componentes críticos do processo de aprendizado, essenciais para desenvolver modelos eficientes, rápidos e capazes de generalizar bem a partir de conjuntos complexos de dados (LECUN et al., 1989; LECUN; BENGIO; HINTON, 2015).

2.10 Técnicas de regularização para reduzir o sobreajuste

As técnicas de regularização são importantes no treinamento de modelos para prevenir o sobreajuste. O *dropout*, camada que abordamos em 2.6.3, é uma dessas técnicas. Durante o treinamento, o *dropout* desliga aleatoriamente um subconjunto de neurônios em cada iteração, significando que esses neurônios não participam da propagação para frente nem do processo de retropropagação. Isso impede que os neurônios co-adaptem excessivamente seus pesos uns aos outros, forçando a rede a aprender representações mais robustas e redundantes, pois não pode contar com a presença de qualquer neurônio específico. Como resultado, o

modelo se torna menos sensível a pequenas flutuações nos dados de entrada e melhora na generalização para novos exemplos.

Outra técnica é a parada precoce (*Early Stopping*) que é uma abordagem que envolve monitorar o desempenho do modelo durante o treinamento e parar o treinamento assim que o desempenho em um conjunto de validação começa a piorar, apesar de melhorias contínuas no conjunto de treinamento. Esta técnica é eficaz porque interrompe o processo de aprendizado antes que o modelo tenha a chance de se ajustar demais aos dados de treinamento (PRECHELT, 2002). O *early stopping* também tem o benefício de reduzir o tempo de treinamento, pois o modelo não continua a aprender além do necessário.

A regularização L1 e L2 são técnicas utilizadas para modificar a função de custo que o modelo otimiza, adicionando termos de penalidade que ajudam a controlar a complexidade do modelo e reduzir o risco de sobreajuste. Na regularização L2, também conhecida como *ridge regression*, adiciona-se à função de custo um termo de penalidade proporcional à soma dos quadrados dos pesos do modelo, conforme mostrado pela equação (10):

(10) $$L_{L2} = L + \lambda \sum_i w_i^2$$

Essa abordagem incentiva o modelo a manter os pesos pequenos, o que pode resultar em um modelo mais simples e com maior capacidade de generalização. Por outro lado, a regularização L1, ou *lasso regression*, adiciona um termo de penalidade proporcional à soma dos valores absolutos dos pesos, conforme descrito pela equação (11):

(11) $$L_{L1} = L + \lambda \sum_i |w_i|$$

Uma característica notável da regularização L1 é que ela pode levar a alguns pesos sendo reduzidos exatamente a zero, o que equivale a uma seleção

automática de características, mantendo apenas aquelas que são mais relevantes para o modelo. Essas técnicas ajudam a encontrar um equilíbrio entre a precisão do modelo e a complexidade, promovendo uma solução mais robusta para novos dados.

3. METODOLOGIA

Neste capítulo, apresentamos a metodologia adotada para o desenvolvimento e validação de um modelo de classificação utilizando redes neurais convolucionais, com o objetivo de identificar a ação de coçar em cães a partir de imagens extraídas de vídeos.

Inicialmente, a preparação do conjunto de dados foi realizada a partir de vídeos que registravam os cães exibindo uma variedade de comportamentos. Esses vídeos foram anotados com precisão e posteriormente segmentados para isolar os trechos contendo as ações específicas. Em seguida, aplicou-se a detecção das caixas delimitadoras e a geração das imagens correspondentes, com base nas áreas de interesse dos vídeos recortados.

Em seguida, realizamos o pré-processamento e o aumento do conjunto de dados, aplicando técnicas para melhorar a qualidade das imagens e aumentar a diversidade dos exemplos disponíveis para o treinamento. Após essa etapa, definimos a arquitetura da CNN, escolhendo as camadas e os parâmetros da rede visando maximizar o desempenho do modelo. E durante o treinamento, o modelo foi ajustado de forma iterativa, visando refinar continuamente sua capacidade de distinguir entre as classes coçando e não coçando.

Para avaliar o desempenho do modelo, isto é, determinar se sua performance é satisfatória e estimar os erros, utilizamos múltiplas métricas, cada uma focando em diferentes aspectos das habilidades do modelo. Entre elas, destacam-se a matriz de confusão, que permite visualizar se o modelo tende a favorecer alguma classe específica, e a acurácia, a métrica mais comum, que indica a porcentagem de acertos nas previsões realizadas.

3.1 Criação do conjunto de dados

A primeira etapa na criação do conjunto de dados envolveu a obtenção de vídeos[Figura 5] de longa duração, com aproximadamente 4 a 5 horas de gravação do comportamento dos cachorros, e uma tabela detalhada de anotações. Esta tabela incluía anotações de ações de coceira, com os tempos de início e fim de cada

ocorrência. O vídeo, originalmente no formato .avi, foi utilizado para a extração de imagens que seriam classificadas posteriormente.

Figura 5: *Frames* de vídeos mostrando o comportamento dos cachorros.

Fonte: elaborado pelo autor, 2024.

Tanto o vídeo original quanto a tabela de ações anotadas foram fornecidos por uma multinacional de saúde animal.

3.1.1 Ajuste dos tempos de eventos

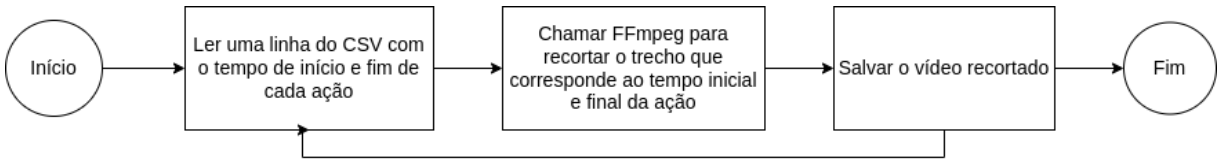
Para assegurar a precisão na identificação das ações de conceira, foi necessário ajustar os tempos registrados na tabela de eventos. Esse ajuste foi essencial porque o tempo de duração do vídeo era diferente do tempo exibido na câmera, que mostrava a data e a hora da gravação. Para corrigir essa discrepância, subtraímos a data e hora exibidas no vídeo pelo tempo real de duração do vídeo. Os tempos corrigidos foram então registrados em um arquivo CSV, que serviu como referência para os recortes subsequentes.

3.1.2 Recorte dos vídeos

Com os tempos ajustados, criamos automatizamos a tarefa de usar o FFmpeg (TOMAR, 2006) para recortar temporalmente múltiplos vídeos a partir do vídeo original. Cada vídeo recortado corresponde a uma ação de coceira especificada na tabela de eventos.

O script começa lendo o arquivo CSV que contém os tempos ajustados de início e fim das ações de coceira. O caminho para o vídeo original é especificado e o script então itera sobre cada linha do CSV, executando o FFmpeg para recortar o vídeo original nos intervalos de tempo especificados, conforme mostrado na figura 6.

Figura 6: Fluxograma do processo de recorte de vídeos.



Fonte: elaborado pelo autor, 2024.

Cada clipe gerado é salvo com um nome distinto que indica sua sequência e permite ao usuário entender de onde aquele recorte vem. Além disso, todos os clipes são organizados em pastas específicas para facilitar o processamento posterior.

O uso do FFmpeg permite a cópia exata dos segmentos de vídeo preservando a qualidade original. Ao final de cada iteração, o script gera uma mensagem de confirmação indicando que o vídeo foi cortado com sucesso, mencionando o intervalo de tempo correspondente. Essa abordagem garantiu que todas as ações de coceira fossem isoladas em clipes individuais de forma precisa e organizada.

3.1.4 Geração de imagens das caixas delimitadoras

As caixas delimitadoras (*bounding box*) são retângulos que delimitam uma área específica de uma imagem ou quadro de vídeo, usados para indicar a localização de objetos de interesse, como um cão, em um cenário mais amplo. Matematicamente, um *bounding box* é definido por um conjunto de coordenadas que formam um retângulo ao redor do objeto de interesse. Normalmente, essas coordenadas são especificadas em termos do ponto superior esquerdo e do ponto inferior direito.

As caixas delimitadoras são importantes porque permitem ao modelo da CNN focar apenas na área relevante da imagem. Isso é crucial para o desempenho do modelo, pois os dados de entrada são filtrados para conter apenas as informações necessárias, reduzindo o impacto de ruídos ou elementos de fundo que não

contribuem para a classificação. Ao delimitar o objeto de interesse, o modelo pode extrair características visuais específicas que são essenciais para determinar se ele está se coçando ou não.

Se utilizássemos a imagem inteira, a CNN precisaria lidar com muitos píxeis que não são relevantes para a tarefa de classificação. Isso aumentaria a complexidade do problema e poderia prejudicar o desempenho do modelo, uma vez que a rede estaria aprendendo padrões não relacionados ao comportamento do cão. A aplicação das caixas delimitadoras reduz essa complexidade ao limitar o espaço de busca, tornando o aprendizado mais eficiente.

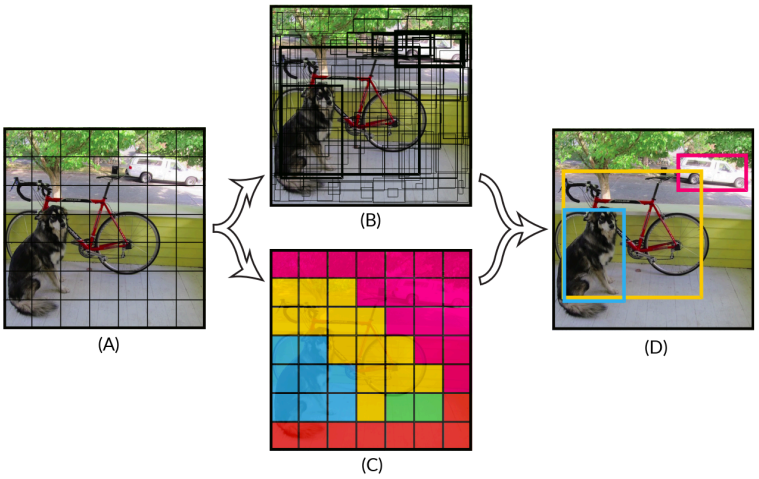
Além disso, as caixas delimitadoras são fundamentais para o rastreamento do cão nos vídeos, pois permitem acompanhar sua posição ao longo do tempo. Nos vídeos, as caixas delimitadoras são geradas quadro a quadro, formando uma sequência temporal que representa o movimento do cão. Portanto, matematicamente, as caixas delimitadoras servem como uma forma de simplificação espacial para as redes neurais convolucionais, reduzindo a dimensionalidade do problema e otimizando o processo de extração de características relevantes. A precisão das caixas delimitadoras é muito importante para garantir que o modelo seja treinado com dados de qualidade, que representem fielmente o comportamento de interesse, melhorando assim a capacidade da rede de generalizar para novos dados.

Dessa forma, após o recorte temporal dos vídeos, automatizamos o processo de produção das imagens a partir de recortes espaciais nos vídeos usando caixas delimitadoras. Este processo foi aplicado a todos os vídeos presentes na pasta raiz, facilitando o rastreamento e a extração de imagens de interesse. Para realizar essa detecção, aplicamos o modelo YOLO, uma rede neural convolucional pré-treinada que é muito utilizada para tarefas de detecção de objetos.

Para identificar e rastrear o cão nos *frames*, a YOLO divide a imagem em uma grade e aplica uma rede neural convolucional para prever simultaneamente as coordenadas das caixas delimitadoras e as probabilidades das classes associadas a esses objetos. Cada célula da grade é responsável por prever múltiplas caixas delimitadoras, além das respectivas probabilidades de o objeto estar presente na área delimitada. A YOLO então seleciona as caixas delimitadoras com as maiores

pontuações, filtrando as detecções redundantes com o uso de uma técnica chamada Non-Maximum Suppression (NMS), que retém apenas as detecções mais confiáveis.

Figura 7: Processo de Detecção e Classificação com YOLO - Grade, *Bounding Boxes* e Probabilidades de Classe.



Fonte: Redmon et al. (2016).

A imagem acima[figura 7] mostra as etapas do processo de detecção realizado pelo modelo YOLO. As etapas incluem a divisão da imagem em uma grade[figura 7, A], a previsão de *bounding boxes* com níveis de confiança[figura 7, B], a geração de um mapa de probabilidades das classes[figura 7, C] e a aplicação de Non-Maximum Suppression (NMS) para obter as detecções finais [figura 7, D].

Para o rastreamento do cão ao longo dos quadros, o processo envolve associar as detecções feitas pelo YOLO em quadros consecutivos para garantir que o mesmo objeto seja identificado ao longo do tempo. Essa associação é feita por algoritmos de *tracking*, que avaliam a similaridade entre os *bounding boxes* gerados em quadros consecutivos e tentam encontrar a melhor correspondência entre eles. O rastreamento consiste em minimizar uma função de custo que mede a diferença entre as posições dos *bounding boxes* em quadros consecutivos, levando em consideração a proximidade espacial, a consistência no tamanho dos retângulos e, em alguns casos, a semelhança visual do objeto detectado.

Os algoritmos específicos, como BoTSORT, DeepOCSORT, OCSort, HybridSORT, BYTETracker ou StrongSORT, combinam esses fatores para criar uma

matriz de custo que representa as associações possíveis entre os *bounding boxes* dos quadros consecutivos. O problema é então resolvido como uma tarefa de correspondência ótima, frequentemente usando algoritmos de otimização, como o algoritmo de Kuhn-Munkres, para encontrar a correspondência que minimize o custo total. Esse processo de rastreamento garante que o sistema consiga identificar consistentemente o cão ao longo dos quadros, gerando uma sequência contínua de detecções.

Além disso, para cada detecção, é extraída a região delimitada pela *bounding box*, a imagem é salva com um nome que indica sua origem e sequência, e organiza-se essas imagens em pastas específicas baseadas na identificação dos objetos rastreados. Este processo garante que todas as detecções sejam armazenadas de maneira estruturada, facilitando a análise e classificação subsequente.

A abordagem adotada permitiu a geração eficiente de um grande número de imagens, que são fundamentais para o treinamento e validação do modelo CNN. A estrutura organizada das pastas e a precisão na extração das imagens asseguraram um conjunto de dados robusto e confiável para as etapas seguintes do projeto.

3.1.5 Organização e filtragem das imagens

Após a geração das imagens das caixas delimitadoras, o próximo passo envolveu a organização e filtragem dessas imagens para criar conjuntos de dados específicos de coceira e não-coceira. Esta etapa foi fundamental para garantir que o conjunto de dados final estivesse bem organizado e classificado com precisão para o treinamento do modelo.

Primeiramente, criamos duas pastas distintas denominadas *Scratching* e *No_scratching*. Pastas essas destinadas a armazenar, respectivamente, as imagens em que os cachorros estavam se coçando e as imagens em que não estavam. Em seguida, realizamos uma revisão cuidadosa das imagens em que o foco principal foi identificar visualmente as ações de se coçar em que o cachorro estava claramente se coçando e aparecia sozinho no rastreamento. Apenas essas imagens foram copiadas para a pasta *Scratching*. Da mesma forma, as imagens em que o cachorro não estava se coçando e aparecia sozinho foram copiadas para a pasta

No_scratching. Esse processo seletivo garantiu que somente as imagens apropriadas fossem incluídas em cada categoria.

Para garantir a qualidade dos dados, adotamos uma abordagem rigorosa na gestão de dúvidas e exceções. Se houvesse qualquer dúvida sobre o conteúdo de uma imagem, como incerteza se o cachorro estava se coçando ou se a imagem estava muito misturada, optamos por não utilizá-la. Imagens que cortavam mais de um quinto do corpo do cachorro também foram excluídas. Além disso, mantivemos os nomes originais das imagens para preservar a rastreabilidade dentro do processo. O processo de seleção e organização foi repetido para cada subpasta dentro da *output*.

Com as pastas *Scratching* e *No_scratching* preenchidas com as imagens apropriadas, a etapa final envolveu a construção do conjunto de dados (Dataset) definitivo, que seria utilizado para treinamento e validação do modelo.

Para a criação do conjunto de dados final, as imagens das pastas *Scratching* e *No_scratching* foram copiadas para um diretório temporário, preservando a organização inicial. Na primeira fase, o programa utilizou uma biblioteca especializada para dividir as imagens em conjuntos de treinamento (train) e validação (val), com uma proporção de 80% para treinamento e 20% para validação, de forma estratificada. Essa divisão estratificada garantiu que a proporção das classes *Scratching* e *No_scratching* fosse mantida em ambos os conjuntos, assegurando que o modelo recebesse uma amostra representativa para o aprendizado e uma base equilibrada para avaliar seu desempenho.

No entanto, após essa divisão inicial, observamos que muitas imagens parecidas estavam presentes tanto nos conjuntos de treino quanto nos de validação. Isso ocorreu porque as imagens foram extraídas de sequências de quadros de vídeos, resultando em várias imagens consecutivas com pouca variação sendo distribuídas entre os dois conjuntos. Isso poderia comprometer a capacidade do modelo de generalizar adequadamente. Para resolver esse problema, decidimos separar as imagens com base nas câmeras de origem. Especificamente, todas as imagens de determinadas câmeras foram alocadas exclusivamente para o conjunto de treinamento, enquanto as imagens de outras câmeras foram destinadas ao conjunto de validação. Essa abordagem garantiu que as imagens usadas para

validação fossem significativamente diferentes das usadas para treinamento, melhorando a robustez e a eficácia da validação do modelo.

O resultado final foi um conjunto de dados estruturado em diretórios *train* e *val*, cada um contendo pastas para as classes *Scratching* e *No_scratching*. Conforme explicado anteriormente, esse processo foi totalmente automatizado e não apenas garantiu a integridade e a organização dos dados, mas também preparou um conjunto de dados robusto e bem estruturado, essencial para o sucesso na etapa subsequente de desenvolvimento do modelo. O cuidado meticuloso na organização do dataset final é um passo fundamental para a obtenção de resultados precisos e confiáveis no treinamento de modelos de aprendizado de máquina.

3.2 Classificação

3.2.1 Pré-processamento dos dados

Os principais métodos usados no pré-processamento de imagens são redimensionamento, normalização, técnicas de aumento de dados (*data augmentation*), remoção de ruído, equalização de histograma, transformação de cor, segmentação, detecção de bordas, remoção de fundo e centralização. O redimensionamento ajusta o tamanho das imagens para dimensões uniformes, a normalização escala os valores dos píxeis para um intervalo padrão, e o *data augmentation* aumenta a variabilidade dos dados aplicando transformações aleatórias.

Em nosso caso, utilizamos apenas redimensionamento, normalização e *data augmentation*. É importante destacar que equalização de histograma e transformação de cor não foram usados, pois não mostraram melhora no aprendizado do modelo.

Utilizando a classe *ImageDataGenerator* da biblioteca *TensorFlow*, a normalização dos píxeis (LECUN; BENGIO; HINTON, 2015) foi aplicada para garantir que os valores dos píxeis das imagens estivessem na mesma escala. Especificamente, os valores dos píxeis foram escalados para o intervalo de 0 a 1, dividindo cada valor de pixel por 255. A normalização foi importante pois evita que

valores de píxeis muito altos causem instabilidades nos cálculos das ativações das redes neurais.

Aplicamos também uma rotação aleatória às imagens para ajudar o modelo a se tornar invariante à orientação dos objetos. A rotação aleatória permite que o modelo aprenda a identificar o comportamento em diferentes ângulos, melhorando sua capacidade de generalização.

A inversão horizontal (flip horizontal) e a inversão vertical (flip vertical) também foram utilizadas para aumentar a variabilidade dos dados de treinamento. A inversão horizontal gira a imagem horizontalmente, como se estivesse refletida em um espelho vertical, enquanto a inversão vertical gira a imagem verticalmente, como se estivesse refletida em um espelho horizontal. Essas transformações ajudam o modelo a lidar com diferentes orientações do objeto.

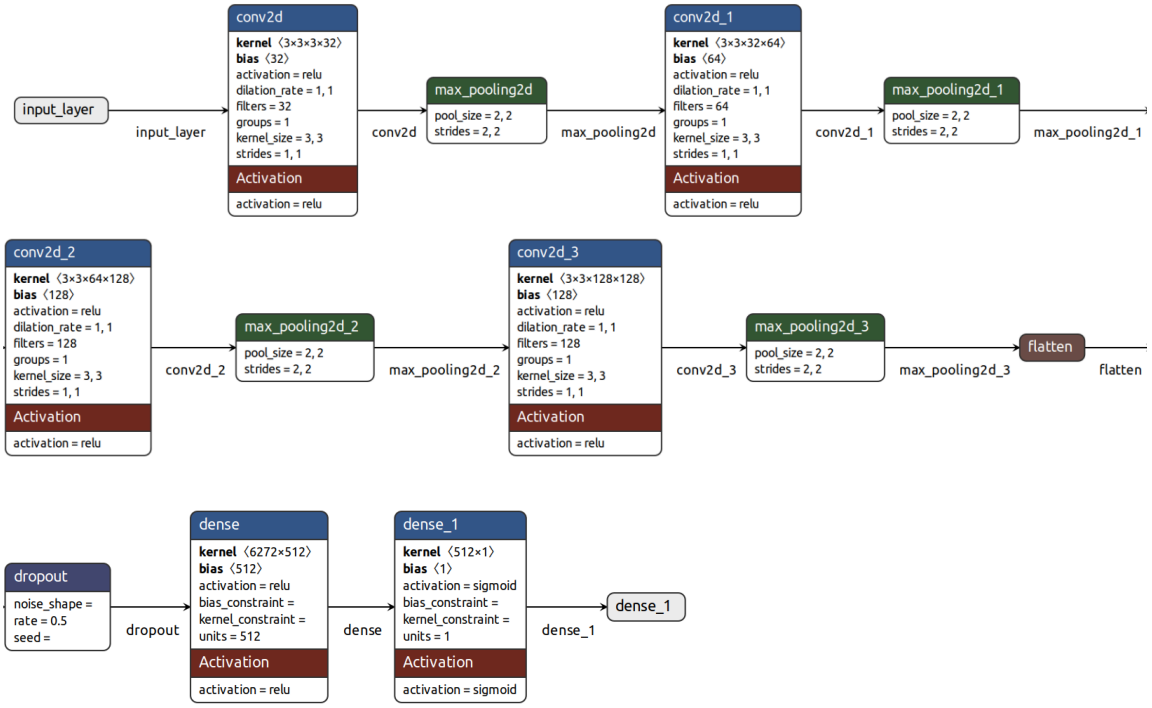
Para lidar com píxeis ausentes que surgem após a aplicação das transformações, utilizamos um método de preenchimento conhecido como '*nearest*'. Este método é importante para evitar artefatos indesejados nas bordas das imagens transformadas, que poderiam confundir o modelo durante o treinamento. Ao usar os valores dos píxeis adjacentes mais próximos, garantimos uma transição suave e natural nas áreas transformadas da imagem, preservando a integridade visual e a qualidade dos dados.

Essas transformações ajudaram a aumentar a variabilidade dos dados, permitindo que o modelo generalize melhor para novas imagens que possam apresentar variações similares.

3.2.2 Definição da arquitetura do modelo

Nesta seção, definiremos a arquitetura do modelo de CNN que foi construído utilizando a biblioteca *TensorFlow*. Nosso modelo foi projetado para processar imagens de entrada com dimensões de 150x150 píxeis e três canais de cor (RGB).

Figura 8: Arquitetura da CNN para Detecção de 'coceira' em Cães.



Fonte: elaborado pelo autor, 2024.

O modelo iniciou com várias camadas convolucionais. As camadas convolucionais são fundamentais para a extração de características, pois aplicam filtros sobre a imagem de entrada para detectar padrões como bordas, texturas e formas. A primeira camada convolucional utilizou 32 filtros de tamanho 3x3, com a função de ativação de unidade linear retificada (*ReLU — Rectified Linear Unit*), que introduz não linearidades no modelo. Após a primeira camada convolucional, foi adicionada uma camada de *pooling* (MaxPooling2D) para reduzir a dimensionalidade espacial da imagem e controlar o *overfitting*, extraindo as características mais importantes.

O processo de convolução e *pooling* foi repetido com um número crescente de filtros. Especificamente, utilizamos camadas convolucionais adicionais com 64 e 128 filtros, cada uma seguida por uma camada de *pooling*. Cada camada convolucional e de *pooling* foi cuidadosamente configurada para garantir que as características essenciais das imagens fossem extraídas de maneira eficiente, mantendo a complexidade do modelo gerenciável.

Para prevenir o sobreajuste, introduzimos uma camada de *dropout* após as camadas convolucionais e de *pooling*. A camada de *dropout* desativa aleatoriamente uma fração dos neurônios durante o treinamento, forçando o modelo a aprender representações redundantes e mais robustas das características das imagens. Neste caso, a fração de *dropout* foi definida em 0.5, ou seja, desativando metade dos neurônios em cada iteração do treinamento.

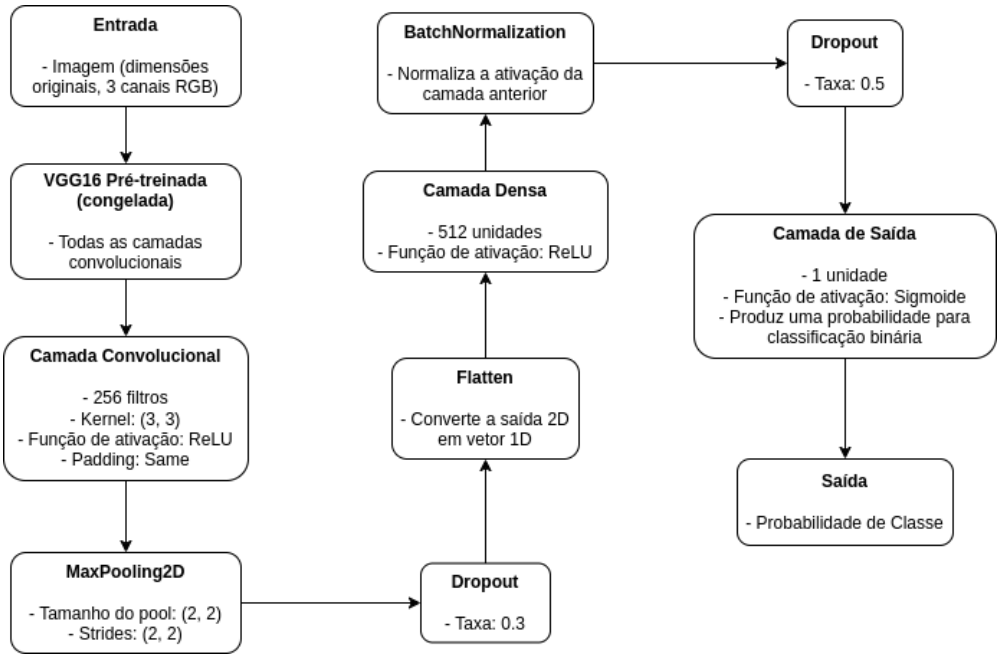
Finalmente, adicionamos camadas densamente conectadas (fully connected layers) para a classificação final. Essas camadas são responsáveis por combinar as características extraídas pelas camadas convolucionais e realizar a predição. A primeira camada densa tinha 512 neurônios com a função de ativação ReLU, proporcionando uma combinação não linear das características extraídas. A última camada do modelo era uma camada densa com um único neurônio e uma função de ativação sigmoid.

Esta configuração permite prever a probabilidade de um evento de *Scratching*, gerando uma saída entre 0 e 1, onde valores próximos de 1 indicam alta probabilidade de *Scratching* e valores próximos de 0 indicam baixa probabilidade.

Para obter a melhor performance do modelo, realizamos experimentos com diversas configurações de arquitetura, ajustando o número de camadas convolucionais, filtros, e camadas densas. Essas variações foram testadas para identificar a combinação que proporciona a melhor capacidade de generalização e acurácia na tarefa de classificação de *Scratching*. A avaliação de cada configuração envolveu a análise das métricas de desempenho, como a acurácia e a perda tanto no conjunto de treinamento quanto no de validação, permitindo identificar quais ajustes contribuem para a melhoria do modelo.

Além da arquitetura mostrada, também experimentamos a utilização de modelos de transferência de aprendizado (transfer learning), utilizando a arquitetura VGG16 pré-treinada. O VGG16 é uma rede neural convolucional profunda, que já foi treinada em um grande conjunto de dados de imagens (ImageNet) e, portanto, possui uma capacidade robusta de extração de características. Ao utilizar VGG16, removemos as camadas de classificação superiores e adicionamos novas camadas densas e uma camada de saída.

Figura 9: Arquitetura da CNN Baseada no VGG16 para Classificação.



Fonte: elaborado pelo autor, 2024.

As camadas adicionadas incluíram uma convolucional com 256 filtros, *kernel* (3, 3) e ativação ReLU, seguida por *MaxPooling2D* e *dropout* de 0.3. Após o achatamento (*flattening*), uma camada densa com 512 unidades e ativação ReLU foi adicionada, acompanhada de *BatchNormalization* e *dropout* de 0.5. A camada final consistiu em uma unidade densa com ativação sigmoide para a classificação binária. Este processo permite que o modelo aproveite as características previamente aprendidas, adaptando-se rapidamente ao nosso conjunto de dados específico.

No entanto, os experimentos com *VGG16* não resultaram em uma melhora significativa na acurácia final, conforme mostrado no capítulo de resultados. Embora a utilização de *VGG16* tenha acelerado a convergência do modelo durante o treinamento inicial, a performance em termos de acurácia final não superou a arquitetura mostrada na figura 9.

3.2.3 Compilação e treinamento do modelo

A próxima etapa na construção do modelo envolveu a compilação e o treinamento. Na compilação do modelo, utilizamos a função de perda *binary_crossentropy*, que é particularmente adequada para problemas de classificação binária, como o nosso, em que o objetivo é classificar as imagens entre

coceira e não-coceira. A escolha desta função de perda permite que o modelo calcule a discrepância entre as predições e os valores reais, ajustando seus pesos para minimizar essa diferença.

Para otimizar o processo de aprendizado, selecionamos o otimizador Adam, que é utilizado devido à sua eficiência e capacidade de ajuste adaptativo das taxas de aprendizado. Configuramos o Adam com uma taxa de aprendizado de 0.0001, um valor escolhido para balancear a velocidade de convergência e a estabilidade do treinamento. Uma taxa de aprendizado muito alta pode causar oscilações e impedir que o modelo alcance uma boa solução, enquanto uma taxa muito baixa pode tornar o processo de treinamento excessivamente lento. A taxa de 0.0001 foi encontrada como um ponto ideal através de experimentação.

O treinamento do modelo foi realizado ao longo de 50 épocas, e alimentamos o modelo com lotes de imagens processadas e aumentadas em tempo real. Durante o treinamento, o modelo ajustou seus parâmetros internamente para minimizar a função de perda. A cada época, o modelo foi avaliado tanto no conjunto de treinamento quanto no conjunto de validação, permitindo um monitoramento constante do progresso e ajudando a identificar qualquer sinal de sobreajuste ou subajuste. Além disso, utilizamos o *early stopping* para interromper o treinamento caso não houvesse melhora significativa na métrica de validação por várias épocas consecutivas, evitando treinamento excessivo e melhorando a capacidade de generalização do modelo.

3.2.4 Avaliação do modelo

As métricas usadas – acurácia, precisão, sensibilidade, *F1-Score* e especificidade – permitem avaliar diferentes aspectos do desempenho do modelo de classificação de *scratching*. Enquanto a acurácia fornece uma visão geral da taxa de acertos, a precisão foca na qualidade das previsões de *scratching* feitas. A sensibilidade avalia a capacidade do modelo em detectar corretamente os casos de *scratching*, e a especificidade mede a habilidade em identificar corretamente os casos de no *scratching*. O *F1-Score*, por sua vez, oferece um equilíbrio entre precisão e sensibilidade, sendo útil para lidar com possíveis desbalanceamentos entre as classes.

Também geramos a matriz confusão que permite identificar a quantidade de verdadeiros positivos (quando o modelo prevê corretamente um evento de *scratching*), verdadeiros negativos (quando o modelo prevê corretamente a ausência de *scratching*), falsos positivos (quando o modelo prevê *scratching* incorretamente) e falsos negativos (quando o modelo não detecta *scratching* que realmente ocorreu).

Além da análise de desempenho, organizamos as imagens de validação com base nas predições do modelo. As imagens foram copiadas para diretórios correspondentes a verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN). Este sistema de organização facilitou a análise visual dos erros de classificação, permitindo identificar padrões ou características específicas que poderiam estar contribuindo para esses erros.

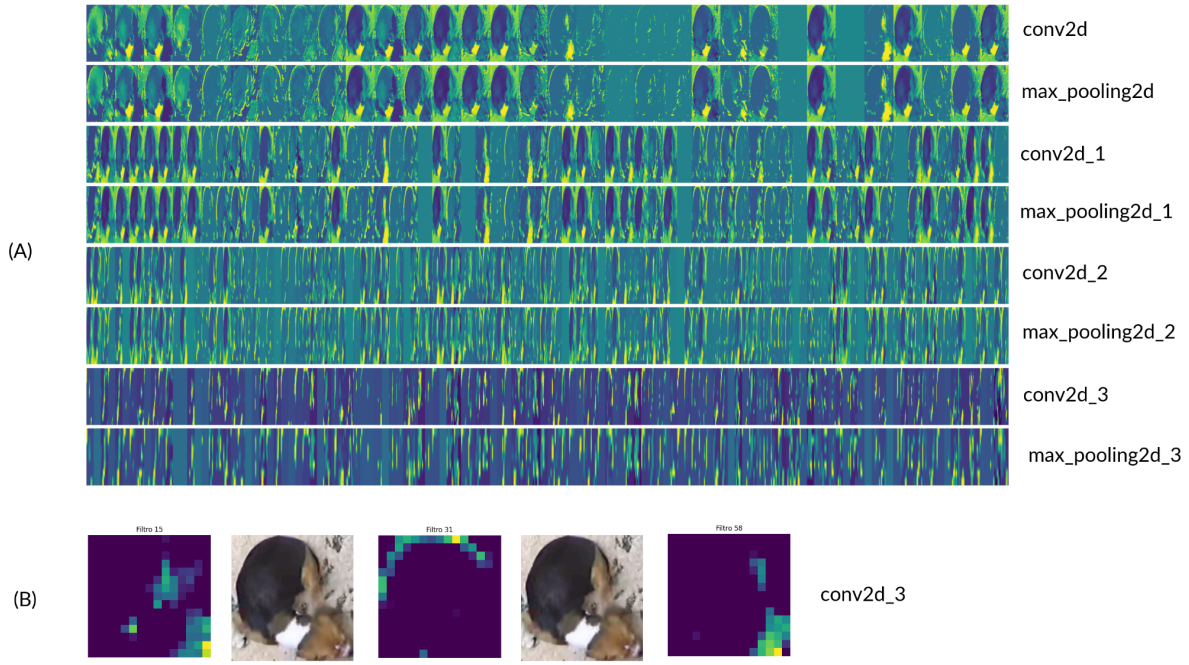
4. RESULTADOS E DISCUSSÕES

Neste capítulo, apresentamos e discutimos os resultados obtidos durante o desenvolvimento do modelo de identificação da ação de coçar em cães utilizando CNNs. Avaliamos a eficácia do modelo com base em diferentes métricas de desempenho, incluindo acurácia, precisão, sensibilidade e especificidade. Além disso, abordamos os desafios encontrados ao longo do desenvolvimento do modelo, como a quantidade limitada de dados anotados disponíveis para treinamento. Para enfrentar essas limitações, foram implementadas estratégias como o uso de técnicas de aumento de dados para aumentar a variabilidade dos dados e técnicas de regularização, como *dropout* e Regularização L2, para reduzir o sobreajuste e melhorar a capacidade de generalização do modelo.

4.1 CNN com aumento de dados, ReLU, regularização L2 e *dropout*: CNN-DATAUG-RELU-L2-DROPOUT

A arquitetura do modelo CNN-DATAUG-RELU-L2-DROPOUT foi projetada com várias camadas convolucionais. Conforme descrito anteriormente, a camada de entrada recebeu imagens de 150x150 píxeis com 3 canais de cor (RGB). O uso de múltiplas camadas convolucionais com aumento progressivo no número de filtros melhorou a capacidade do modelo de capturar características relevantes, como mostrado na figura 8.

Figura 10 - Mapa de ativação do modelo CNN-DATAUG-RELU-L2-DROPOUT mostra a saída das camadas convolucionais e de *pooling*.



Fonte: elaborado pelo autor, 2024.

A figura 10 (A) mostra mapas de ativação que é uma visualização das saídas de uma camada convolucional em uma rede neural. Ele mostra quais regiões da imagem de entrada ativam mais os filtros da camada, ou seja, onde o modelo presta mais atenção para identificar padrões específicos. Essas ativações refletem os padrões que a rede está usando para tomar decisões em tarefas como classificação de imagens ou detecção de objetos.

Na figura 10, os mapas de ativação das camadas convolucionais mostram a progressão das características extraídas ao longo das camadas. À medida que avançamos para camadas convolucionais mais profundas, os mapas de ativação se tornam menos detalhados e mais abstratos, capturando informações de nível superior sobre a imagem.

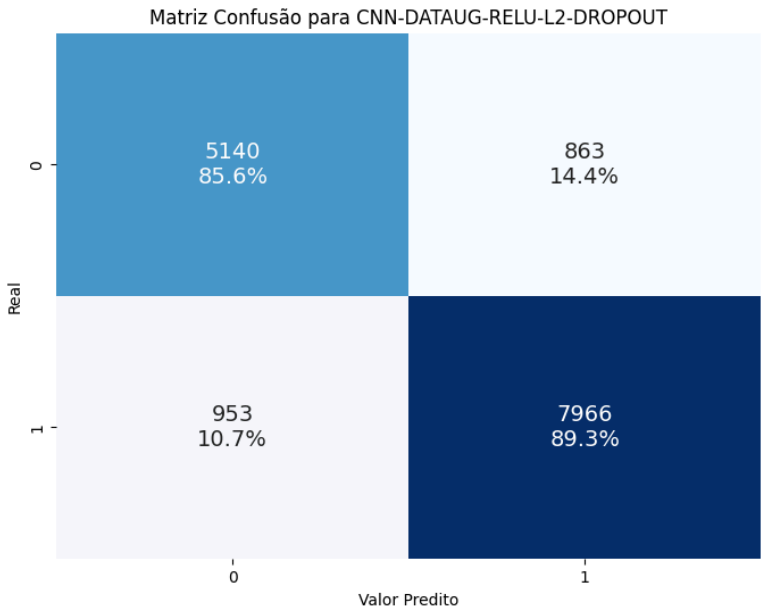
Para o caso da ação de coceira em cães, observa-se que os filtros nas camadas convolucionais mais profundas, como mostrado na figura 10 (B), capturam a posição das patas, a postura corporal e a posição da cabeça. Esses elementos são componentes que indicam o movimento característico de coçar. Enquanto as camadas iniciais focam em características básicas, como bordas e texturas, as

camadas mais profundas conseguem integrar essas informações para identificar padrões mais complexos e significativos. Isso mostra que o modelo aprendeu a reconhecer características específicas que são importantes para diferenciar esse comportamento de outros movimentos.

Após a etapa de convolução, as camadas densas assumiram o papel de realizar a classificação final das imagens. A matriz 2D gerada pelas camadas convolucionais foi convertida em um vetor 1D por meio de uma operação *Flatten*, facilitando o processamento pelos neurônios das camadas densas. A primeira camada densa continha 512 neurônios, utilizando a função de ativação de unidade linear retificada para permitir a modelagem de relações não lineares entre as características extraídas. Além disso, a regularização L2 foi aplicada para reduzir o risco de overfitting. Para aumentar ainda mais a robustez do modelo, foi inserida uma camada de *dropout* com taxa de 0,5, que ajuda a evitar a dependência excessiva de neurônios específicos durante o treinamento. Por fim, a camada de saída era composta por um único neurônio com função de ativação sigmoide.

Essa configuração permitiu que o modelo CNN-DATAUG-RELU-L2-DROPOUT gerasse uma saída binária, indicando se a ação de coçar estava presente ou não na imagem analisada. O modelo foi configurado para utilizar a função de perda chamada de entropia cruzada binária, que é apropriada para problemas de classificação com duas categorias, e o otimizador *Adam*, com uma taxa de aprendizado de 0,0001, escolhido por sua eficiência em ajustar os parâmetros durante o treinamento. Para evitar o sobreajuste, foi implementada uma técnica de parada antecipada, que monitorava a perda de validação. Se a perda não apresentasse melhora por cinco épocas consecutivas, o treinamento era interrompido, e os melhores valores dos parâmetros, alcançados até aquele momento, eram restaurados automaticamente para garantir o melhor desempenho do modelo na validação.

Figura 11 - A matriz confusão do modelo CNN-DATAUG-RELU-L2-DROPOUT mostra a capacidade do modelo de classificar a ação *scratching* em cães.



Fonte: elaborado pelo autor, 2024.

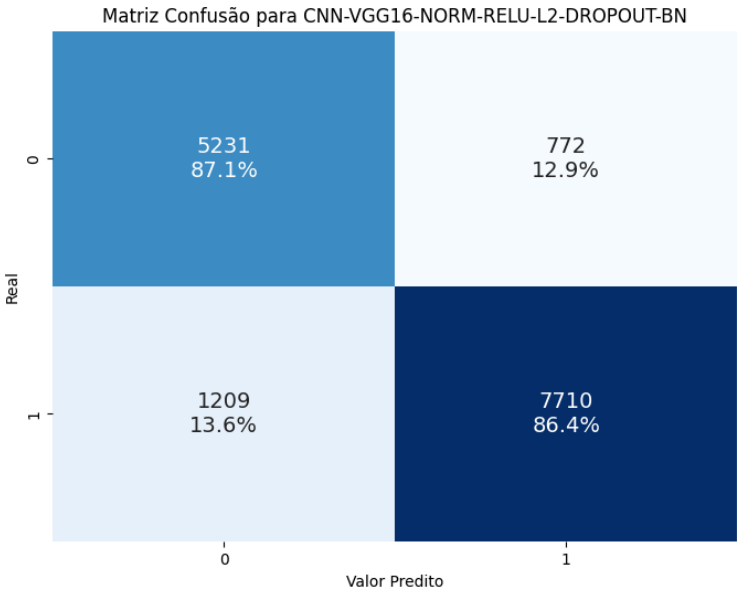
A matriz confusão apresentada na figura 11, fornece uma visão detalhada do desempenho do modelo CNN-DATAUG-RELU-L2-DROPOUT. Observamos que o modelo conseguiu classificar corretamente 5140 imagens como não *scratching* (verdadeiros negativos) e 7966 imagens como *scratching* (verdadeiros positivos). No entanto, 863 imagens foram incorretamente classificadas como *scratching* (falsos positivos) e 953 imagens foram incorretamente classificadas como não *scratching* (falsos negativos).

Com base nesses valores, as métricas de desempenho indicam que o CNN-DATAUG-RELU-L2-DROPOUT possui capacidade de classificar corretamente a ação de *scratching*, tanto em termos de precisão quanto de cobertura. A acurácia de 87.83% reflete uma alta taxa de acertos globais, enquanto a precisão de 90.23% indica que a maioria das previsões positivas do modelo foram corretas. Além disso, o F1-Score de 89.77%, que equilibra precisão e cobertura, sugere que o modelo é bem balanceado. Ele não apenas minimiza os falsos positivos, mas também garante que a maioria dos casos de *scratching* sejam detectados. Esse equilíbrio é essencial para garantir que o modelo seja útil em um contexto prático, onde tanto a precisão quanto a sensibilidade são necessárias.

Portanto, com base na arquitetura e resultados esse modelo pode ser considerado relativamente pequeno e eficiente, quando comparado com redes neurais convolucionais mais profundas que envolvem múltiplas camadas e milhões de parâmetros. Devido à sua menor complexidade, e ao optar pela classificação de ações com base em frames únicos em vez de sequências de imagens, a carga computacional é significativamente reduzida. Além disso, a implementação de técnicas de quantização pode ser efetuada com mínima perda de desempenho. Isso possibilita que o modelo processe imagens de maneira mais rápida e com menor consumo de energia, características essenciais para dispositivos embarcados, onde a eficiência energética e o tempo de resposta rápido são fundamentais.

Outro modelo testado que se destacou foi o CNN-VGG16-NORM-RELU-L2-DROPOUT-BN. A base do modelo foi a rede VGG16, carregada com pesos pré-treinados no conjunto de dados ImageNet, excluindo as camadas superiores. Todas as camadas do VGG16 foram congeladas para evitar o ajuste durante o treinamento, mantendo as características gerais aprendidas no ImageNet.

Figura 12 - A matriz confusão do modelo CNN-VGG16-NORM-RELU-L2-DROPOUT-BN destaca a performance do modelo na detecção da ação scratching em cães.



Fonte: elaborado pelo autor, 2024.

A matriz confusão apresentada para o modelo CNN-VGG16-NORM-RELU-L2-DROPOUT-BN revela um desempenho ligeiramente inferior em comparação ao modelo CNN-DATAUG-RELU-L2-DROPOUT na classificação da ação de coçar em cães.

Tabela 1 - tabela comparativa das métricas de validação para os modelos CNN-DATAUG-RELU-L2-DROPOUT e CNN-VGG16-NORM-RELU-L2-DROPOUT-BN.

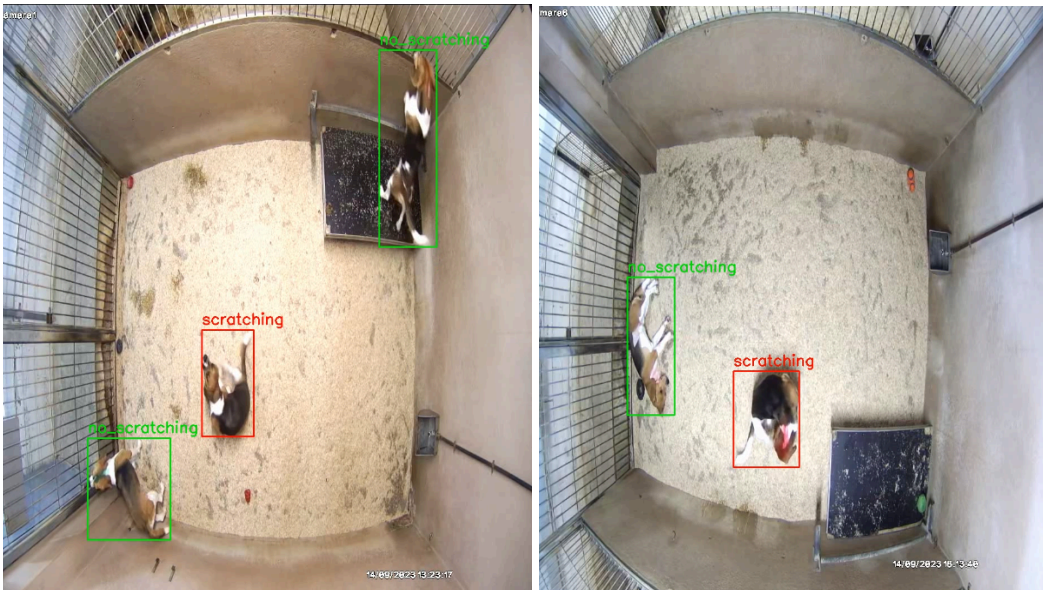
Métrica	**CNN-DATAUG-RELU-L2-DROPOUT**	**CNN-VGG16-NORM-RELU-L2-DROPOUT-BN**
Precisão	90,23%	**90,89%**
Cobertura (*Recall*)	**89,31%**	86,43%
Especificidade	85,62%	**87,14%**
F1-Score	**89,77%**	88,61%
Acurácia	**87,83%**	86,72%

Fonte: elaborado pelo autor, 2024.

A acurácia do modelo CNN-VGG16-NORM-RELU-L2-DROPOUT-BN é de aproximadamente 86.72%, que é ligeiramente inferior à acurácia de 87.83% do modelo CNN-DATAUG-RELU-L2-DROPOUT. Isso sugere que o modelo VGG16 é um pouco menos eficaz na classificação geral das ações de coçar e não coçar.

A precisão do modelo CNN-VGG16-NORM-RELU-L2-DROPOUT-BN é ligeiramente superior, com 90.89% em comparação com 90.23% do modelo CNN-DATAUG-RELU-L2-DROPOUT. A cobertura do modelo CNN-VGG16-NORM-RELU-L2-DROPOUT-BN é de 86.43%, inferior à cobertura de 89.31% do modelo CNN-DATAUG-RELU-L2-DROPOUT. O *F1-Score* do modelo CNN-VGG16-NORM-RELU-L2-DROPOUT-BN é de 88.61%, que é inferior ao *F1-Score* de 89.77% do modelo CNN-DATAUG-RELU-L2-DROPOUT.

Figura 12 - Imagens de classificação mostrando a detecção das ações coçar e não coçar em cães. A imagem foi analisada pelo modelo CNN-DATAUG-RELU-L2-DROPOUT.



Fonte: elaborado pelo autor, 2024.

A Figura 12 mostra duas imagens que foram extraídas de vídeos que tiveram seus *frames* analisados pelo modelo CNN-DATAUG-RELU-L2-DROPOUT. O modelo identificou corretamente os cães quando estavam se coçando (destacados em vermelho) e quando não estavam se coçando (destacados em verde).

4.2 Limitações

Porém, embora o modelo tenha demonstrado um bom desempenho, uma consideração importante é como esse desempenho pode ser afetado por variabilidades nas condições de captura das imagens. A falta de um estudo específico sobre esses fatores limita a compreensão completa da robustez do modelo e impede a sua implementação segura e eficaz em ambientes reais.

Por exemplo, em ambientes com pouca luz, a qualidade das imagens pode ser significativamente reduzida. Por outro lado, condições de iluminação intensa podem causar reflexos ou sombras fortes, que podem ser confundidos com características relevantes pelo modelo. Essa variabilidade pode levar a uma diminuição na acurácia do modelo, com um aumento potencial nos falsos positivos e falsos negativos.

Além disso, câmeras posicionadas em diferentes ângulos podem capturar as ações de formas variadas, afetando a visibilidade das características que o modelo utiliza para fazer suas previsões.

Para abordar essas questões, é necessário realizar experimentos com imagens capturadas sob diferentes condições de iluminação e posicionadas em ângulos variados. Somente através desses estudos adicionais será possível garantir que o modelo seja robusto e confiável em diversas situações práticas.

5. CONCLUSÃO

Este Trabalho de Conclusão de Curso (TCC) teve como objetivo principal desenvolver e validar um modelo de rede neural convolucional capaz de classificar, a partir de uma única imagem, se um cão está se coçando ou não.

O modelo desenvolvido, CNN-DATAUG-RELU-L2-DROPOUT, mostrou-se eficaz na classificação da ação de coçar em cães, com uma acurácia de 87.83%, que indica um desempenho robusto e equilibrado.

A arquitetura relativamente pequena e eficiente do modelo CNN-DATAUG-RELU-L2-DROPOUT torna-o adequado para implementação em dispositivos embarcados. A simplicidade do modelo, combinada com sua eficiência computacional, permite que ele opere dentro das restrições de hardware com recursos limitados.

No entanto, a variabilidade nas condições de captura das imagens, como iluminação e ângulos de câmera, precisa ser cuidadosamente considerada e abordada para garantir um desempenho consistente em ambientes reais.

Ainda assim, foi possível mostrar que embora modelos de reconhecimento de ações frequentemente utilizem sequências de *frames*, este estudo, assim como outros trabalhos (MATHEW et al., 2023), demonstra que a análise de *frames* individuais pode ser uma abordagem simples e eficaz.

5.1 Sugestões para Trabalhos Futuros

Uma das principais direções para trabalhos futuros é aumentar significativamente a base de dados utilizada para treinar o modelo. A coleta de dados adicionais deve incluir uma ampla variedade de raças de cães, diferentes condições de iluminação, múltiplos ângulos de câmera e diversos ambientes. Testar o modelo em diversas condições é crucial para validar sua robustez e garantir que ele mantenha um desempenho consistente e confiável em cenários do mundo real.

Outra direção promissora para trabalhos futuros é Adaptar o modelo CNN-DATAUG-RELU-L2-DROPOUT para TinyML visando permitir a execução eficiente em sistemas embarcados.

REFERÊNCIAS

1. VIRGA, V. Behavioral dermatology. Veterinary Clinics of North America: Small Animal Practice, v. 33, n. 2, p. 231-251, 2003.
2. STABACH, Jared A. et al. Short-term effects of GPS collars on the activity, behavior, and adrenal response of scimitar-horned oryx (Oryx dammah). PloS One, v. 15, n. 2, e0221843, 2020.
3. MARR, D.; VAINA, L. Representation and recognition of the movements of shapes. Proceedings of the Royal Society of London. Series B, Biological Sciences, v. 214, p. 501–524, 1982.
4. RIBEIRO, Cristina et al. Canine pose estimation: A computing for public safety solution. In: 2009 Canadian Conference on Computer and Robot Vision. IEEE, 2009. p. 37-44.
5. HESTER, C. F.; CASASENT, D. Multivariant technique for multiclass pattern recognition. Applied Optics, v. 19, p. 1758–1761, 1980.
6. SHAH, M. A.; JAIN, R. Detecting time-varying corners. Computer Vision, Graphics, and Image Processing, v. 28, n. 3, p. 345–355, 1984.
7. SETHI, I. K.; SALARI, V.; VEMURI, S. Feature point matching in image sequences. Pattern Recognition Letters, v. 7, n. 2, p. 113–121, 1988.
8. CÉDRAS, C.; SHAH, M. Motion-based recognition: a survey. Image and Vision Computing, v. 13, n. 2, p. 129–155, 1995.
9. HUNT, E. B. Artificial intelligence. Academic Press, 2014.
10. WANG, L.; HU, W.; TAN, T. Recent developments in human motion analysis. Pattern Recognition, v. 36, n. 3, p. 585-601, 2003.
11. BOBICK, A. F.; DAVIS, J. W. The recognition of human movement using temporal templates. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 23, n. 3, p. 257-267, 2001.
12. SEJNOWSKI, T. J. The deep learning revolution. MIT Press, 2018.
13. LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, v. 86, n. 11, p. 2278-2324, 1998.
14. JI, S.; XU, W.; YANG, M.; YU, K. 3D convolutional neural networks for human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 35, n. 1, p. 221-231, 2012.
15. NGUYEN, T. N.; MEUNIER, J. Anomaly detection in video sequence with appearance-motion correspondence. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. IEEE, 2019. p. 1273-1283.

16. RAJALAXMI, R. R.; GOTHAI, E.; SUGANTH, V.; VIGNESH, S.; VARUN, T. Vision based fall detection using optimized convolutional neural network. In: 2022 International Conference on Computer Communication and Informatics (ICCCI). IEEE, 2022. p. 01-06.
17. GARCIA, B.; VIESCA, S. A. Real-time American sign language recognition with convolutional neural networks. Convolutional Neural Networks for Visual Recognition, v. 2, p. 225-232, 2016.
18. ZHANG, C.; BERGER, C. Pedestrian behavior prediction using deep learning methods for urban scenarios: a review. IEEE Transactions on Intelligent Transportation Systems, v. 24, n. 10, p. 10279-10301, 2023.
19. WU, Z.; YAO, T.; FU, Y.; JIANG, Y. G. Deep learning for video classification and captioning. In: Frontiers of Multimedia Research. 2017. p. 3-29.
20. FENG, L.; ZHAO, Y.; SUN, Y.; ZHAO, W.; TANG, J. Action recognition using a spatial-temporal network for wild felines. Animals, v. 11, n. 2, p. 485, 2021.
21. DEAKIN, A. G.; BUCKLEY, J.; ALZU'BI, H. S.; COSSINS, A. R.; SPENCER, J. W.; AL'NUAIMY, W.; SNEDDON, L. U. Automated monitoring of behaviour in zebrafish after invasive procedures. Scientific Reports, v. 9, n. 1, p. 9042, 2019.
22. SCHMIDT, T. B.; LANCASTER, J. M.; PSOTA, E.; MOTE, B. E.; HULBERT, L. E.; HOLLIDAY, A.; PÉREZ, L. C. Evaluation of a novel computer vision-based livestock monitoring system to identify and track specific behaviors of individual nursery pigs within a group-housed environment. Translational Animal Science, v. 6, n. 3, txac082, 2022.
23. KOGER, B.; DESHPANDE, A.; KERBY, J. T.; GRAVING, J. M.; COSTELLOE, B. R.; COUZIN, I. D. Quantifying the movement, behaviour and environmental context of group-living animals using drones and computer vision. Journal of Animal Ecology, v. 92, n. 7, p. 1357-1371, 2023.
24. RAVBAR, P.; BRANSON, K.; SIMPSON, J. H. An automatic behavior recognition system classifies animal behaviors using movements and their temporal context. Journal of Neuroscience Methods, v. 326, p. 108352, 2019.
25. JUKAN, A.; MASIP-BRUIN, X.; AMLA, N. Smart computing and sensing technologies for animal welfare: a systematic review. ACM Computing Surveys (CSUR), v. 50, n. 1, p. 1-27, 2017.
26. SZELISKI, R. Computer vision: algorithms and applications. Springer Nature, 2022.
27. EL NAQA, I.; MURPHY, M. J. What is machine learning? In: Springer International Publishing, 2015. p. 3-11.
28. ALPAYDIN, E. Machine learning. MIT Press, 2021.

29. SOUTO, H.; MELLO, R.; FURTADO, A. An acoustic scene classification approach involving domestic violence using machine learning. In: Anais do XVI Encontro Nacional de Inteligência Artificial e Computacional. SBC, 2019. p. 705-716.
30. POWERS, D. M. Evaluation: from precision, cobertura and F-measure to ROC, informedness, markedness and correlation. arXiv preprint, arXiv:2010.16061, 2020.
31. HANLEY, J. A.; MCNEIL, B. J. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology, v. 143, n. 1, p. 29-36, 1982.
32. CRAWFORD, K. The atlas of AI: Power, politics, and the planetary costs of artificial intelligence. Yale University Press, 2021.
33. FUKUSHIMA, K. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, v. 36, n. 4, p. 193-202, 1980.
34. LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation applied to handwritten zip code recognition. Neural Computation, v. 1, n. 4, p. 541-551, 1989.
35. LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. Nature, v. 521, n. 7553, p. 436-444, 2015.
36. GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning. MIT Press, 2016.
37. GUO, Y.; LIU, Y.; OERLEMANS, A.; LAO, S.; WU, S.; LEW, M. S. Deep learning for visual understanding: a review. Neurocomputing, v. 187, p. 27-48, 2016.
38. PRINCE, S. J. Computer vision: models, learning, and inference. Cambridge University Press, 2012.
39. NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010. p. 807-814.
40. IOFFE, S.; SZEGEDY, C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. 2015. p. 448-456. PMLR.
41. BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010: 19th International Conference on Computational Statistics, Paris, France, August 22-27, 2010: Keynote, Invited and Contributed Papers. Physica-Verlag HD, 2010. p. 177-186.
42. KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, v. 25, 2012.
43. SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from

overfitting. The Journal of Machine Learning Research, v. 15, n. 1, p. 1929-1958, 2014.

44. BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, v. 5, n. 2, p. 157-166, 1994.
45. SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; RABINOVICH, A. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. p. 1-9.
46. RADFORD, A.; METZ, L.; CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint, arXiv:1511.06434, 2015.
47. HE, K.; ZHANG, X.; REN, S.; SUN, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 37, n. 9, p. 1904-1916, 2015.
48. LOWE, D. G. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, v. 60, p. 91-110, 2004.
49. BAY, H.; TUYTELAARS, T.; VAN GOOL, L. SURF: speeded up robust features. In: Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I. Springer Berlin Heidelberg, 2006. p. 404-417.
50. DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). IEEE, 2005. v. 1, p. 886-893.
51. BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. 1992. p. 144-152.
52. BENGIO, Y. Practical recommendations for gradient-based training of deep architectures. In: Neural Networks: Tricks of the Trade: Second Edition. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 437-478.
53. BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. Journal of Machine Learning Research, v. 13, n. 2, 2012.
54. SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. Advances in Neural Information Processing Systems, v. 25, 2012.
55. PRECHELT, L. Early stopping-but when? In: Neural Networks: Tricks of the Trade. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. p. 55-69.

56. TOMAR, S. Converting video formats with FFmpeg. Linux Journal, n. 146, p. 10, 2006.
57. MATHEW, S.; SUBRAMANIAN, A.; MS, B.; RAJAGOPAL, M. K. Human activity recognition using deep learning approaches and single frame CNN and convolutional LSTM. arXiv preprint, arXiv:2304.14499, 2023.
58. REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. *You Only Look Once: Unified, Real-Time Object Detection*.