

DOGREC: APLICATIVO PARA RECONHECIMENTO DE CACHORROS A PARTIR DE IMAGENS DO FOCINHO

Everton Luiz Piccoli, Aurélio Faustino Hoppe – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

epiccoli@furb.br, aureliof@furb.br

Resumo: Com o objetivo de auxiliar na busca de cachorros abandonados, maltratados ou perdidos, este artigo apresenta uma forma não invasiva de realizar o reconhecimento de cachorros através do desenvolvimento de uma aplicação Android que utilize algoritmos de aprendizado de máquina e imagens de seus focinhos. Utilizou-se a rede neural U-Net para realizar a segmentação do focinho, obtendo uma assertividade de 74,30%. Para o processo de reconhecimento do cachorro, utilizou-se a função de perda Contrastive Loss para auxiliar no aprendizado das características extraídas das redes neurais VGG16, VGG19, ResNet50, Inception-V3 e Xception, sendo classificadas por redes siamesas. Dentre as redes testadas, a VGG19 apresentou o melhor resultado, atingindo uma acurácia de 99,30%.

Palavras-chave: Reconhecimento. Focinho. Cachorro. Segmentação. Classificação. VGG16. VGG19. ResNet50. Inception-V3. Xception. One-Shot-Learning. Redes siamesas. Aplicativo.

1 INTRODUÇÃO

Nos dias de hoje, é muito comum encontrar famílias que adotam animais de estimação pelo simples fato deles proporcionarem benefícios emocionais ou até mesmo auxiliarem em questões relacionadas a parte física das pessoas. Além disso, é comum muitas pessoas realizarem a adoção um animal de estimação com alguns objetivos em **mentes**, seja eles o de ter a companhia de um amigo no seu dia a dia, auxiliar de certa forma a ensinar crianças sobre a importância e responsabilidade de saber cuidar de alguém, possuir um animal de guarda, ou ajudar no controle de fatores psicológicos como a ansiedade e o estresse. Porém, cuidar de um animal de estimação nem sempre foi uma tarefa fácil, pois eles precisam de cuidados constantes, como alimentação adequada, higiene, exercícios físicos e atenção emocional ou médica. Vale ressaltar que hoje em dia existem muitas pessoas que realizam a adoção de um animal sem realmente entender a realidade de se cuidar da forma correta de um animal de estimação. Elas podem achar que ter um animal de estimação é divertido e fácil, mas quando percebem que é um trabalho árduo e que requer tempo, dinheiro e energia, acabam abandonando o animal (Fernandes, 2019).

Segundo pesquisa realizada pelo Instituto Brasil Pet (IBP), atualmente no Brasil existem mais de 185 mil animais abandonados ou resgatados após maus-tratos. Desses, 60% dos cachorros resgatados são por maus-tratos e 40% por abandono (Puentes, 2022). Para entender tal contexto, realizou-se uma pesquisa para tentar identificar os motivos dos abandonos, dentre eles, os principais motivos apontados foram: mau comportamento de seus cães (46,8%), mudança no espaço onde mora o ser humano (29,1%), a mudança no estilo de vida do dono (25,4%) e expectativa frente à realidade de cuidar de um cachorro (14,9%) (Alves, 2013).

No Brasil, segundo Morel (2022), os animais abandonados são recolhidos por Organizações Não Governamentais (ONGs) e abrigos de cachorros. No entanto, os abrigos geralmente enfrentam vários desafios para conseguir cumprir com seu objetivo em cuidar destes animais, eles vão desde a superlotação do local, falta de recursos financeiros, falta de pessoas no quadro funcional, falta de espaço e instalações inadequadas. Além disso, muitos animais que são deixados nos abrigos têm problemas de saúde, comportamentais ou de idade, o que faz com que seja mais difícil deste animal encontrar um novo lar. Mas por outro lado, existem pessoas preocupados com seus pets, que têm buscado maneiras de monitorar e cuidar de seus cachorros, a fim de garantir a segurança e conhecimento de onde ele se encontra. Para isso, muitas vezes, optam pela utilização de métodos de monitoramentos que são invasivos como os microchips de Radio-Frequency Identification (RFID), Near Field Communication (NFC), entre outros. No entanto, o custo desses equipamentos encontra-se fora da capacidade financeira de grande parte da sociedade brasileira, perfazendo que muitos brasileiros optem pelo uso de uma simples coleira para reconhecer seus cachorros, e caso fujam, utilizem muitas vezes cartazes e redes sociais para encontrá-los (MOREL, 2022).

Melo (2020) destaca que assim como o ser humano pode ser identificado pela impressão digital de sua mão, os cachorros também podem ser identificados através das características presentes em seu focinho. A autora ainda reforça que o focinho do cachorro possui padrões complexos no conjunto de características, fazendo com que cada focinho seja único para cada cachorro, sendo assim possível o reconhecimento do cachorro com o uso de imagens. Com base neste contexto, o uso da técnica de foto-identificação com o auxílio de redes neurais para encontrar padrões e características

únicas como pigmentações, assimetrias, entre outras características pertinentes presentes no focinho se tornam uma grande aliada para este processo de identificação do cachorro através do focinho.

Diante deste cenário, o objetivo principal deste trabalho é disponibilizar uma aplicação móvel que seja capaz de identificar cães utilizando o processo de foto-identificação, considerando marcas naturais presentes em seus focinhos buscando auxiliar na identificação de cachorros encontrados nas ruas (abandonados ou que foram perdidos pelos seus responsáveis). Com isso os objetivos específicos são: a) realizar o reconhecimento de cachorros a partir de imagens, utilizando algoritmos de processamento de imagem e redes neurais para extrair características e padrões dos focinhos dos cachorros; b) avaliar a precisão e eficácia do modelo desenvolvido em diferentes condições de iluminação, ângulos de captura de imagem e variações de características dos focinhos; c) disponibilizar a aplicação desenvolvida em uma plataforma online para utilização por ONGs, abrigos de animais e proprietários de cachorros, visando auxiliar na identificação de animais perdidos ou abandonados.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo está subdividido em três seções. A seção 2.1 abordará alguns modelos de Redes Neurais Convolucionais, discutindo suas características, arquiteturas e aplicações. Na seção 2.2, será descrito o algoritmo de One-Shot-Learning e suas aplicações, com ênfase no uso de redes siamesas. Por fim, a seção 2.3 apresentará os trabalhos correlatos identificados durante a revisão bibliográfica, contextualizando a pesquisa atual em relação ao estado da arte.

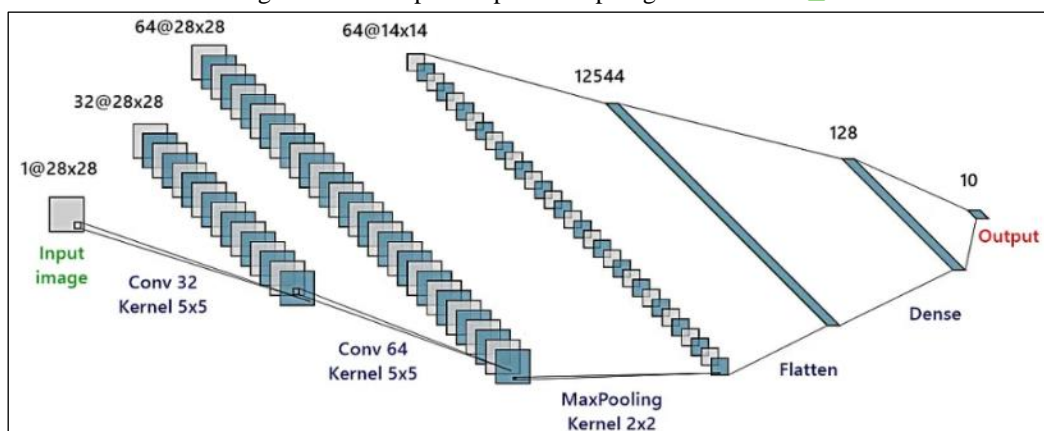
2.1 REDES NEURAIS CONVOLUCIONAIS

Segundo Clappis (2019), redes neurais convolucionais (CNNs, do inglês *Convolutional Neural Network*) são arquiteturas geralmente utilizadas na área de detecção de imagens, com o objetivo de classificá-las. As CNNs destacam-se por sua arquitetura única, composta por duas vertentes interligadas: uma responsável por extrair características da imagem e outra por classificá-la. A primeira parte, responsável pela extração de características, é composta por camadas e processos que realizam essa tarefa, sendo elas:

- camada de convolução: camada responsável por executar um *kernel* que transforma a imagem de entrada;
- processo de *padding*: processo em que são adicionados pixels ao redor da imagem, com o foco em manter sua dimensionalidade. esse processo geralmente é executado antes da operação de convolução;
- ReLU: função de ativação não linear que pode ter várias camadas de neurônios ativadas por ela. Sua função é: $y = \max(0, x)$;
- processo de *pooling*: processo de redução da imagem, com o objetivo de diminuir a variância a pequenas alterações e reduzir a quantidade de parâmetros a serem treinados pela rede. Existem três tipos de *pooling* (*MaxPooling*, *SumPooling* e *AveragePooling*), os quais seguem o mesmo princípio, mudando apenas o cálculo do valor final;
- flatten*: transforma a matriz da imagem, alterando seu formato para um *array*. A segunda parte, chamada de rede neural tradicional ou Camada Densa, é capaz de reconhecer padrões em uma grande quantidade de dados e classificá-los em alguma categoria.

A Figura 1 representa um exemplo de topologia/arquitetura de uma CNN.

Figura 1 – Exemplo simples de topologia de uma CNN.

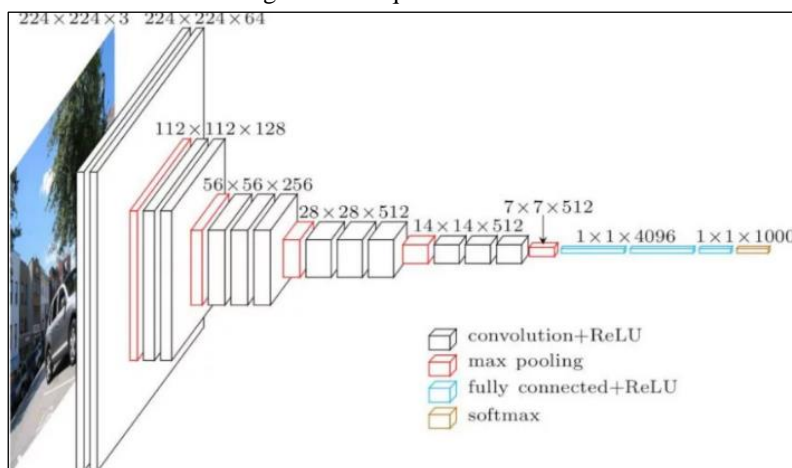


Fonte: Clappis (2019).

Segundo Mishra (2020), existem vários modelos de redes Neurais Convolucionais que podem extrair características dos focinhos dos cachorros, sendo eles: VGG16, VGG19, ResNet50, Inception-V3, Xception e U-Net. O modelo VGG16 (SIMONYAN; ZISSERMAN) recebeu este nome com base no departamento onde foi criado e pelo número de camadas que o compõe, sendo a sigla VGG (Visual Geometric Group) e o número 16 referente ao número de

camadas presente no modelo. A VGG16 comumente é utilizada para detecção de objetos e classificação de imagens, considerada até hoje um dos melhores modelos de classificação computacional. Os criadores deste modelo utilizaram o conceito de aplicar pequenos filtros/*kernel* de 3x3 nas camadas de convoluções que mostraram uma melhora significativa no resultado de seu modelo. A **VGG 16** é composta por 13 camadas de convoluções, 5 camadas de *MaxPooling* e 3 camadas densas, o que somando resulta em 21 camadas, porém apenas 16 delas possuem pesos de treinamento. A Figura 2 demonstra a arquitetura da **VGG 16**.

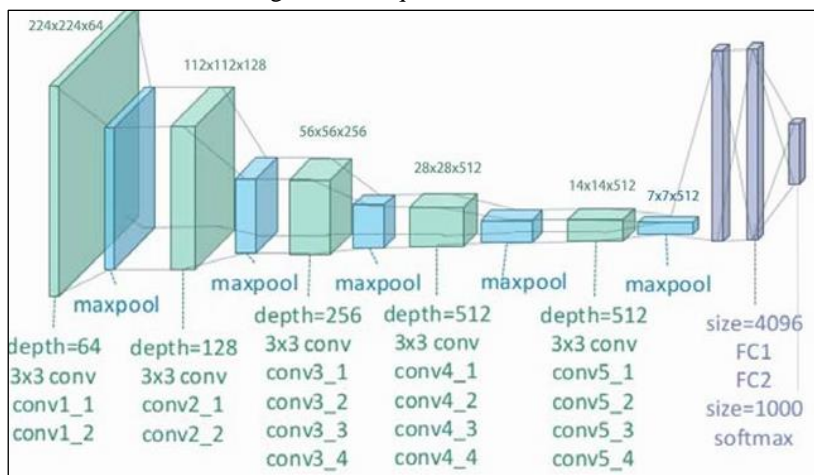
Figura 2 – Arquitetura VGG16



Fonte: Simonyan e Zisserman (2015).

A VGG19, assim como a VGG16, é uma arquitetura de rede neural convolucional conhecida por sua profundidade e simplicidade. Ambas as redes se destacam pelo uso predominante de camadas convolucionais com filtros 3x3 e passo 1, o que permite a extração de características complexas em diferentes níveis de abstração. A principal diferença entre a VGG16 e a VGG19 reside, no número de camadas convolucionais. Enquanto a VGG16 possui 13 camadas convolucionais, a VGG19 expande essa estrutura para 16 camadas, como ilustra a Figura 3. Essa diferença, embora pareça sutil, aumenta consideravelmente a capacidade da rede de aprender representações mais ricas e realizar classificações mais precisas.

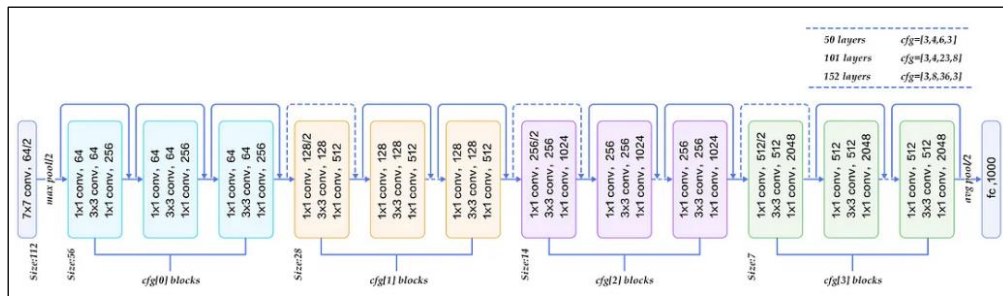
Figura 3 – Arquitetura **VGG 19**



Fonte: Simonyan e Zisserman (2015).

Segundo He *et al.* (2016), a ResNet50 possui esse nome como abreviação de Residual Network e o número 50, se refere ao número de camadas existentes no modelo. É considerado um dos modelos mais poderosos para o uso de classificação de imagens, por ter sido treinado com uma grande base de dados e obter resultados de última geração. Uma de suas principais técnicas utilizadas na estrutura são o uso de conexões residuais, que fazem com que a rede aprenda um conjunto de funções residuais capazes de mapear a entrada e saída da rede, possibilitando com que a rede seja capaz de aprender arquiteturas muito mais profundas. A Figura 4 apresenta a arquitetura do modelo ResNet50.

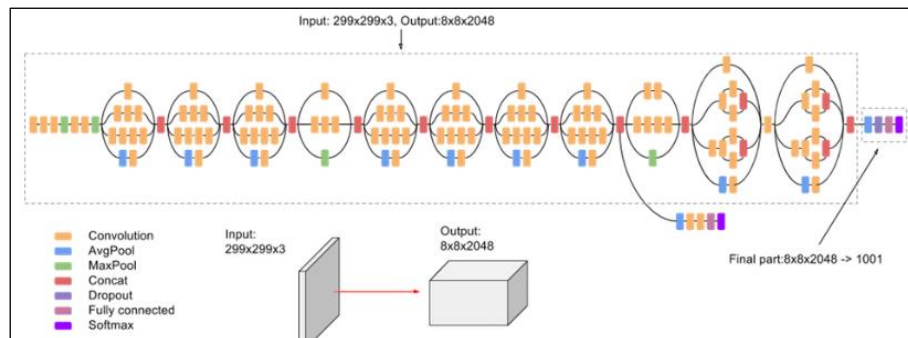
Figura 4 – Arquitetura da ResNet50



Fonte: He *et al.* (2016).

A arquitetura Inception-V3 (Szegedy, 2015), modelo desenvolvido com o objetivo primário de construir uma rede capaz de gastar menos poder computacional em relação as arquiteturas anteriores. Se mostrou uma rede neural muito eficiente em questões computacionais, tanto na questão dos pesos de treinos utilizados até a quantidade de custo de memória necessário para se realizar as operações. Para isso, na época em que foi desenvolvido esta arquitetura foram sugeridas diversas técnicas afins de alcançar o melhor custo computacional. Estas técnicas consistem em camadas de convoluções fatorizadas, regularização de imagens, redução de dimensões e cálculos paralelizados. A Figura 5 apresenta o formato da arquitetura de uma Inception-V3.

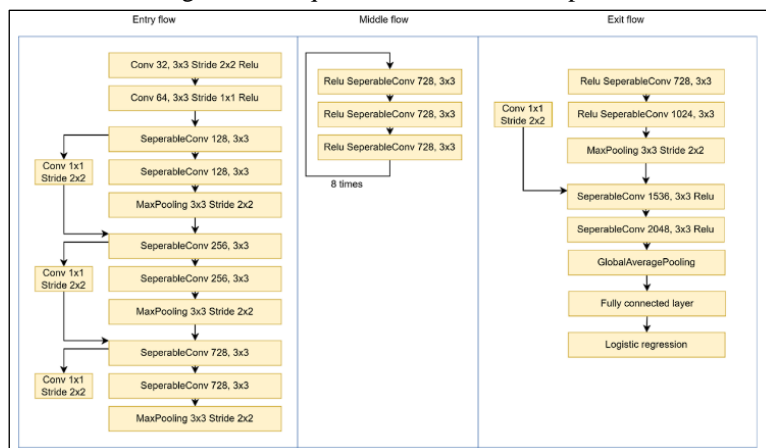
Figura 5 – Arquitetura do modelo Inception-V3



Fonte: Szegedy (2015).

De acordo com Cholett (2017), a Xception recebeu este nome devido a técnica que foi utilizada em seu desenvolvimento, chamada de Extreme Inception. Esta técnica se baseia na rede neural Inception, porém faz o uso de uma técnica chamada Depthwise Separable Convolution, que tem como objetivo a utilização de filtros diferentes para cada canal de imagem, onde ao invés de utilizar um filtro de 3x3, é realizado um filtro para cada canal de imagem e após isso se realiza uma convolução padrão com filtro de 1x1. Além desta técnica, a rede neural também faz o uso de blocos residuais com o objetivo de conseguir aprender camadas mais profundas durante a operação. A Figura 6 apresenta a estrutura de uma Xception.

Figura 6 – Arquitetura do modelo Xception

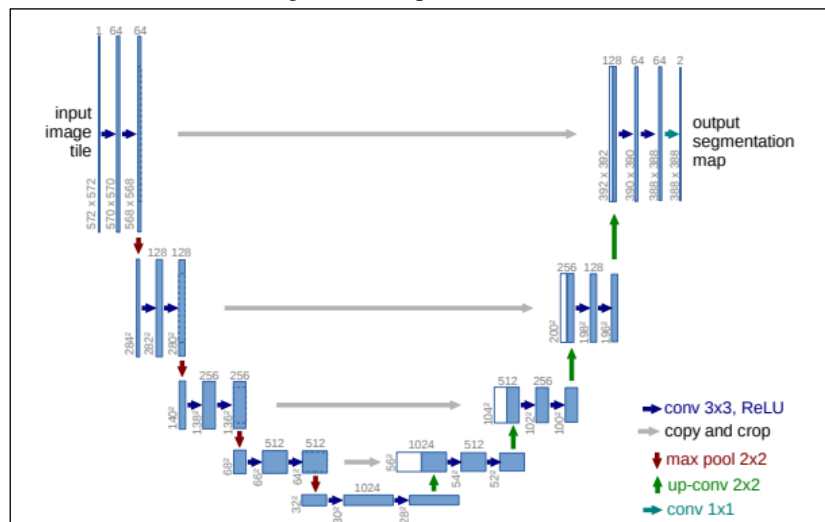


Fonte: Cholett (2017).

Por fim, a arquitetura U-Net (Ronneberger, 2015), desenvolvida inicialmente para a área da medicina, tem como sua principal funcionalidade realizar a segmentação de imagens. O nome U-Net é devido ao formato de sua arquitetura

que é composta por diversas camadas de convoluções em formato de U, onde cada bloco de convolução consiste em blocos 3x3, seguidos da ativação de uma função ReLU que auxilia na generalização dos dados de treino. Com o passar dos anos, esta rede neural deixou de ser apenas utilizada na área da medicina e se expandiu para outras áreas devido a sua grande capacidade em realizar tarefas de segmentações. A Figura 7 demonstra a arquitetura de uma rede U-Net.

Figura 7 – Arquitetura U-Net.



Fonte: Ronneberger (2015).

Segundo Marques (2017), para se realizar a avaliação de CNNs são utilizadas algumas métricas para comparação e validação dos modelos, onde cada métrica busca avaliar um aspecto diferente no modelo. As principais métricas são:

- acurácia: busca avaliar a quantidade de segmentos de classes positivas e classes negativas que foram classificadas de forma correta;
- precisão: avalia a quantidade de segmentos que foram classificados como positivos, ou seja, que pertencem ao segmento de classe positiva;
- recall: avalia a quantidade de segmentos positivos que foram corretamente classificados. Garantindo uma boa taxa de acerto para a classe positiva dos dados;
- F1 ou F-Score: considera na sua avaliação a taxa da precisão quanto a da métrica Recall. Onde o valor calculado se baseia na média entre a precisão e a recall, multiplicado pela **contaste 2**;
- Area Under the Receiver Operating Characteristic (AUROC): busca avaliar através de um gráfico cada segmento apresentado ao classificador, onde no eixo vertical são apresentados os números referentes as classes positivas, enquanto no eixo horizontal são apresentados os números referentes a classe negativa;
- Contrastive Loss: **Métrica** de função de perda geralmente utilizado no uso de redes siamesas para definir similaridade ou dissimilaridade de um par de imagens.

Marques (2017) também ressalta que cada métrica oferece uma perspectiva única sobre o desempenho do modelo, indo além da simples acurácia. Enquanto a precisão e o recall são essenciais em cenários com classes desbalanceadas, priorizando a identificação correta da classe minoritária, a área sob a curva ROC (AUROC) fornece uma visão geral da capacidade de classificação em diferentes limiares. Já a F1-Score, por sua vez, busca um equilíbrio entre precisão e recall, sendo particularmente útil quando ambas as métricas são importantes. Por fim, a Contrastive Loss se destaca em tarefas de comparação de imagens, como na verificação de similaridade. Portanto, a seleção das métricas deve ser guiada pelos objetivos específicos do projeto e pelas características do problema em questão, garantindo uma avaliação abrangente e significativa do modelo.

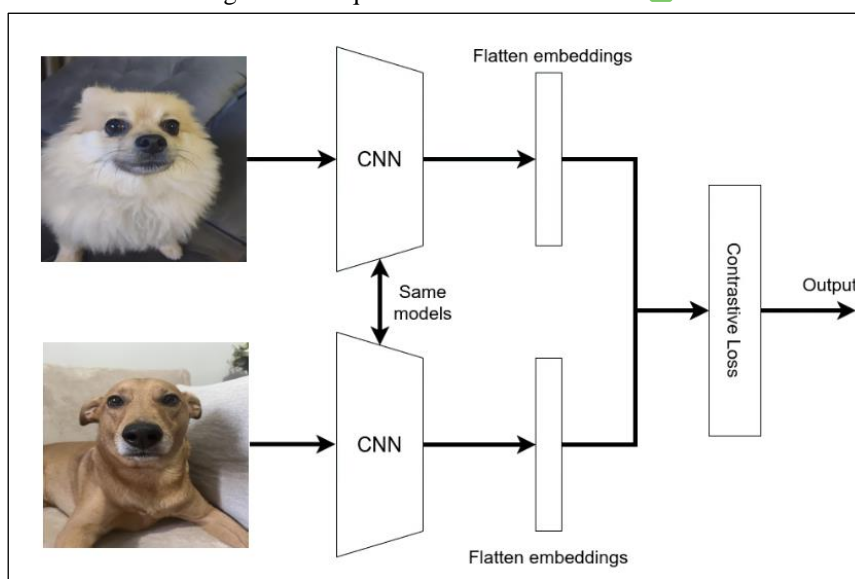
2.2 ONE-SHOT LEARNING

Segundo Logunova (2022), nos dias de hoje o uso de reconhecimento de imagens para executar tarefas simples como reconhecimentos faciais, detecção de objetos, entre outras tarefas, tem se mostrado cada vez mais promissor com o uso do **Machine Learning**. O autor, porém, explica que um dos grandes problemas enfrentados se encontra na necessidade de utilizar grandes bases de dados para treinar um modelo capaz de executar estas tarefas de maneira eficiente. Este problema, motivou o autor a realizar a abordagem de um algoritmo de **Machine Learning** capaz de realizar o reconhecimento de imagens com um número limitado de dados, o **One-shot learning**. O **One-shot learning** é um algoritmo de classificação de objeto responsável por realizar a avaliação de semelhanças entre um par de imagens, tendo como objetivo ensinar um modelo de rede neural a ser capaz de distinguir a semelhança entre um par de imagens, com o menor número de imagens possível para treinamento. Uma de suas vantagens em relação a outros algoritmos de **machine learning** tradicionais se encontra na capacidade de reconhecer novos dados sem a necessidade de realizar novos treinamentos. Para

realizar tarefas de reconhecimento de imagens, o **One-shot learning** realiza o uso de uma rede neural convolucional específica chamada de rede siamesa.

Segundo Srikan (2023), redes siamesas são redes neurais projetadas para realizar a comparação e medir a similaridade entre um par de imagens. Esta rede neural trabalha com uma arquitetura composta por redes gêmeas que compartilham da mesma estrutura e pesos, onde cada rede processa os dados de sua entrada e na saída são comparadas para determinar a semelhança ou dissimilaridade do par de entrada. Para definir a saída desta rede são utilizadas métricas de similaridades, que variam de acordo com a tarefa ou natureza dos dados, onde métricas comuns utilizadas envolvem o uso de distância euclidiana, similaridade do cosseno, entre outras. A métrica de similaridade quantifica ou realiza a correlação do vetor de características gerado por cada rede com o objetivo de definir a semelhança entre um par de imagens. Por fim o autor menciona que para realizar o treinamento de uma rede neural siamesa é realizado o uso de uma função de perda chamada **Contrastive Loss**, que incentiva a rede a produzir pares semelhantes e pares dissimilares. A Figura 8 apresenta a arquitetura de uma rede siamesa.

Figura 8 – Arquitetura de uma rede siamesa.



Fonte: Srikan (2023).

Para se realizar o cálculo de similaridade na função **Contrastive Loss**, segundo Dutt (2021), pares de imagens iguais ou diferentes são utilizados e treinados para definir que valores mais perto de 0 representam a semelhança entre imagens, enquanto valores próximos de 1 representam a dissimilaridade de imagens. A Equação abaixo demonstra o exemplo da função de cálculo utilizada na **Contrastive Loss** (ROSEBROCK, 2021).

$$Y * D^2 + (1 - Y) * \max(\text{margin} - D, 0)^2 \quad (1)$$

Segundo Rosebrock (2021), o uso da função Contrastive Loss é essencial para avaliar a eficácia do resultado obtido pelas redes siamesas ao classificar a similaridade de um par de imagens. A variável Y é definida como o label, onde será 1 para imagens iguais e 0 para imagens diferentes, na variável D tem-se a distância euclidiana gerado pelos vetores de características enquanto a função \max resgata o maior valor entre 0, definido pelo cálculo da margem de erro definida menos a distância euclidiana. O erro gerado por essa função tem como principal objetivo distanciar pares de imagens diferentes e aproxima pares de imagens semelhantes.

Dutt (2021) ressalta que a função Contrastive Loss tem se mostrado eficaz em diversos contextos onde a similaridade entre pares de dados é uma necessidade fundamental. Por exemplo, em sistemas de segurança baseados em reconhecimento facial, a capacidade de distinguir entre rostos semelhantes e diferentes é crucial para prevenir falsos positivos e negativos. Ou seja, a Contrastive Loss permite que modelos de redes siamesas aprendam a criar representações únicas e discriminativas para cada indivíduo, melhorando significativamente a precisão e a confiabilidade dos sistemas de segurança.

Além disso, de acordo com Rosebrock (2021), a Contrastive Loss é também aplicável em tarefas de aprendizado de representação em ambientes com poucos dados rotulados. Em tais cenários, onde a rotulagem manual é custosa e demorada, a capacidade de aprender a partir de um número limitado de exemplos é extremamente valiosa. A abordagem de One-Shot Learning, que frequentemente utiliza a Contrastive Loss, permite que modelos sejam treinados com um número mínimo de exemplos, mantendo uma alta performance em termos de generalização e precisão. Por fim, o autor

ressalta que ao focar em pares de dados sem a necessidade de rótulos extensivos, essa função de perda facilita a adaptação de modelos a novos domínios e tarefas com pouca ou nenhuma supervisão adicional. Isso abre portas para a aplicação de redes siamesas em áreas emergentes e complexas, onde a rotulagem de dados é impraticável ou inviável, mas a necessidade de entender a similaridade entre instâncias permanece crucial.

2.3 TRABALHOS CORRELATOS

Nesta seção serão apresentados 3 trabalhos correlatos que contribuíram para o desenvolvimento deste trabalho. No Quadro 1 será abordado o trabalho desenvolvido por Tu *et al.* (2018), que teve como objetivo desenvolver uma aplicação para realizar a identificação de um cachorro, usando como parâmetro fazer o reconhecimento da raça dele, pois segundo o autor ao reconhecer a raça você aumentaria as possibilidades de identificar o cachorro pois reduziria o campo de pesquisa. No Quadro 2, detalha-se o trabalho desenvolvido por Bhavani *et al.* (2019), que teve como objetivo realizar a identificação da raça de um cachorro usando redes neurais convolucionais pré-treinadas para extrair as características do cachorro. Por fim, o Quadro 3 apresenta o trabalho desenvolvido por Vaidya *et al.* (2022), que teve como objetivo identificar a raça do cachorro usando a face dele, e realizar o processo de detectar faces humanas e realizar a tentativa de assemelhar a face humana detectada a uma raça de cachorro.

Quadro 1 – Transfer Learning on convolutional neural networks for dog identification

Referência	Tu <i>et al.</i> (2018)
Objetivos	Identificar a raça e identidade do cachorro.
Principais funcionalidades	Realizar o reconhecimento da raça, para a partir deste ponto realizar o reconhecimento da identidade do cachorro. Para este processo foram usadas técnicas de extração de características com imagens que foram escolhidas levando em conta pontos chave importantes considerados, como face, partes do corpo etc. A base de dados utilizada neste trabalho foram bases públicas: Columbia Dogs Dataset (CD_Dogs), Standard Dogs Dataset (ST_Dogs), Flickr-dog Dataset
Ferramentas de desenvolvimento	Extração de características usando redes neurais convolucionais como GoogLeNet, BreedNet, DogNet
Resultados e conclusões	Com a base ST_Dogs foram alcançadas 81,74% de acurácia, enquanto para a base CD_Dogs foram alcançadas 86,63% de acurácia, no caso destas duas bases foram utilizadas apenas para identificar a raça do cachorro com a rede BreedNet. Para realizar a identidade do cachorro, foi utilizada a base de dados Flickr-dog utilizando a DogNet, onde para raça Pugs alcançou se a acurácia de 76,18%, enquanto a raça Husky obteve 90,05% de acurácia.

Fonte: elaborado pelo autor.

Quadro 2 – Dog breed identification using convolutional neural networks on android

Referência	Bhavani <i>et al.</i> (2019)
Objetivos	Identificar a raça do cachorro
Principais funcionalidades	Realizar a identificação da raça utilizando redes neurais convolucionais para extrair características. Utilizou-se a base de dados Standard Dogs Dataset.
Ferramentas de desenvolvimento	Utilizou as redes neurais convolucionais VGG16, Inception-V3, Xception e Inception-ResNet-v2 para extrair as características das imagens e realizar a classificação das mesmas.
Resultados e conclusões	A rede neural convolucional VGG16 alcançou 81% de acurácia, a Inception-V3 alcançou 89% de acurácia, a Xception alcançou 93% de acurácia, enquanto a Inception-ResNet-v2 alcançou a maior porcentagem de acurácia atingindo os 94%.

Fonte: elaborado pelo autor.

Quadro 3 – A novel dog breed identification using convolutional neural network

Referência	Vaidya <i>et al.</i> (2022)
Objetivos	Identificar raça do cachorro, assimilando-a a uma face humana.
Principais funcionalidades	Realizar a identificação de faces humanas, identificação de cachorros, utilizando o auxílio de redes neurais convolucionais e classificadores. Foram utilizadas 8351 imagens, sendo subdividida em 6680 imagens para treinamento, 836 imagens para os testes e 835 imagens para validação. Enquanto para a base de dados para detecção da face humana existia um conjunto de 13233 imagens.
Ferramentas de desenvolvimento	Utilizou a rede neural convolucional VGG16 para auxiliar na identificação dos cachorros, enquanto para a detecção de faces humanas foi usado o HaarCascade.
Resultados e conclusões	Para detecção de faces humanas o HaarCascade alcançou acurácia de 98%, enquanto para cachorros se alcançou apenas 17% de acurácia. Já em relação a VGG16, foi alcançado a acurácia de 81% no reconhecimento de faces humanas e cachorros.

Fonte: elaborado pelo autor.

3 DESCRIÇÃO DA APLICAÇÃO

Para melhor entendimento do desenvolvimento da aplicação, subdividiu-se este capítulo em três seções. A seção 3.1 abordará o desenvolvimento do modelo de segmentação do focinho, detalhando cada etapa realizada e apresentando ao final os resultados alcançados em relação a segmentação do focinho. A seção 3.2 apresentará o desenvolvimento do modelo de reconhecimento do cachorro, detalhando o processo de treinamento e testes realizados com cada rede neural convolucional, assim como os resultados obtidos. E, por fim, a seção 3.3 irá abordar sobre a aplicação móvel desenvolvida para apresentar os resultados obtidos pelos modelos desenvolvidos.

3.1 SEGMENTAÇÃO DO FOFINHO

Nesta seção será apresentado os principais Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) relacionados aos modelos de redes neurais utilizados na identificação do cachorro. Os Quadros 4 e 5 apresentam os requisitos relacionados ao módulo de segmentação do focinho.

Quadro 4 – Requisitos Funcionais

RF01 – Realizar o realce e melhoramento de ruídos, distorções e problemas de iluminação utilizando técnicas de processamento de imagens
RF02 – Utilizar redes neurais convolucionais para segmentar o focinho e identificar marcas naturais que possam caracterizar o indivíduo. Utilizado a U-net para segmentação do focinho.

Fonte: elaborado pelo autor

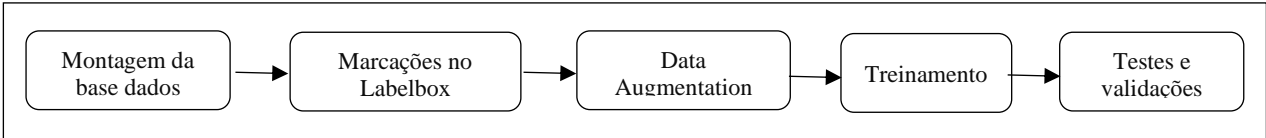
Quadro 5 – Requisitos Não Funcionais

RNF01 – Utilizar o ambiente de desenvolvimento Visual Studio Code
RNF02 – Utilizar as bibliotecas OpenCV para o processamento de imagens e o Tensorflow para a construção da rede neural artificial
RNF03 – Utilizar a linguagem Python para desenvolvimento.

Fonte: elaborado pelo autor.

Nesta etapa, optou-se pelo uso da rede neural convolucional U-net que está entre uma das melhores redes com o propósito de realizar a tarefa de segmentação sobre um objeto. Inicialmente desenvolvida para o uso na área de medicina, ela teve sua arquitetura modificada para conseguir trabalhar com bases menores de dados e conseguir produzir segmentações de forma mais precisa, e com o passar do tempo foi se estendendo o seu uso em outras áreas. Neste contexto, a U-net se tornou a responsável por realizar a parte de segmentação do focinho do cachorro, e para entender melhor o processo de desenvolvimento, ele foi dividido em 5 etapas: (i) Montagem da base de dados, (ii) utilização da ferramenta Labelbox para marcação da área de interesse, (iii) uso da técnica de DataAugmentation, (iv) treinamento dos dados e por fim (v) validação, testes e resultados obtidos. A Figura 9 apresenta o fluxo do desenvolvimento desta etapa do trabalho.

Figura 9 – Fluxo do desenvolvimento da etapa de segmentação do focinho.



Fonte: elaborado pelo autor.

Para a etapa de montagem da base de dados, utilizou-se uma base pública de dados chamada Stanford Dogs Dataset que continha cerca de 120 raças e um número de 20.580 imagens, no qual para cada raça continha em torno de 155 imagens para serem utilizadas. A Figura 10 demonstra exemplo de imagens encontradas dentro do dataset.

Figura 10 – Exemplo de imagens dentro do dataset



Fonte: elaborado pelo autor.

A partir da base de dados finalizada, procedeu-se à seleção das imagens, priorizando aquelas em que os cães apresentavam o focinho próximo, bem definido e em posição frontal. Ao final do processo, 355 imagens foram selecionadas, demandando a segmentação manual das máscaras. Esse procedimento é crucial para o treinamento da U-Net, que utiliza as imagens originais e suas respectivas máscaras para aprender a segmentar os focinhos de acordo com o padrão estabelecido. Para a segmentação manual, utilizou-se a ferramenta Labelbox, que permitiu delimitar a área de interesse – neste caso, o focinho do cão – em cada imagem, gerando a base de dados para o treinamento da U-Net. A Figura 11, apresenta respectivamente, um exemplo de segmentação manual realizada com a ferramenta Labelbox e o resultado obtido na imagem utilizada para o treinamento da U-Net.

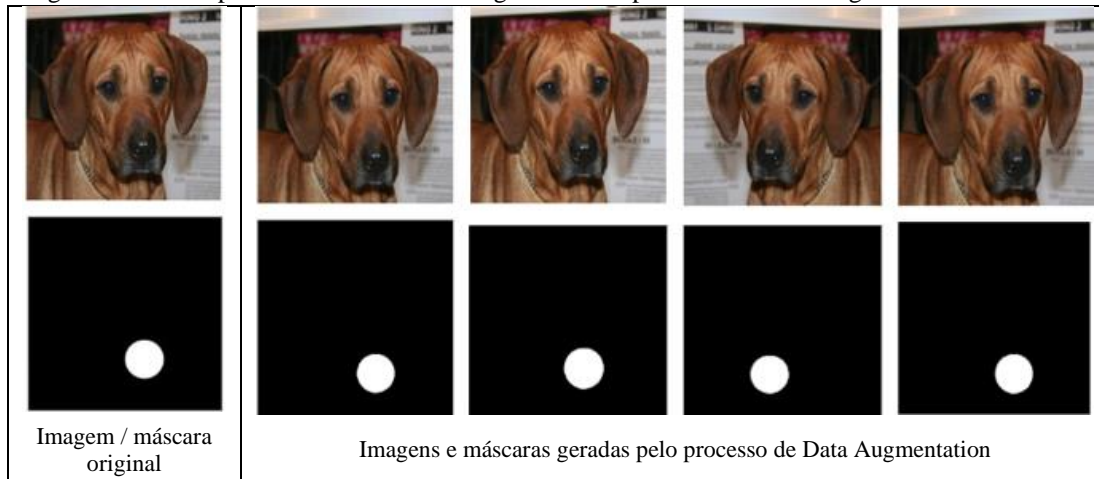
Figura 11 – Exemplo de segmentação manual do focinho realizada pela ferramenta Labelbox



Fonte: elaborado pelo autor.

Após a construção da base de dados, iniciaram-se os treinamentos e testes para verificar a assertividade do modelo. Contudo, o baixo número de imagens inicialmente utilizadas comprometeu a capacidade de generalização, limitando a segmentação a imagens muito semelhantes às da base de treinamento. Para solucionar essa limitação, empregou-se a técnica de Data Augmentation, utilizando a biblioteca ImageDataGenerator do Keras. Essa técnica permitiu gerar, a partir de cada imagem original, 10 novas imagens e máscaras, totalizando 11 imagens e 11 máscaras por imagem original, resultando em um total de 3.905 imagens e máscaras. As técnicas aplicadas no processo de Data Augmentation incluíram rotação, espelhamento, distorções aleatórias, cortes e redimensionamentos. A Figura 12 exemplifica a aplicação da Data Augmentation em uma imagem da base de dados.

Figura 12 – Exemplo da técnica de Data Augmentation aplicada sobre as imagens da base de dados



Fonte: elaborado pelo autor.

Ao iniciar o treinamento, constatou-se que a configuração da rede neural, em conjunto com o uso de imagens com dimensões de 256x256x3, gerava um alto custo computacional, especialmente em termos de memória RAM. Buscando otimizar o processo sem comprometer a qualidade do treinamento da U-Net, optou-se por utilizar um Gerador Customizado. Esse método, baseado em iterações, aplica o conceito de Data Augmentation à base de dados, mas com um menor impacto nos recursos computacionais.

Nesse contexto, a base de dados foi dividida em dois grupos: um de treinamento, contendo 80% das imagens, e outro de validação/teste, com os 20% restantes. Em ambos os grupos foi aplicado o conceito de gerador de dados, definindo-se os parâmetros de Data Augmentation a serem aplicados durante o treinamento. O processo utilizou 10 épocas, representando 10 ciclos completos de treinamento com as imagens geradas em cada lote. O parâmetro "steps_per_epoch" definiu a quantidade de lotes de imagens utilizados por época. A cada iteração, o gerador customizado criava, dinamicamente em memória, variações das imagens e máscaras da base de dados, otimizando o aprendizado do modelo. A Figura 13 apresenta o exemplo do código fonte onde é possível entender melhor o mecanismo de funcionamento de um gerador customizado. Nas duas primeiras linhas são definidos os parâmetros de Data Augmentation como rotação, espelhamento, distorções entre outras técnicas que serão utilizadas sobre o conjunto de imagens e máscaras. Nas próximas duas linhas, definiu-se os geradores de imagens e máscaras utilizando o método "flow_from_directory", que possui alguns parâmetros considerados importantes como o "target_size", que é o responsável por definir o tamanho das imagens geradas durante o treinamento, sendo definido como padrão 256x256. A partir da linha 5 é realizado a utilização do método "zip" unificando o grupo de imagens e máscaras, gerando um gerador de treino para imagens e máscaras, que será utilizado durante o processo de treino. E, por fim, na última linha do método têm-se uma iteração de for sobre o gerador de treino criados na linha 5, que retorna uma imagem e máscara a cada iteração, ao qual na linha 6 são normalizadas para o treino. Depois disso, elas são alocadas dinamicamente em memórias com o uso do yield, que basicamente gera um objeto destas duas imagens que são utilizadas durante o treinamento da rede.

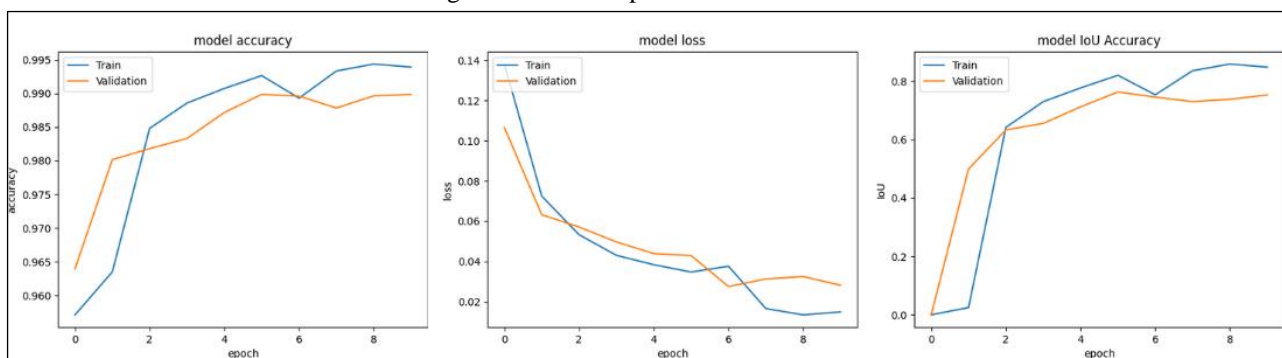
Figura 13 – Exemplo de código fonte do gerador customizado

```
def CustomGenerator(batch_size,train_path,image_folder,mask_folder,parametrosAug,save_to_dir = None,target_size = (256,256),seed = 1):
    image_datagen = ImageDataGenerator(**parametrosAug)
    mask_datagen = ImageDataGenerator(**parametrosAug)
    image_generator = image_datagen.flow_from_directory(
        train_path,
        classes = [image_folder],
        class_mode = None,
        target_size = target_size,
        batch_size = batch_size,
        save_to_dir = save_to_dir,
        save_prefix = 'imagem',
        seed = seed)
    mask_generator = mask_datagen.flow_from_directory(
        train_path,
        classes = [mask_folder],
        class_mode = None,
        target_size = target_size,
        batch_size = batch_size,
        save_to_dir = save_to_dir,
        save_prefix = 'mascara',
        seed = seed)
    train_generator = zip(image_generator, mask_generator)
    for (img,mask) in train_generator:
        img,mask = normalizaDados(img,mask)
        yield (img,mask)
```

Fonte: elaborado pelo autor.

Durante o treinamento, foram utilizados 150 lotes por época, enquanto a validação utilizou 45 lotes. A Figura 14 ilustra o desempenho do treinamento após a implementação dessas modificações.

Figura 14 – Desempenho do treinamento.



Fonte: elaborado pelo autor.

A Figura 14 demonstra que o modelo alcançou resultados excelentes em termos de acurácia e perdas durante o treinamento. No entanto, o gráfico mais relevante é o "Model IoU Accuracy", que representa a assertividade do modelo na segmentação das máscaras. A métrica Intersection-Over-Union (IoU) avalia a qualidade da segmentação, comparando a máscara gerada pela rede neural com a máscara original. Os resultados demonstram uma acurácia de 83,50% para os dados de treinamento e 74,30% para os dados de teste, o que pode ser considerado um resultado positivo, considerando a quantidade de dados utilizada no treinamento. A Figura 15 apresenta resultados da segmentação do focinho utilizando a arquitetura U-Net.

Figura 15 – Resultados de segmentação do focinho com a U-Net.



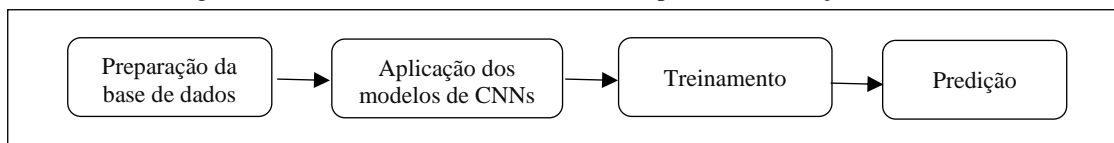
Fonte: elaborado pelo autor.

A partir da Figura 15, pode-se perceber a importância da conformidade da imagem aos padrões aprendidos pelo modelo durante o treinamento para a segmentação precisa do focinho. Na imagem da esquerda, o cão apresenta o focinho claramente visível, o que permite uma segmentação bem-sucedida. Em contraste, na imagem da direita, o cão não se enquadra nos padrões reconhecidos pelo modelo, resultando na geração de uma máscara vazia. Embora o modelo de segmentação de focinhos de cães tenha apresentado resultados promissores, a análise da Figura 15 expõe a necessidade de aprimorar sua capacidade de generalização. A disparidade de performance entre as duas imagens indica um possível viés em relação aos dados de treinamento, sugerindo que a diversidade de poses, raças e condições de iluminação presentes no conjunto de dados pode não ter sido suficiente. Para superar essa limitação e construir um modelo mais robusto e confiável de segmentação do focinho, capaz de lidar com a variabilidade do mundo real, é crucial investir na expansão e diversificação do conjunto de dados, na exploração de técnicas de Data Augmentation mais agressivas, no ajuste fino da arquitetura da rede neural e na análise aprofundada das falhas do modelo.

3.2 IDENTIFICAÇÃO DO CACHORRO

Esta seção abordará as etapas do desenvolvimento do modelo responsável por realizar o reconhecimento do cachorro a partir de seu focinho segmentado. A Figura 16 demonstra o fluxo do desenvolvimento, no qual em cada etapa foram realizados diversos testes com o objetivo de conseguir a máxima eficiência possível na identificação dos cachorros.

Figura 16 – Fluxo do desenvolvimento da etapa de identificação do cachorro.



Fonte: elaborado pelo autor.

Para a identificação de cães por meio da segmentação do focinho, foi necessário criar uma base de dados própria, dada a inexistência de opções públicas contendo apenas imagens de focinhos caninos. Para tanto, foram elaborados formulários nas plataformas JotForm e Google Forms, ambos acompanhados de Termo de Consentimento Livre e Esclarecido, convidando voluntários a cederem fotos de seus cães para o treinamento do modelo. Os formulários especificavam critérios para as imagens, como: evitar o uso de *zoom*, manter uma distância mínima de 10cm do focinho, garantir nitidez, e enviar pelo menos quatro fotos de ângulos distintos. Ao final, foram recebidas 46 respostas, totalizando 142 imagens. Após rigorosa seleção, apenas 30 imagens, de 15 cães, foram consideradas adequadas para o treinamento, priorizando qualidade e adequação aos parâmetros do modelo. A Figura 17 ilustra um exemplo de imagem aprovada e outra descartada durante a seleção.

Figura 17 – Exemplos de imagens capturadas pelos usuários



Fonte: elaborado pelo autor.

Após a seleção das imagens e a criação da base de dados inicial, procedeu-se à segmentação dos focinhos, resultando em 30 imagens. Esse número, contudo, mostrou-se insuficiente para o treinamento robusto de uma rede neural. Para suprir a demanda por dados, optou-se novamente pela técnica de Data Augmentation, que gerou 26 novas imagens a partir de cada original, totalizando 771 imagens.

Com a base de dados expandida, iniciou-se a fase de testes, visando identificar qual rede neural convolucional e classificador são os mais eficazes no contexto de identificação canina. Cinco arquiteturas foram avaliadas: VGG16, VGG19, ResNet50, Inception-V3 e Xception. O objetivo era utilizar essas redes para extrair características relevantes dos focinhos, como formas, texturas e porosidade. Para essa finalidade, contudo, modificações foram implementadas. As camadas totalmente conectadas das redes foram "congeladas" utilizando o parâmetro `"include_top = false"`, configurando-as para a extração de características, e não para a classificação, propósito original dessas arquiteturas. A Tabela 1 apresenta alguns exemplos de valores de características gerados por cada CNN, as saídas geradas por cada CNN, assim como o número de posições do vetor gerado após a normalização dos dados, transformando a matriz em um vetor 1D para adequar os dados a serem utilizados pelos classificadores.

Tabela 1 – Demonstração de resultados gerados por cada CNN

Redes Neurais	Saída	Posições do vetor gerado	Exemplo de características geradas por CNN			
VGG16	7x7x512	25088	0.055526536	0.033978578	0.0066558123	0.035707954
VGG19	7x7x512	25088	0.17795132	0.31073073	0.15854058	0.15726838
Inception-V3	5x5x2048	51200	0.08365497	0.5871211	0.04584863	0.14092232
ResNet50	7x7x2048	100352	0.032836914	0.6365967	0.6861572	0.2841797
Xception	7x7x2048	100352	0.7402344	1.0	0.9589844	0.3750000

Fonte: elaborado pelo autor.

Na primeira parte dos testes, utilizou-se das redes neurais convolucionais para extrair as características dos focinhos e posteriormente foram utilizados alguns classificadores pré-treinados com o propósito de determinar se algum deles se adequaria ao conjunto de características enviado e assim trazer o melhor resultado, sendo eles: Decision Tree (DT), K-Nearest Neighbors (kNN), Logic Regression (LR), Suport Vector Machine (SVM), RandomForest (RF), Gradient Boosting Classifier (GBC) e Naive Bayes Gaussian (GaussianNB). Após definidos os classificadores que seriam utilizados nos testes, adaptou-se as características extraídas pelas redes neurais aos classificadores pois eles recebem como entrada um vetor de características, enquanto as características que eram retornada pelas redes neurais possuíam padronizações diferentes, como por exemplo a VGG16, que retornava uma matriz de características com as dimensões 7x7x512, onde aplicando técnicas para normalizar os dados se obteve um vetor1D com 25088 posições contendo as características extraídas entre os valores 0 e 1. A Figura 18 demonstra um exemplo do código fonte que realiza a normalização de dados e a adaptação feita para o vetor de característica estar apto a ser utilizado pelos classificadores. Na primeira linha é realizado o uso de um *MinMaxScaler*, no qual definiu-se que os valores do vetor de características ficariam entre 0 e 1. A partir da segunda linha é realizado a predição das características por cada rede neural com seu pré-processamento de imagem e na linha seguinte é onde padronizamos os dados para o classificador transformando a matriz em um vetor de posições contendo as características extraídas e por fim na última linha temos a normalização dos valores entre 0 e 1.

Figura 18 – Exemplo de código fonte demonstrando normalização dos dados

```
scaler = MinMaxScaler(feature_range=(0, 1))
feat_treino_extracted = modelo.predict(pre_process(dados_treino))
feat_treino_reshape=feat_treino_extracted.reshape(feat_treino_extracted.shape[0],-1)
feat_treino_normalizado = scaler.fit_transform(feat_treino_reshape)
```

Fonte: elaborado pelo autor.

Para realizar o treinamento dos classificadores foram realizadas a separação dos dados de treino e teste, no qual optou-se pela proporção de 80% de imagens para treino e 20% de imagens para teste, gerando 616 imagens de treino e 155 imagens para testes. Com o conjunto de dados subdivido, foram realizadas as predições das características do grupo de imagens de treino e teste, e realizada a normalização dos dados, convertendo as matrizes de características retornadas pelas CNN em vetores e padronizado os seus valores entre 0 e 1, para assim estarem aptos a serem treinados pelos classificadores. A Tabela 2 demonstra os percentuais de acurácia obtidos sobre o grupo de teste após os dados de treino terem sido treinados com cada classificador.

Tabela 2 – Acurácia obtida por classificador e CNN

Redes CNN	DT	kNN	LR	SVM	RF	GBC	GaussianNB
VGG16	64.86%	91.89%	100.00%	94.59%	97.30%	86.49%	97.30%
VGG19	51.35%	75.68%	100.00%	100.00%	100.00%	72.97%	97.30%
Inception-V3	18.92%	10.81%	18.92%	18.92%	13.51%	8.11%	5.41%
ResNet50	18.92%	8.11%	8.11%	8.11%	8.11%	18.92%	2.70%
Xception	18.92%	8.11%	18.92%	18.92%	8.11%	18.92%	5.41%

Fonte: elaborado pelo autor.

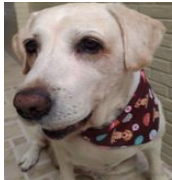
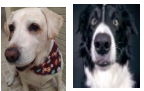
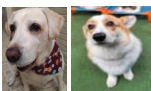
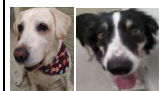
A Tabela 2 demonstra que as redes VGG16 e VGG19 tiveram um bom desempenho na tarefa de classificação das características do focinho. Elas atingiram altas taxas de acurácia com a maioria dos classificadores testados (DT, kNN, LR, SVM, RF e GaussianNB). Nota-se que ambas possuem alta capacidade para extrair e apresentar características relevantes para a identificação. Em contraste, as redes Inception-V3, ResNet50 e Xception apresentaram resultados significativamente inferiores, evidenciando uma possível dificuldade na adaptação dos classificadores às características por elas extraídas.

É importante ressaltar, no entanto, que a quantidade limitada de dados utilizada (apenas 30 imagens originais) pode ter influenciado os resultados. As altas taxas de acerto obtidas pelas redes VGG16 e VGG19, embora promissoras, podem não se sustentar em cenários reais com maior variabilidade. A escassez de dados aumenta o risco de *overfitting*, onde o modelo aprende os padrões específicos do conjunto de treinamento sem generalizar para novas amostras. Contudo, apesar do bom desempenho alcançado pelas redes VGG16 e VGG19 em conjunto com os classificadores, especula-se que o aumento na quantidade de dados de treinamento poderia gerar resultados ainda melhores e mais confiáveis. A baixa quantidade de dados pode ter limitado a capacidade de generalização dos classificadores, tornando-os potencialmente inadequados para os objetivos finais deste trabalho que seria a disponibilização da aplicação de reconhecimento para a comunidade em geral.

Seguindo para a última etapa dos testes e, considerando que o uso dos classificadores não seria o ideal para o trabalho, foram pesquisadas novas soluções até se chegar no algoritmo de *One-shot learning* com uso de redes siamesas, que é uma das técnicas que melhor se adapta ao cenário do trabalho, pois ela é capaz de alcançar ótimos resultados com um baixo número de dados. Para o treinamento das redes siamesas, desenvolveu-se um método para criar pares de imagens, no qual eram criados pares do mesmo *labels* e pares com *labels* diferentes, a fim de validar a questão da

similaridade e dissimilaridade das imagens da base de dados. A Figura 19 demonstra a verificação de pares e suas similaridades (em verde) e as dissimilaridades (em vermelho).

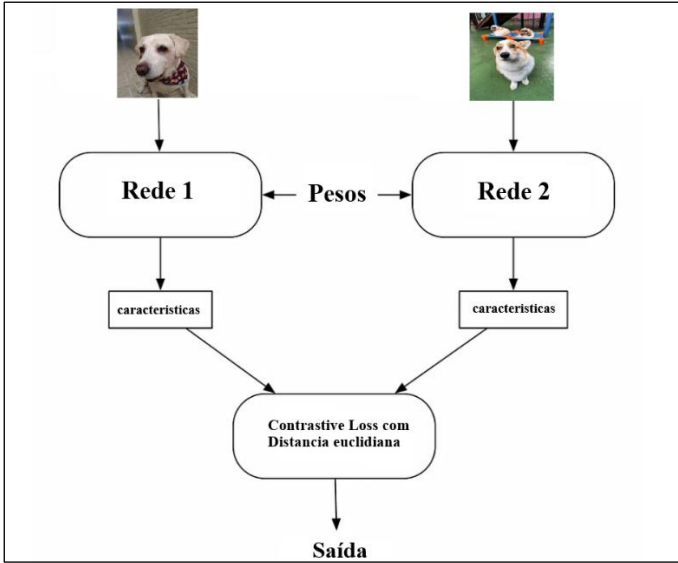
Figura 19 – Exemplo de pares demonstrando similaridade e dissimilaridade

					
Imagem entrada	Grau de similaridade: 0.46	Grau de similaridade: 0.67	Grau de similaridade: 0.67	Grau de similaridade: 0.67	Grau de similaridade: 0.67

Fonte: elaborado pelo autor.

Ao fim deste processo, obteve-se um total de 3780 pares de imagens, divididos entre pares iguais e pares diferentes. A partir da base de dados criada, realizou-se o treinamento de cada rede siamesa, no qual cada uma tinha como modelo base as respectivas redes neurais convolucionais VGG16, VGG19, ResNet50, Inception-V3 e Xception. O objetivo destas redes era gerar o vetor de características a ser calculado pela rede siamesa com a distância euclidiana. A partir da Figura 20 é possível analisar o uso das redes siamesas. Nela, são passadas duas imagens de entrada que utilizam o mesmo modelo base, responsáveis por gerar o vetor de características que serão utilizados no cálculo da distância euclidiana. A partir do cálculo realizado é utilizado a função Contrastive Loss para avaliar a semelhança ou dissimilaridade do par de imagens e a partir do resultado desta função, ele é enviado para uma camada totalmente conectada, que utiliza a função sigmoid, obtendo valores entre 0 e 1, que será responsável por apresentar a saída da rede apresentando o resultado de similaridade.

Figura 20 – Representação do cálculo da distância euclidiana com os vetores de características de duas imagens.



Fonte: elaborado pelo autor.

A Tabela 3 revela um desempenho superior das redes siamesas na tarefa de classificação, superando os resultados alcançados pelos classificadores tradicionais em todas as redes CNN avaliadas. A rede VGG19, em particular, se destaca com uma acurácia excepcional, tanto no conjunto de treino (99.80%) quanto no conjunto de teste (99.30%). Esse resultado sugere uma alta capacidade de aprendizado e generalização da VGG19 quando integrada à arquitetura siamesa. Além disso, também pode-se perceber uma melhoria significativa no desempenho das redes Inception-V3, ResNet50 e Xception ao utilizarem o classificador siamês. Enquanto essas redes apresentaram dificuldades com os classificadores tradicionais, limitadas por baixa quantidade de dados e possível *overfitting*, a abordagem siamesa permitiu que alcançassem acurácias acima de 87%, demonstrando a robustez da técnica.

Tabela 3 – Resultado do treinamento das redes siamesas

Redes Utilizada	Treino	Teste
VGG16	93,65%	93,78%
VGG19	99,80%	99,30%
Inception-V3	91,30%	90,20%
ResNet50	92,40%	93,10%
Xception	85,10%	87,20%

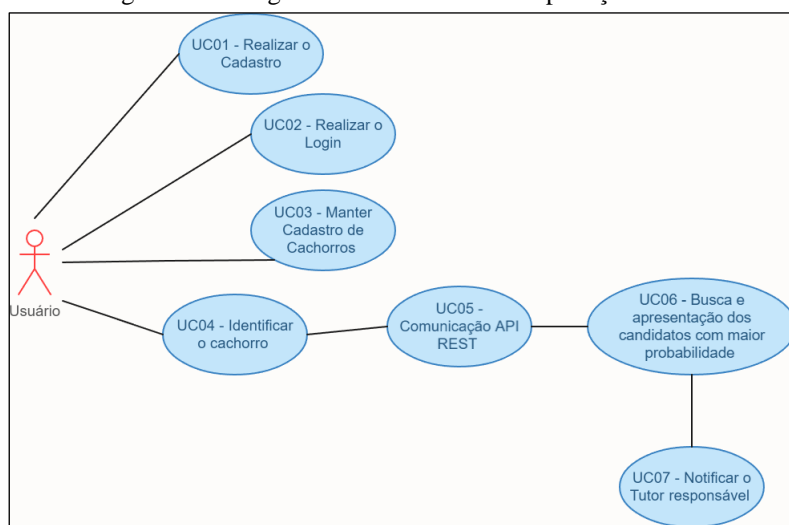
Fonte: elaborado pelo autor.

Por fim, conclui-se que a efetividade das redes siamesas, nesse contexto, pode ser atribuída à sua capacidade intrínseca de aprender relações de similaridade entre pares de imagens. Em vez de depender de um grande volume de dados para generalizar padrões complexos, as redes siamesas aprendem a diferenciar características-chave que distinguem as classes, tornando-se menos suscetíveis ao *overfitting*, especialmente em cenários com dados limitados. Portanto, os resultados da Tabela 3 indicam que a utilização de redes siamesas como classificadores se mostra altamente promissora para a identificação de cães por meio do focinho.

3.3 APLICAÇÃO MÓVEL

A aplicação móvel destinada à identificação cães contará com as funcionalidades detalhadas no diagrama de caso de uso apresentado na Figura 21. O diagrama ilustra que o usuário, após realizar seu cadastro (UC01) e efetuar login na plataforma (UC02), terá acesso a diversas funcionalidades. Dentre elas, destaca-se a gestão do cadastro de cães, que permite incluir, editar e excluir informações (UC03). Em caso de busca por um animal perdido, o usuário poderá utilizar a funcionalidade de identificação (UC04), que enviará uma requisição ao servidor da API REST (UC05) e retornará uma lista de possíveis candidatos (UC06). Se o cão procurado for encontrado na lista, o usuário terá a opção de notificar o tutor ou responsável (UC07).

Figura 21 – Diagrama de caso de uso da aplicação móvel



Fonte: elaborado pelo autor.

A partir do diagrama de caso de uso apresentado, realizou-se o levantamento dos requisitos da aplicação móvel, onde no Quadro 6 são apresentados os Requisitos Funcionais (RF), enquanto no Quadro 7 são apresentados os Requisitos Não Funcionais.

Quadro 6 – Requisitos Funcionais

RF01 – Manter o cadastro dos cachorros
RF02 – Permitir ao usuário carregar ou capturar imagens a partir de um dispositivo Android
RF03 – Enviar notificação via SMS para o responsável do cachorro (maior probabilidade) após reconhecimento, contendo a posição geográfica na qual ele possivelmente foi encontrado.
RF04 – Permitir ao usuário visualizar as detecções e resultados obtidos no processo de reconhecimento

Fonte: elaborado pelo autor

Quadro 7 – Requisitos Não Funcionais

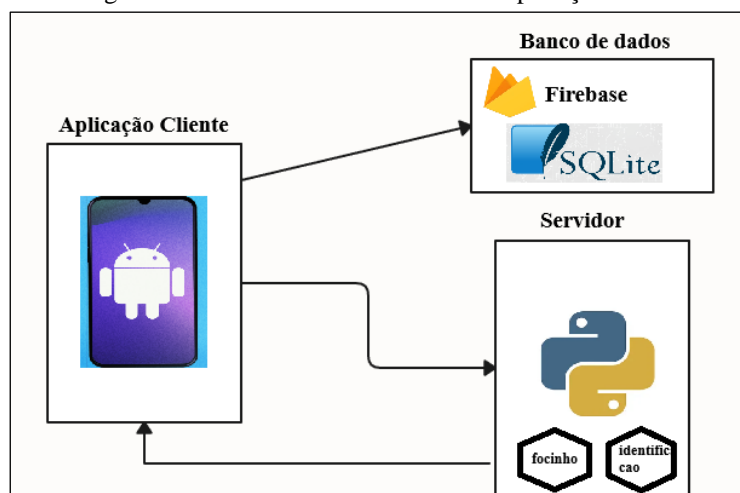
RNF01 – Ser desenvolvido para a plataforma Android
RNF02 – Utilizar a linguagem Dart com o SDK Flutter para o desenvolvimento
RNF03 – Utilizar o ambiente de desenvolvimento Visual Studio Code

RNF04 – Utilizar o banco de dados SQLite para persistir os dados off-line
RNF05 – Utilizar um banco de dados No-SQL
RNF06 – Utilizar servidor da API REST para executar os modelos desenvolvidos e retornar os candidatos.

Fonte: elaborado pelo autor.

A Figura 22 apresenta a arquitetura e o fluxo do processo da aplicação móvel, estruturada em três módulos principais: (i) Aplicação Cliente: Interface com o usuário, desenvolvida em Flutter, que oferece as funcionalidades principais da aplicação móvel; (ii) Banco de Dados: utiliza o Firebase para armazenar os dados dos cães cadastrados pelos usuários, garantindo sincronização online, e o SQLite para manter a consistência dos dados em modo offline; (iii) Servidor: Responsável por executar o processo de busca dos candidatos, utilizando os modelos de redes neurais desenvolvidos.

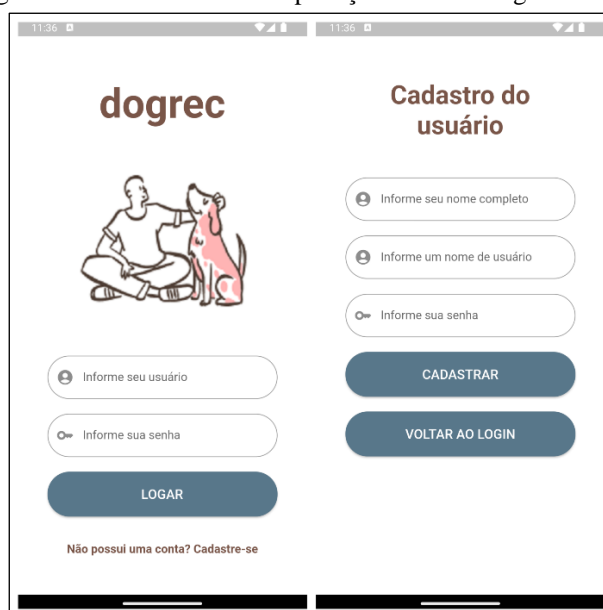
Figura 22 – Demonstrativo do fluxo da aplicação móvel



Fonte: elaborado pelo autor.

O desenvolvimento da aplicação móvel para a plataforma Android foi realizado utilizando a linguagem Flutter. Para utilizar a aplicação, o usuário precisa primeiramente realizar seu cadastro. Uma vez cadastrado, o usuário deve inserir suas credenciais na tela de login para acessar as funcionalidades da aplicação. A Figura 23 ilustra as telas de login e de cadastro de usuários da aplicação móvel.

Figura 23 – Telas iniciais da aplicação móvel – Login/Cadastro



Fonte: elaborado pelo autor.

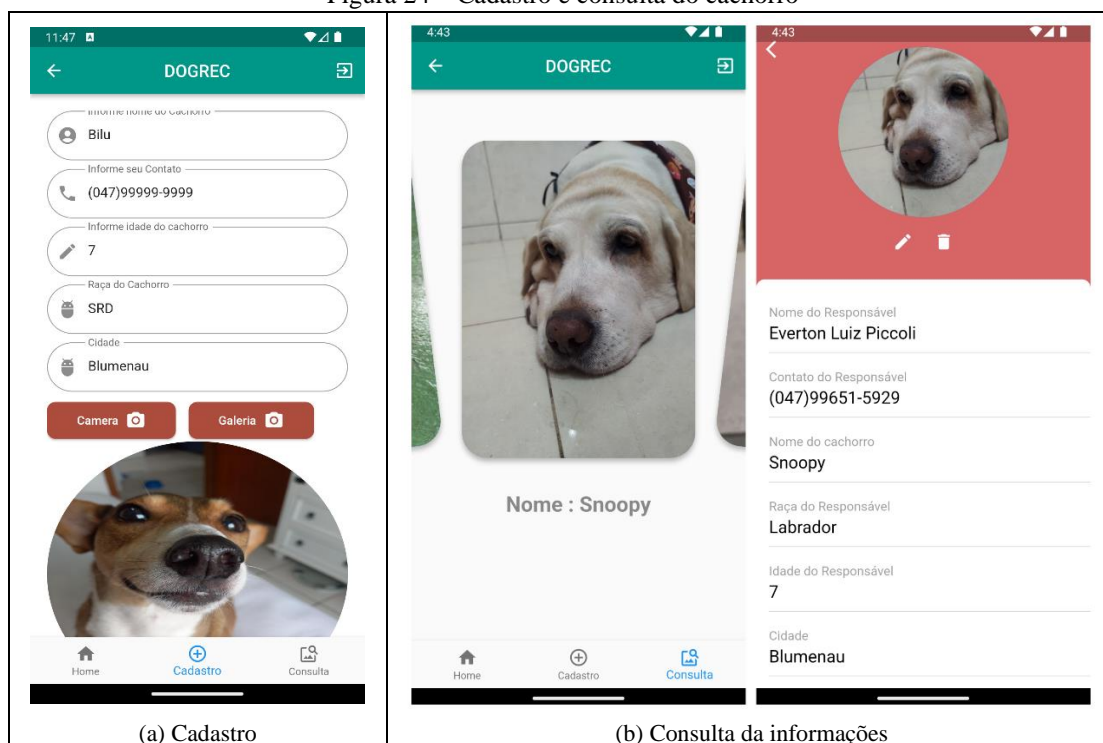
Após o login, o usuário terá acesso a tela inicial da aplicação, que se divide em três seções principais:

- Página Inicial (Home):** Permite ao usuário realizar a busca por cães utilizando uma imagem como entrada.
- Página de Cadastro:** Oferece a funcionalidade de cadastro de novos cães no sistema, incluindo informações

- relevantes para a identificação.
- c) **Página de Listagem:** Exibe os cães cadastrados pelo usuário, possibilitando a edição ou exclusão de cadastros existentes.

Na **página de cadastro**, o usuário deve fornecer informações precisas sobre o cão, pois esses dados são cruciais para o processo de identificação. A precisão das informações é fundamental, visto que dados incorretos podem prejudicar a busca pelo animal em caso de perda. Ao finalizar o cadastro, as informações são salvas localmente no banco de dados SQLite, permitindo edições offline, e simultaneamente registradas na nuvem através do Firebase, garantindo a persistência e disponibilidade dos dados. A Figura 24 item (a) ilustra um exemplo de cadastro sendo realizado na aplicação. Já a Figura 24 item (b) ilustra a página de consulta e um exemplo de visualização de informações de um cadastro.

Figura 24 – Cadastro e consulta do cachorro

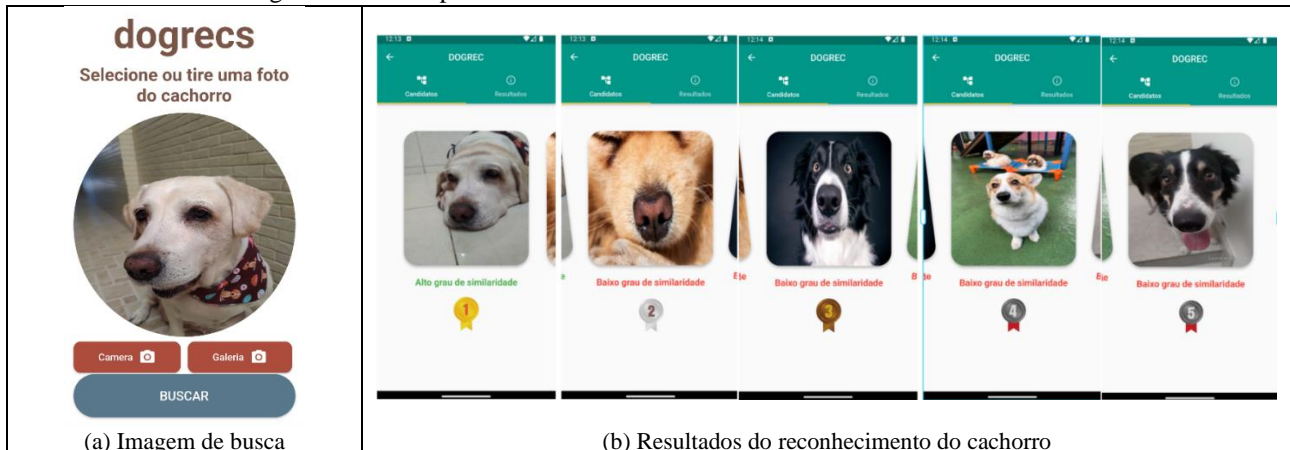


Fonte: elaborado pelo autor.

Após o cadastrado do cão, o usuário pode realizar alterações ou exclusões por meio da página de consulta. Essa seção tem como objetivo permitir a atualização das informações do cadastro ou a remoção completa do cão do sistema, caso o usuário considere necessário.

A funcionalidade de identificação do cão está disponível na página inicial (**Home**) da aplicação. Para realizar a busca, o usuário deve fornecer uma imagem do cão, seja capturando uma fotografia diretamente pela câmera do dispositivo móvel ou selecionando uma imagem existente na galeria. Após a seleção da imagem, o usuário inicia a busca clicando no botão **"Buscar"**, conforme exemplifica o item (a) da Figura 25. A imagem é enviada para o servidor, que hospeda a API REST e os modelos de comparação, juntamente com as imagens dos cães cadastrados na base de dados do Firebase. No servidor, cada imagem da base de dados é comparada com a imagem de entrada utilizando os modelos de redes neurais. Ao final do processo, o servidor retorna para a aplicação móvel os cinco cães com maior similaridade à imagem fornecida. A aplicação, então, apresenta os resultados ao usuário, indicando o grau de similaridade (alto ou baixo) para cada um dos cinco candidatos, conforme demonstra o item (b) da Figura 25.

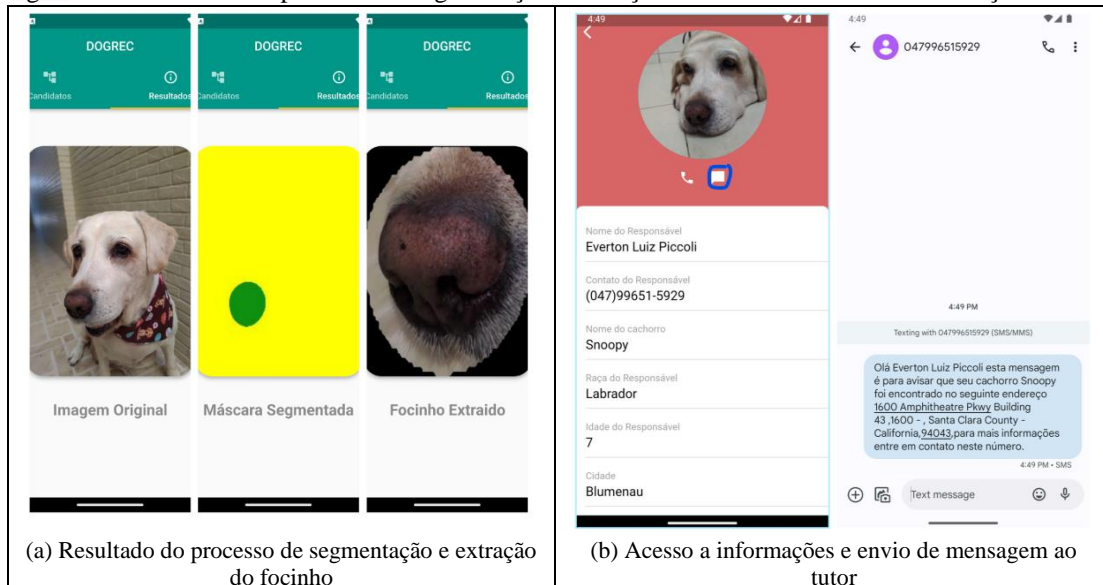
Figura 25 – Exemplo de busca e resultados do reconhecimento do cachorro



Fonte: elaborado pelo autor.

Após o término da busca, os candidatos são exibidos ao usuário, juntamente com a aba "Resultados", que permite visualizar o processo de segmentação do focinho, conforme ilustrado na Figura 26 item (a). Cada candidato apresentado possui um botão que, ao ser acionado, redireciona o usuário para uma página contendo informações detalhadas sobre o cão, incluindo o nome do tutor e seu telefone de contato. Para notificar o tutor, o usuário pode clicar no ícone de mensagem. Essa ação gera uma mensagem automática, a ser enviada ao tutor, informando a localização geográfica onde o cão foi encontrado, conforme exemplificado na Figura 26 item (b).

Figura 26 – Resultado do processo de segmentação e extração do focinho e acesso e notificação ao tutor



Fonte: elaborado pelo autor.

Por fim, pode-se concluir que a aplicação móvel desenvolvida apresenta um conjunto robusto de recursos que a tornam uma ferramenta importante para identificação e localização de cães. A sinergia entre interface amigável, banco de dados híbrido e processamento de imagem baseado em redes neurais siamesas oferece uma experiência completa e eficiente para os usuários. A interface, desenvolvida em Flutter, garante navegação intuitiva e visualmente agradável, tornando o processo de cadastro, busca e notificação simples e acessível a todos os públicos. A combinação de Firebase e SQLite no banco de dados garante a sincronização online e a disponibilidade offline dos dados, tornando a aplicação confiável em diferentes situações. A utilização de modelos de redes neurais siamesas para análise de similaridade entre imagens de focinhos garante alta precisão na identificação dos animais, mesmo com variações de ângulo, iluminação e idade. A funcionalidade de segmentação do focinho aumenta ainda mais a robustez do sistema, focando na área mais relevante para o reconhecimento. Contudo, a integração com a localização geográfica e a possibilidade de enviar notificações automáticas para os tutores tornam a aplicação uma ferramenta relevante para a localização de animais perdidos, contribuindo para o reencontro rápido e seguro com seus donos.

4 CONCLUSÕES

O trabalho desenvolvido teve como proposta disponibilizar uma aplicação capaz de identificar cachorros através do uso de foto-identificação, considerando as marcas naturais presentes no focinho, buscando de forma geral auxiliar na identificação de cachorros encontrados na rua (abandonados ou perdidos por seus donos). Esta pesquisa teve como principal objetivo avaliar a eficácia do uso de redes neurais convolucionais para extrair características do focinho do cachorro e realizar o reconhecimento (similaridade) através do uso de redes siamesas, disponibilizando uma aplicação simples e intuitiva para uso do usuário.

Os resultados se deram pelo treinamento de dados coletados durante a pesquisa, onde foram utilizadas duas técnicas a fim de buscar o melhor resultado para o trabalho proposto. A primeira técnica utilizada consistia no uso de CNN para extrair características e em cima destas características utilizar classificadores pré-treinados para validar os dados e realizar o reconhecimento do cachorro. Neste processo, as redes neurais VGG16 e VGG19 alcançaram excelentes resultados, porém, devido a quantidade limitada de dados utilizadas para o treinamento (30 imagens apenas), acreditou-se que o uso de classificadores para esta tarefa não seria o ideal. Com isso, utilizou-se o algoritmo **One-shot learning** em conjunto com as redes siamesas, alcançando um resultado promissor com a rede VGG19, (99.80%) no conjunto de treino e (99.30%) no conjunto de teste.

Em relação aos trabalhos correlatos, Tu *et al.* (2018) propuseram um aplicativo **desktop** para identificar a raça de um cachorro e, em seguida, reconhecer o indivíduo utilizando partes do seu corpo. Já Bhavani *et al.* (2019) focaram no desenvolvimento de um aplicativo Android apenas para reconhecimento de raça, enquanto Vaidya *et al.* (2022) buscaram desenvolver um aplicativo desktop também voltado somente para reconhecimento de raças, com a adição de busca por faces humanas e tentativas de associá-las a raças caninas.

Diante dos resultados alcançados e apresentados, conclui-se que a pesquisa atingiu seu objetivo de reconhecer cachorros utilizando redes siamesas para calcular a similaridade entre seus focinhos e distingui-los. Uma das grandes vantagens do uso de redes siamesas reside na capacidade de reconhecer novos indivíduos sem a necessidade de retrainar o modelo. No entanto, vale ressaltar que, apesar das assertividades alcançadas, o modelo ainda precisa de mais treinamento para aprimorar a precisão dos resultados e alcançar resultados ainda mais promissores. Para trabalhos futuros, acredita-se que esta pesquisa possa auxiliar o estudo de identificação de outros animais, além de cachorros.

A solução proposta possui certas limitações das quais pode se destacar a conexão da aplicação cliente com o servidor, que é realizada de forma local. Para resolver esta limitação, seria necessário hospedar o servidor em um serviço na nuvem para que a aplicação cliente conseguisse acessar o servidor de qualquer lugar. Outra limitação encontrada durante o desenvolvimento foi a amostra de dados coletada, que ao se apresentar limitada devido ao baixo número de imagens alcançadas (30 imagens), acabou dificultando a pesquisa devido a não ser a amostragem ideal para o treinamento de uma rede neural, levando em conta um cenário real de uso. Para resolver esta questão de amostra seria proposto outras maneiras de coleta de dados sem ser apenas pelo uso de formulários.

Por fim, as possíveis extensões para este trabalho são: 1) o uso de novas redes neurais convolucionais para reconhecimento do focinho do cachorro, analisando a capacidade das mesmas para extração de características; 2) o uso de classificadores mais avançados para treinar as características e realizar o reconhecimento do cachorro; 3) aumento da área de estudo para face inteira do cachorro, considerando outras partes chave além do focinho para o reconhecimento do cão; 4) realizar a disponibilização de novos campos no cadastro dos cachorros como peso, altura, feridas ou marcas de nascença, afim de ampliar o histórico de informações do cachorro e ajudar a distingui-lo de cachorros semelhantes.

REFERÊNCIAS

ALVES, A. J. S. Abandono de cães na América Latina: revisão de literatura. **Revista de Educação Continuada em Medicina Veterinária e Zootecnia do CRMV-SP**, v. 11, n. 2, p. 34-41, 1 jul. 2013.

BHAVANI, D. *et al.* **Dog Breed Identification Using Convolutional Neural Networks on Android**. **CVR Journal of Science and Technology**, [s.l.], v. 17, n. 1, p. 62-66, dez. 2019.

CLAPPIS, A. M. **Uma introdução as redes neurais convolucionais utilizando o Keras**: Saiba como funciona uma CNN através desse exemplo com o dataset MNIST, [s.l.], 2019. Disponível em: < <http://medium.com/data-hackers/uma-introdu%C3%A7%C3%A3o-as-redes-neurais-convolucionais-utilizando-o-keras-41ee8dcc033e> >. Acesso em: 04 jun. 2024.

DUTT, A. **Siamese Networks Introduction and Implementation: Learn how to achieve good accuracy on a classification task with just a few samples per class and imbalanced distribution of classes**, [S.L.], 2021. Disponível em: < <http://towardsdatascience.com/siamese-networks-introduction-and-implementation-2140e3443dee> >. Acesso em: 20 jun. 2024.

FERNANDES, C. **O que ensinamos aos nossos filhos quando abandonamos animais nas ruas**: Ter um pet é uma excelente oportunidade para as crianças desenvolverem maior senso de responsabilidade, solidariedade e afeto, [s.l.], 2019. Disponível em: <<http://www.metropoles.com/e-ducacao/o-que-ensinamos-aos-nossos-filhos-quando-abandonamos-animais-nas-ruas>>. Acesso em: 03 jun. 2024.

HE, K. et al. **Deep Residual Learning for Image Recognition**. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. Proceedings... Las Vegas: IEEE, 2016. p. 1-12

LOGUNOVA, I. **A Guide to One-Shot Learning**, [s.l.], 2022. Disponível em: <<http://serokell.io/blog/nn-and-one-shot-learning>>. Acesso em: 19 jun. 2024.

MARQUES, V. G. O. **Avaliação do desempenho das redes neurais convolucionais na detecção de ovos de esquistossomose**. 2017. 49 f. Monografia (Curso de Bacharelado em Engenharia da Computação) – Universidade Federal de Pernambuco – UFPE, Recife.

MELO, J. **Você sabia que o nariz de cachorro funciona como uma impressão digital? Entenda mais sobre essa curiosidade!** [s.l.], 2020. Disponível em: <<http://www.patasdacasa.com.br/noticia/voce-sabia-que-o-nariz-de-cachorro-funciona-como-uma-impressao-digital-entenda-mais-sobre>>. Acesso em: 03 jun. 2024.

MISHRA, M. **Convolutional Neural Networks, Explained**, [s.l.], 2020. Disponível em: <<http://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>>. Acesso em: 04 jun. 2024.

MOREL, L. **Com abrigos lotados e sem espaço de resgate, leitor quer novo lar para cãozinho**: Marcus lamenta não ter condições de cuidar do filhote que ele recolheu e mesmo assim, não ter onde deixá-lo, Campo Grande, 2022. Disponível em: <<http://www.campograndenews.com.br/direto-das-ruas/com-abrigos-lotados-e-sem-espaco-de-resgate-leitor-quer-novo-lar-para-caozinho>>. Acesso em: 03 jun. 2024.

NAG, R. **A Comprehensive Guide to Siamese Neural Networks**. [S.L.], 2022. Disponível em: <<http://medium.com/@rinkinag24/a-comprehensive-guide-to-siamese-neural-networks-3358658c0513>>. Acesso em: 06 jun. 2024.

PUENTE, B. **Brasil tem quase 185 mil animais resgatados por ONGs, diz instituto**: Cerca de 60% deles foram salvos de situações de maus-tratos, Rio de Janeiro, 2022. Disponível em: <<http://www.cnnbrasil.com.br/nacional/brasil-tem-quase-185-mil-animais-resgatados-por-ongs-diz-instituto/>>. Acesso em: 03 jun. 2024.

RONNEBERGER, O., FISCHER, P., BROX, T. **U-Net: Convolutional Networks for Biomedical Image Segmentation**. In: Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science (), vol. 9351. Springer, Cham. https://doi.org/10.1007/978-3-319-24574-4_28

ROSEBROCK, A. **Contrastive Loss for Siamese Networks with Keras and TensorFlow**, [S.L.], 2021. Disponível em: <<http://pyimagesearch.com/2021/01/18/contrastive-loss-for-siamese-networks-with-keras-and-tensorflow/>>. Acesso em: 20 jun. 2024.

SIMONYAN, K.; ZISSERMAN, A. **Very deep convolutional networks for large-scale image recognition**. In: 3rd International Conference on Learning Representations (ICLR 2015), 2015. Proceedings... SAN DIEGO, ICLR, 2015. p. 1 – 14

SRIKARAN. **Exploring Siamese Networks for Image Similarity using Contrastive Loss**, [S.L.], 2023. Disponível em: <<http://medium.com/@hayagriva99999/exploring-siamese-networks-for-image-similarity-using-contrastive-loss-f5d5ae5a0cc6>>. Acesso em: 22 jun. 2024.

SZEGEDY, C. et al. **Rethinking the Inception Architecture for Computer Vision**. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. Proceedings... Las Vegas: IEEE, 2016. p. 1-10

TU, X. et al. **Transfer Learning on Convolutional Neural Networks for Dog Identification**. In: 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), 2018. Proceedings... Beijing: IEEE, 2018. p. 1-4

VAIDYA, P. et al. **A Novel Dog Breed Identification using Convolutional Neural Network**. *PriMera Scientific Engineering*, [s.l.], v. 2, n. 1, p. 16-21, dez, 2022.