Revista de Informática Teórica e Aplicada - RITA - ISSN 2175-2745
Vol. 32, Num. 1 (2025) 158-165

RESEARCH ARTICLE

YOLO and *CNN* for Cat Detection and Recognition

YOLO e *CNN* para detecção e reconhecimento de gatos

Dayllon V. X. Lemos[1]*, Humberto J. Longo[1], Márcia R. Cappelle[1], Mariana D. X. S. Santos[1], Ronaldo Costa[1], Sanderson Macedo[2]

Resumo: This paper proposes a processing architecture for recognizing domestic felines in videos and images based on facial features. Some photos of the cats to be identified in the videos and images need to be collected in advance. A key aspect of this study is avoiding the need to retrain the model whenever the set of cats to be identified changes. The architecture involves five processing stages: video processing, *YOLOv8* for cat detection, *InceptionV3* for feature extraction, *KNN* for database searching, and voting process to improve classification. Transfer learning and fine-tuning techniques are employed to improve performance. The proposed method achieves high accuracy, offering a scalable and non-invasive solution for cat recognition, which can contribute to urban management, cat loss prevention, and medical insurance control for domestic cats, among other applications.
Keywords: Cat – Detection – Recognition – YOLO – CNN – KNN

Resumo: Este artigo propõe uma arquitetura de processamento para o reconhecimento de felinos domésticos em vídeos e imagens com base em características faciais. Algumas fotos dos gatos a serem identificados nos vídeos e imagens precisam ser coletadas com antecedência. Um aspecto-chave deste estudo é evitar a necessidade de treinar o modelo novamente sempre que o conjunto de gatos a serem identificados mudar. A arquitetura envolve cinco etapas de processamento: processamento de vídeo, *YOLOv8* para detecção de gatos, *InceptionV3* para extração de características, *KNN* para busca no banco de dados e um processo de votação para melhorar a classificação. Técnicas de *Transfer learning* e *fine-tuning* são empregadas para melhorar o desempenho. O método proposto alcança alta precisão, oferecendo uma solução escalável e não invasiva para o reconhecimento de gatos, que pode contribuir para o manejo urbano, prevenção de perda de gatos e controle de seguros de saúde para gatos domésticos, entre outras aplicações.
Palavras-Chave: Gato – Detecção – Reconhecimento – YOLO – CNN – KNN

[1] *Instituto de Informática, Universidade Federal de Goiás, Brazil*
[2] *Instituto Federal de Goiás, Brazil*
***Corresponding author**: dayllonxavier@gmail.com
DOI: http://dx.doi.org/10.22456/2175-2745.143684 • **Received:** 30/10/2024 • **Accepted:** 02/12/2024
CC BY-NC-ND 4.0 - This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

1. Introdução

Segundo a *International Fund for Animal Welfare – IFAW* [1] (uma organização sem fins lucrativos que ajuda animais), a população global de gatos domésticos é estimada em cerca de 600 milhões e continua a crescer. Esse fato torna o reconhecimento de gatos cada vez mais relevante [2], com potencial para facilitar a gestão urbana, a prevenção de perdas e/ou extravios de animais, e o controle do seguro médico desses *felinos*. Atualmente, o reconhecimento de gatos é realizado por meio de dois métodos principais: permanentes e não-permanentes. Os métodos permanentes incluem tatuagens, etiquetas nas orelhas e implantação de microchips, enquanto os métodos não-permanentes envolvem identificação por rádio frequência (*RFID – Radio Frequency Identification*) e biometria. Contudo, a técnica de identificação permanente é invasiva e pode causar no animal intercorrências como infecções, hemorragias, sépsis e até mesmo câncer [3]. Além disso, esses métodos não são totalmente confiáveis e propensos a atividades fraudulentas, como a troca, a duplicação e a falsificação dos chamados números de identificação únicos marcados no corpo do animal [4].

No que se refere às técnicas não-permanentes, a *RFID* não oferece robustez suficiente para garantir um nível significativo de segurança para os animais de estimação [5]. Por outro lado, a biometria oferece uma solução natural e confiável para certos aspectos da gestão da identidade, utilizando esquemas automatizados para reconhecer indivíduos com base nas suas

características fisiológicas ou na aparência fenotípica do animal [6]. O sistema biométrico visual animal é baseado no reconhecimento de padrões. Esse procedimento geralmente envolve a aquisição de dados biométricos de um indivíduo, a extração de um conjunto de características desses dados, a comparação dessas características com as armazenadas em uma base de dados e a execução de uma ação conforme o resultado da comparação.

A identificação biométrica é realizada por meio de imagens ou vídeos. Nesse sentido, *Convolutional Neural Network* (*CNN*), um tipo de rede neural de *Deep Learning* (*DL*), representa uma potencial no reconhecimento de imagens. Elas frequentemente servem como o motor por trás da categorização de imagens e são amplamente utilizadas para a análise visual de imagens, detecção e reconhecimento de objetos [7].

Entretanto, uma *CNN*, quando inicializada com pesos aleatórios, pode necessitar de um tempo de treinamento longo e exigir uma grande quantidade de dados para começar a apresentar resultados satisfatórios [8]. Nesse sentido, *Transfer Learning* é uma técnica que aproveita os pesos de um modelo pré-treinado como ponto de partida para o retreino em novos conjuntos de dados específicos ou de diferentes domínios [9]. Assim, é possível obter um compartilhamento de conhecimento entre uma rede e outra.

Esse procedimento é conhecido como *Fine-Tuning*. Ele envolve várias etapas, como o congelamento de algumas camadas ou pesos do modelo base para preservar o conhecimento existente, a substituição da camada final do modelo por uma nova mais adequada à tarefa desejada, e o retreino em um conjunto de dados específico que visa otimizar o desempenho em uma determinada tarefa [10].

O presente trabalho propõe uma arquitetura de processamento para o reconhecimento de felinos domésticos em vídeos e fotos baseando-se em suas caraterísticas faciais. Os gatos a serem identificados devem ser conhecidos a priori e algumas poucas fotos de cada um deles devem ser coletadas com antecedência. Um dos principais objetivos deste estudo é evitar a necessidade de retreinar qualquer modelo utilizado, sempre que o conjunto de gatos a serem identificados for alterado. Nesse contexto, a Seção 2 apresenta um breve resumo sobre os trabalhos relacionados ao reconhecimento facial/visual de gatos e animais de estimação. A Seção 3 apresenta a arquitetura proposta, explicando cada uma das suas cinco etapas do processamento. Os resultados obtidos são expostos na Seção 4 e, por fim, na Seção 5 as considerações finais e propostas para trabalhos futuros são exibidas.

2. Trabalhos relacionados

Alguns pesquisadores têm explorado a tarefa de reconhecimento de animais domésticos (*pets*). A fim de detectar a presença dos *pets* nas imagens, Mukai et al. [11] utilizaram 1000 imagens de gatos do *dataset* [12], 600 imagens de cachorros da base de dados [13] e outras 3000 imagens sem nenhum *pet*. Ao utilizar o *AdaBoost Algorithm* com *Haar-like Features*, foi alcançada uma precisão de 75,7% e um *recall* de 96,6% no *dataset* de gatos. Já no *dataset* de cachorros, os resultados foram precisão de 90,8% e *recall* de 98,3%.

Outro trabalho que utiliza imagens de cachorros é o de Kumar [5]. Nesse estudo, foram empregadas 560 imagens, das quais 400 foram destinadas ao treino. Os autores realizaram a comparação entre duas imagens para verificar se elas retratavam o mesmo animal. Os algoritmos aplicados para extrair as características foram *Principal Component Analysis* (*PCA*), *Local Discriminant Analysis* (*LDA*) e *Independent Component Analysis* (*ICA*). Esse trabalho alcançou uma acurácia de 94,86%. Além disso, Zhang [14] apresentou uma abordagem semelhante, utilizando 14808 imagens de 509 gatos diferentes. Ele aplicou a *Siamese Network* [15] com a *VGG16* [16] para verificar se duas imagens retratavam um mesmo felino. A acurácia obtida foi de 72,91%.

Fan et al. [2] utilizaram um *dataset* de 30 gatos e as técnicas de *Mel-Frequency Cepstrum Coefficients* (*MFCC*) e *Gaussian Mixture Model* (*GMM*) para a extração e tratamento das características. Além disso, também foi utilizado o algoritmo *K-Means*, com $K = 4$, para inicializar os parâmetros do modelo. Dos 30 gatos, apenas 5 foram identificados incorretamente, resultando assim em uma acurácia de 83,3%.

Lin e Kuo [17] empregaram 1500 imagens de 150 gatos diferentes para fazer o reconhecimento. A detecção foi conduzida por meio da arquitetura *Faster R-CNN*, enquanto a classificação foi realizada pelo algoritmo *Support Vector Machine (SVM)*. Nesse trabalho, a acurácia alcançada foi de 94,1%. Além desse trabalho, Cahyo et al. [18] empregaram a arquitetura de *CNN* chamada *Xception* no banco de dados [19], que contém 2400 imagens de 12 gatos diferentes. Os testes computacionais mostraram uma acurácia de 92,5%.

As estratégias de reconhecimento empregadas pela maioria dos trabalhos apresentados nesta seção requerem um conjunto fixo de gatos conhecido antes do treinamento. Isso significa que os modelos precisam ser retreinados caso esse conjunto seja modificado.

3. Abordagem proposta

Seja S o conjunto de gatos (classes) que devem ser reconhecidos pela abordagem a ser apresentada. Durante o desenvolvimento do trabalho buscou-se alcançar os três seguintes objetivos:

- os gatos de S devem ser detectados e reconhecidos;
- os gatos que não pertencem ao conjunto S devem ser detectados e classificados como "outra"; e
- não deve ser necessário retreinar nenhum modelo caso algum gato seja adicionado ou removido do conjunto S.

Para atingir estes objetivos, foi proposta a arquitetura de processamento sequencial ilustrada na Figura 1. Cada etapa dessa arquitetura será detalhada ao longo desta seção. A Subseção 3.1 descreve o processo de extração dos *frames* do vídeo. Em 3.2 é explicada a forma de detecção dos gatos e em 3.3 como cada gato detectado é convertido em um *embedding*. Na Subseção 3.4 é mostrado o procedimento para executar a

inferência da classe considerando-se o conjunto de gatos S. Em 3.5 é exposto um processamento final realizado para se obter uma melhor precisão da classificação considerando-se *frames* anteriores. Por fim, na Subseção 3.6 são explicadas as alterações necessárias caso seja modificado o conjunto S de gatos a serem reconhecidos.

Nesta seção será abordado apenas o reconhecimento de gatos domésticos em vídeos. Contudo, é fácil adaptar a arquitetura desenvolvida para o reconhecimento em imagens. Para isso, basta considerar a imagem como sendo apenas um único *frame* e remover o pós-processamento final da Subseção 3.5.

3.1 Vídeo

O reconhecimento de gatos em vídeos é feito através do processamento individual de cada um de seus *frames*. Para isso, uma quantidade de *frames* processados a cada segundo deve ser definido. É evidente que uma baixa taxa de *frames* por segundo, embora necessite de menos processamento, pode desencadear resultados ruins no reconhecimento. Contudo, taxas muito altas podem exigir um *hardware* mais potente e preparado. Além disso, elas podem não agregar muita informação no processo de reconhecimento, uma vez que um *frame* se torna muito similar ao seu *frame* anterior. Nesse sentido, o valor de 20 *frames* processados a cada segundo foi empregado nos testes desenvolvidos por este trabalho. Por outro lado, observou-se também que taxas de 10 e 15 *frames* por segundo apresentaram bons resultados.

Cada *frame* processado é convertido para o tamanho $640 \times 640\,px$ e encaminhado à etapa *YOLO* do processamento expresso na Figura 1. É evidente que essa conversão pode distorcer a imagem original, mas o impacto desta distorção não afetou de forma considerável o processo completo de reconhecimento dos gatos nos testes computacionais realizados.

3.2 *YOLO*

A etapa de detecção dos gatos consiste em identificar os gatos que estão contidos no *frame*, recebido pela etapa anterior, e para cada um dos gatos encontrados, uma caixa delimitadora (*bounding box*) é criada. Cada caixa é então recortada da imagem e encaminhada, juntamente com a posição do *bounding box*, para a próxima etapa, denominada Rede Neural (Figura 1). As etapas posteriores também sempre enviam para a etapa à sua frente a posição da caixa delimitadora, até se alcançar a última etapa. Como esse procedimento é comum a todas as próximas etapas, ele será omitido do restante do texto. Um ponto a se atentar é que esta etapa recebe um *frame* como entrada e pode gerar como saída várias imagens recortadas, uma para cada *bounding box*.

Para realizar este processo de detecção dos gatos optou-se pelo uso do *YOLOv8* (*You Only Look Once*, versão 8), que é um modelo de detecção de objetos que utiliza redes neurais convolucionais para identificar e localizar objetos em imagens e vídeos [20]. O *YOLOv8* foi treinado sobre o *dataset COCO (Common Objects in Context)* que contém mais de 200 mil imagens, com cerca de 1,5 milhão de instâncias de objetos anotadas com caixas delimitadoras e compreende 80 categorias, sendo uma delas a classe gato (*cat*) [21]. Além disso, o *YOLOv8* está disponível em diversos tamanhos: *nano*, *small*, *medium*, *large* e *extra-large*. Cada um deles possui um balanço entre a acurácia da detecção e a quantidade de processamento necessária.

Nos testes realizados neste trabalho, empregou-se a versão *nano* do *YOLOv8*, que embora seja a menor em quantidade de parâmetros e possua a menor taxa de acurácia dentre os outros tamanhos é a que exige menos processamento e apresentou bons resultados na detecção.

Um treinamento customizado para o *YOLOv8 nano* foi conduzido, considerando-se apenas a classe "gato", com o *dataset Cats Dataset* [22], o qual possui 1159 imagens distintas de gatos domésticos. Com o auxilio da técnica de Aumento de Dados (*Data Augmentation*), foi possível obter 2433 imagens anotadas para treino e 232 para validação. Assim, o modelo treinado obteve uma precisão e um *mAP50* de 98%. Contudo, quando aplicado ao *pipeline* da Figura 1, o modelo treinado obteve resultados inferiores ao *YOLOv8 nano* pré-treinado com as imagens do *COCO dataset*. Por esse motivo, optou-se por escolher o segundo modelo para realizar esta etapa de detecção.

3.3 *CNN*

Esta etapa consiste na geração de um *embedding* para cada imagem recortada recebida da camada anterior. O *embedding* de uma imagem é uma representação vetorial compacta das suas principais características (assinatura da imagem).

Para isso foi empregado o uso de uma Rede Neural Convolucional *(Convolutional Neural Network – CNN)*. A arquitetura da *CNN* escolhida foi a *InceptionV3* sem a camada densa (apenas as camadas convolucionais foram usadas). A *InceptionV3* é uma arquitetura *CNN* desenvolvida pelo *Google* [23] e treinada na base de dados *ImageNet* [24], sendo principalmente utilizada para a classificação de imagens. Portanto, cada imagem advinda da etapa anterior é redimensionada para $299 \times 299\,px$ e processada pela *InceptionV3*, que por sua vez retorna um vetor com 2048 posições (*embedding*).

Com relação ao treinamento do modelo, foi realizado um *Transfer Learning* e *Fine-Tunning* do *InceptionV3* treinado com a base *ImageNet*. No *Fine-Tunning* todas as camadas após a "conv2d_45" foram descongeladas, permitindo o seu retreino. Optou-se por não alterar os pesos das primeiras camadas convolucionais, umas vez que elas são responsáveis por identificar características mais básicas da imagem, como retas e pontos.

Para o retreino do modelo foi necessário reunir um conjunto de imagens de gatos, sendo que para cada gato várias imagens daquele mesmo felino eram necessárias. A base de dados utilizada consistiu da junção, tratamento e pré-processamento dos *datasets CatFace* [25] e *CAT Database* [12]. O primeiro contém 1850 imagens e o segundo 13879. Após selecionadas, foi possível extrair da segunda base 11956 fotos de 481 gatos diferentes e do primeiro *dataset* 1451 fotos de 6 outros gatos. Para essa seleção, foram descartadas ima-

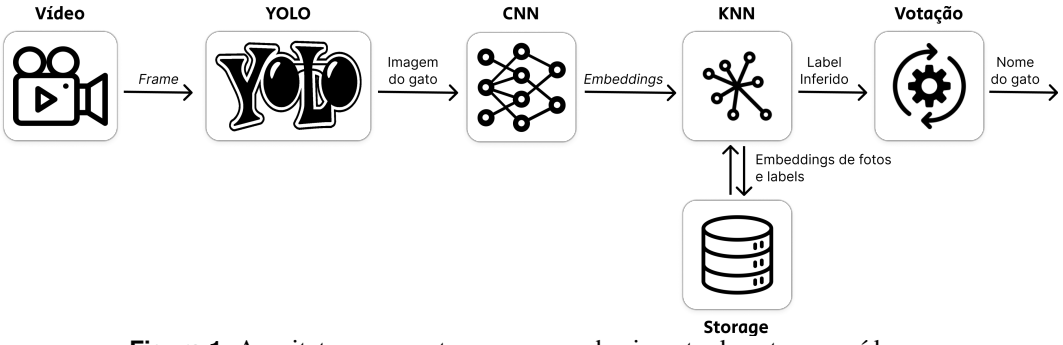

Figura 1. Arquitetura proposta para o reconhecimento de gatos em vídeos.

gens com qualidade muito ruins, com mais de dois gatos, ou que não exibiam a face do animal. Em adição, foram removidas as classes que restaram 5 ou menos imagens. Além disso, para cada imagem foi realizado o recorte apenas da parte onde o gato se encontrava, para se ter uma menor interferência do fundo do ambiente da foto, focando a *CNN* em extrair as características do animal.

Assim, cada um dos gatos se tornou uma classe a ser classificada pela *CNN* e a função de erro escolhida foi a "Categorical Crossentropy". Além disso, foi adicionada uma última classe denominada de "outra", que consistiu da seleção e pré-processamento de 500 imagens advindas do *dataset Cat Individual Images* [26] (cada uma de um gato diferente). Essa classe foi adicionada com o objetivo de agir como uma resposta negativa da *CNN* para quando um gato detectado não pertencer ao conjunto S. No total, foram utilizadas 13907 fotos de 487 classes diferentes, incluindo a classe "outra".

O base de dados final foi dividida de forma aleatória, dentro de cada classe, em 80% treino e 20% validação. Em incremento, durante o processo de treino, foi aplicada a técnica de *Data Augmentation* considerando-se:

- Horizontal e Vertical *Flip*;
- *Rotation* de até 30º;
- *Width Shift* de 0,2;
- *Height Shift* de 0,2;
- *Shear* de 0,2;
- *Zoom* de 0,15; e
- *Brightness* no intervalo de $[0{,}8, \ldots, 1{,}2]$.

O tamanho do *batch* empregado foi de 32 e o *learning rate* inicial foi fixado em 10^{-5}. Também foi empregado um *ReduceLROnPlateau* com fator e persistência definidos como 0,1 e 5, respectivamente. A acurácia obtida foi de 94% e levou 26 épocas de treinamento, ponto em que não houve mais melhorias.

3.4 *KNN*

Esta etapa consiste na inferência do *label* (nome do gato) a partir do *embedding* recebido da etapa anterior. Para se alcançar esse objetivo foi utilizado o algoritmo *KNN* (*K-Nearest Neighbors*) [27]. Esse algoritmo considera cada *embedding* como um ponto no espaço multidimensional (no caso deste trabalho são 2048 dimensões – tamanho do *embedding*). A partir disso ele classifica a classe de um novo ponto com base na maioria das classes de seus K vizinhos mais próximos no espaço de características.

Inicialmente, para que o *KNN* possa funcionar é necessário especificar alguns pontos já classificados com seus respectivos *labels*. Esses pontos estão representados na Figura 1 pelo recurso gráfico *Storage*.

No contexto deste projeto, o *Storage* consiste de um pequeno banco de dados vetorial com os *embeddings* (e *labels*) de algumas imagens de cada gato do conjunto S. *Data Augmentation* também foi empregado nessas imagens, observando uma melhora na precisão durante o reconhecimento. A geração dos *embeddings* dessas fotos foi realizada utilizando-se o mesmo modelo treinado da *CNN* descrito na Subseção 3.3.

Além disso, uma vez que o reconhecimento exige uma resposta negativa, isto é, a indicação da classe "outra" quando o gato detectado não pertencer ao conjunto S, torna-se necessário a adição de outros *embeddings* com o *label* "outra" no *Storage*. Os *embeddings* dessa classe são formados pelo processamento da *CNN* em imagens de vários gatos distintos (cada foto de um felino diferente), como as pertencentes ao *dataset Dogs vs. Cats* [26]. Sendo assim, a quantidade de *embeddings* da classe "outra" altera diretamente a precisão do reconhecimento. Um intervalo de proporção que mostrou bons resultados foi de 15% a 25% dos *embeddings* do *Storage* pertencerem à classe "outra".

A partir do *Storage* com os dados já armazenados, o *KNN* utilizá alguma métrica de distância para computar os K vizinhos mais próximos de um novo ponto. A métrica final utilizada foi a distância por cosseno e o valor de K foi escolhido como 1. Contudo, outras métricas e outros valores de K foram testados e seus desempenhos são detalhados na Seção 4.

Assim, sempre que um *embedding* é recebido da etapa anterior, o *KNN* usa os pontos do *Storage* e infere para ele uma *label*, que é o nome do gato representado pelo *embedding*. Essa *label* é então passada para a última etapa do processamento.

O *KNN* é um algoritmo de classificação simples e não muito preciso. Na verdade, sua acurácia varia muito de acordo com a quantidade de imagens e classes presentes no *Storage*.

Contudo, sua simplicidade e o seu fácil paralelismo levaram a sua utilização neste projeto.

3.5 Votação

Esta é a última etapa da inferência do nome do gato detectado na Subseção 3.2. Ela tem o seu desenvolvimento baseado no principio de memória e busca corrigir as inferências erradas feitas pelo *KNN*.

Sempre que um novo *label* é recebido da etapa anterior, juntamente com a posição do seu *bounding box*, é realizado um procedimento que tenta encontrar nos últimos X *frames*, para $X \in \mathbb{N}$, quais *bounding boxes* representam um mesmo gato. Para fazer isso, o cálculo do *IoU* (*Intersection over Union*) juntamente com um *threshold* de 0,7 é empregado, isto é, dois *bounding box*, de dois *frames* diferentes, são considerados pertencentes ao mesmo gato se a área de interseção deles sobre a área da união for superior a 0,7 (70%).

Assim, o *label* final inferido vai ser o que aparecer em maior quantidade para cada um dos *labels*, inferidos pelo *KNN*, dos *bounding boxes* advindos do mesmo objeto e dos últimos X *frames*. Esse procedimento funciona como uma votação entre as últimas X inferências realizadas pelo *KNN*. Assim, caso a etapa anterior falhe em inferir o *label* correto, ainda existirão X outras inferências anteriores para serem consideradas para a definição final do *label*.

A escolha do valor do parâmetro X deve variar de acordo com a escolha da quantidade de *frames* processados a cada segundo. Neste trabalho, o valor de $X = 60$ foi empregado e obteve bons resultados. Como a taxa de *frames* por segundo escolhida foi de 20, $X = 60$ indica manter memória e tomar a decisão da inferência final com base nos últimos 3 segundos.

Esse pós-processamento final torna-se interessante uma vez que a variação do posicionamento de um gato entre um *frame* e o próximo não deve ser muito grande. Além disso, o procedimento de votação, quando bem implementado, pode ser executado em tempo e espaço constantes, isto é, $\mathcal{O}(1)$.

3.6 Alterando o conjunto S

Quando o reconhecimento dos gatos é realizado em um vídeo, a etapa descrita em 3.1 extrai os *frames* dos vídeos e os entregam ao *YOLO*. Ele por sua vez faz a detecção e recorte de todos os gatos da imagem, encaminhando-os adiante (3.2). A *CNN* então processa cada imagem e gera um *embedding* de 2048 posições que é enviada ao *KNN* (3.3). Este utiliza os *embeddings* das imagens dos gatos de S armazenados no *Storage* e faz a inferência do *label* (3.4). A última etapa (3.5) aplica um procedimento de votação considerando as últimas X respostas para aquele mesmo gato. A classe que ganhar a votação é a classe escolhida como sendo o nome final do gato.

Nesse contexto, é fácil perceber que caso algum outro gato seja adicionado ao conjunto S, é necessário apenas que algumas fotos desse animal sejam coletadas, convertidas em *embeddings* e adicionadas no *Storage*. No mesmo sentido, caso algum gato seja removido do conjunto S, é preciso, para que ele pare de ser reconhecido, apenas remover os *embeddings* do *Storage* que são referentes àquele felino. Dessa forma, é possível alterar o conjunto S sem que nenhum modelo seja retreinado.

4. Resultados

O modelo treinado da *CNN*, descrito na Subseção 3.3, conseguiu obter, considerando-se as 487 classes definidas (486 gatos e a classe "outra"), uma acurácia de 94% e um erro inferior a 0,2. As Figuras 2 e 3 apresentam a evolução da acurácia e do erro durante o treinamento por 26 épocas, obtendo-se ao final uma acurácia de 99% para os dados de treinamento e 94% para os de validação, com erros de 0.02 e 0.18, respectivamente. Após a época 26, o treinamento seguiu sem evolução até a época 46, onde foi encerrado.

Contudo, a *InceptionV3* não é utilizada diretamente na inferência do *label*, apenas as suas camadas convolucionais são responsáveis por extrair boas características e encaminhá-las ao *KNN*, que decide o nome do gato. Dessa forma, a acurácia de 94% obtida pela *CNN* não indica que a arquitetura proposta vai acertar em 94% das vezes, visto que muito provavelmente os 486 gatos do treino não vão ser os gatos do conjunto S. Nesse sentido, o treinamento do modelo buscava apenas aperfeiçoar a *InceptionV3* para reconhecer características melhores com relação a faces de gatos. Assim, um treino muito intenso pode fazer a rede se aperfeiçoar nas características dos 486 felinos do treino, e não de quaisquer gatos.

Para lidar com esse problema, foi construída uma base de dados própria que não continha nenhuma imagem em comum com as que foram utilizadas no treino da *CNN*. A base consistiu de imagens de 12 gatos diferentes mais a classe "outra", sendo que 106 imagens foram convertidas em *embeddings* e adicionadas ao *Storage*. Outras 221 imagens foram processadas pelo modelo da *CNN*, após j épocas de treinamento ($1 \leqslant j \leqslant 26, j \in \mathbb{N}$), e cada *embedding* gerado foi passado para o *KNN* que realizou a inferência da classe. Assim, cada acerto de *label* era contabilizado. As métricas de distância testadas no *KNN* foram: *cosine*, *euclidean*, *hamming*, *manhattan* e *minkowski*. Os valores de K utilizados foram 1, 3 e 5. Valores de K maiores não foram empregados pois necessitariam que cada gato do conjunto S tivesse várias imagens adicionadas no *Storage*.

Uma peculiaridade ocorreu no que diz respeito às distâncias *euclidian* e *minkowski*, que apresentaram a mesma taxa de acerto. Por esse motivo, decidiu-se omitir a distância de *minkowski* nos gráficos gerados. Além disso, a métrica *hamming* também foi omitida, pois como visto na Figura 4, apresentou resultados insatisfatórios e inferiores a 50% de acurácia.

As Figuras 5 a 7 apresentam as taxas de acerto, considerando-se o modelo após cada época de treinamento e as métricas de distâncias, usando-se os valores de K do *KNN* como 1, 3 e 5, respectivamente.

A configuração que resultou em melhores resultados, conforme ilustrado na Figura 5, foi a que utiliza o modelo do *CNN* após 21 épocas de treinamento, a métrica *cosine* de distância e o valor de $K = 1$. Essa configuração de parâmetros resultou em uma taxa de acerto de 94%. Contudo, para as outras

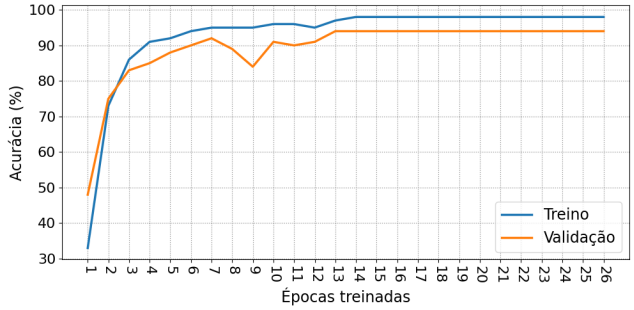

Figura 2. Acurácia do modelo por época de treinamento.

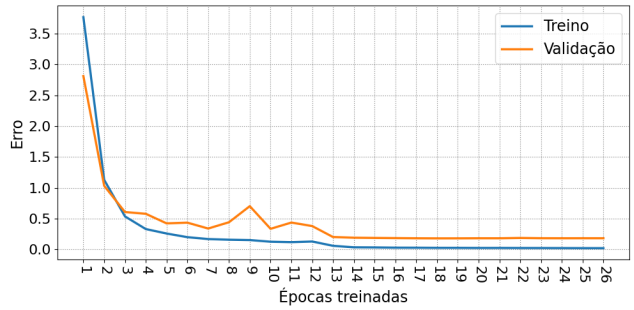

Figura 3. Erro do modelo por época de treinamento.

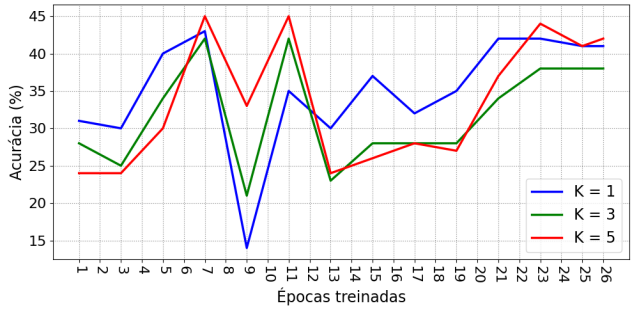

Figura 4. Acurácia por época treinada e K (métrica *hamming*).

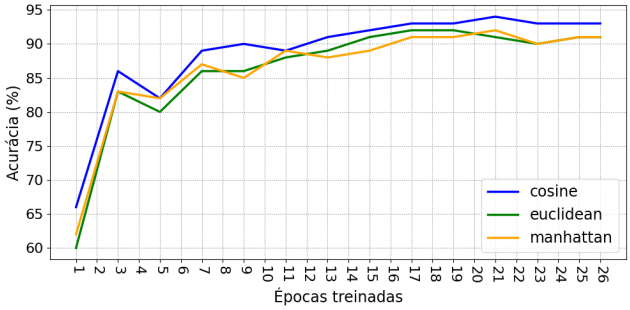

Figura 5. Acurácia por época treinada e métrica ($K = 1$).

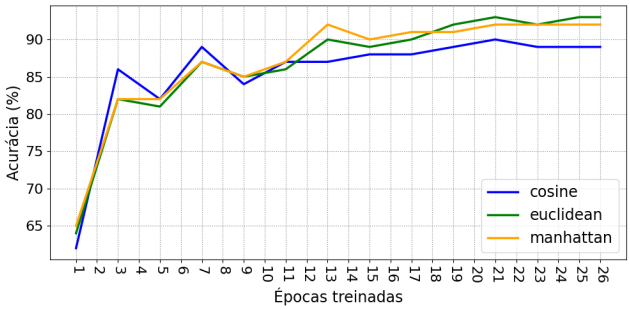

Figura 6. Acurácia por época treinada e métrica ($K = 3$).

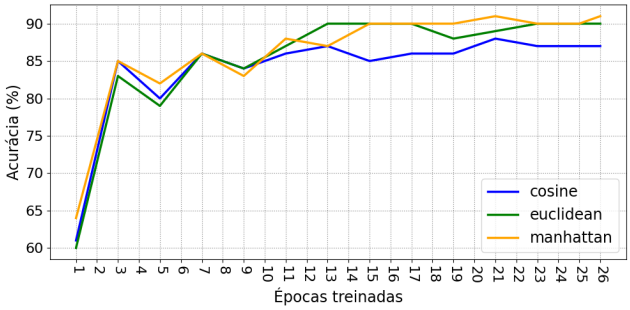

Figura 7. Acurácia por época treinada e métrica ($K = 5$).

configurações do parâmetro K (Figuras 6 e 7), os melhores resultados foram alcançados utilizando-se o modelo após 21 e 26 épocas de treinamento, respectivamente. A configuração com $K = 3$ atingiu o seu ápice com o uso da métrica *euclidean*, com acurácia de 93%, mostrando-se também uma configuração com resultados competitivos. Para o valor de $K = 5$, a melhor taxa de acerto de 91% foi obtida usando-se a métrica *manhattan*.

5. Conclusão

Neste trabalho foi apresentada uma arquitetura de processamento para o reconhecimento de gatos domésticos baseada na identificação facial desses animais. A estratégia apresentada busca reconhecer os gatos de um conjunto S possuindo a priori algumas poucas imagens desses felinos. Um diferencial dessa abordagem é o fato do conjunto S poder ser alterado sem a necessidade de retreinar nenhum modelo. Os testes demonstraram uma taxa de 94% de acerto do reconhecimento quando utilizado a métrica de distância *cosine*, o valor de $K = 1$ e o modelo da *CNN* após 21 épocas de treinamento. Além disso, foi implementado um procedimento de pós-processamento para corrigir inferências incorretas.

Acredita-se que resultados melhores podem ser obtidos caso o modelo do *YOLO* seja treinado especificamente para reconhecer apenas a face do gato, em vez de todo o corpo. A construção de um *dataset* para se obter essa alteração está em desenvolvimento. Como trabalho futuro, também está previsto o desenvolvimento de uma solução para reconhecimento em tempo real. Para isso, acredita-se que seja necessário o processamento de menos *frames* a cada segundo e a troca da *InceptionV3* por outra rede mais rápida, como a *MobileNet* [28].

Agradecimentos: À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Contribuições dos autores

Dayllon Vinícius Xavier Lemos: Principal author, responsible for writing the majority of the article, as well as developing the tests, algorithms, and implementations utilized. He also conceptualized the processing architecture presented in the study, ensuring its alignment with the research objectives, and handled the identification and preprocessing of the datasets,

addressing challenges related to data quality and integration; Mariana Dourado Ximenes de Sena Santos: Conducted a systematic mapping of methods for detecting and recognizing cats in videos and photos. Additionally, she wrote the ”2. Related Work”section, providing a detailed overview of the existing literature and positioning the study within the broader research context; Humberto José Longo and Márcia Rodrigues Cappelle: Contributed significantly by refining the article's writing and improving its clarity, ensuring consistency in style and terminology throughout the manuscript. They also provided valuable input in enhancing the presentation of the figures, including adjustments to improve readability and visual impact. Furthermore, they were instrumental in organizing and structuring the LaTeX document to maintain a professional and cohesive format; Sanderson Macedo and Ronaldo Costa: Reviewed the text for linguistic quality, verified technical accuracy, and provided suggestions to enhance the coherence and precision of the content.

References

[1] WELFARE, I. F. for A. *Cats*. 2024. ⟨https://www.ifaw.org/international/animals/cats⟩. Accessed: 2024-07-22.

[2] FAN, Y.; YANG, C.-C.; CHEN, C.-T. Cat face recognition based on mfcc and gmm. In: *2021 6th International Conference on Image, Vision and Computing (ICIVC)*. [S.l.: s.n.], 2021. p. 81–84.

[3] KUMAR, S.; SINGH, S. Cattle recognition: A new frontier in visual animal biometrics research. *Proceedings of the National Academy of Sciences, India Section A: Physical Sciences*, v. 90, 05 2019.

[4] M.SHEN et al. Review of monitoring technology for animal individual in animal husbandry. *Nongye Jixie Xuebao/Transactions of the Chinese Society for Agricultural Machinery*, v. 45, n. 10, p. 245, 251, 2014.

[5] KUMAR, S. K. S. S. Biometric recognition for pet animal. *Journal of Software Engineering and Applications*, v. 7, n. 5, 2014.

[6] KüHL, T. B. H. S. Animal biometrics: quantifying and detecting phenotypic appearance. *Trends in Ecology & Evolution*, v. 28, n. 7, p. 432–441, 2013.

[7] ZHANG, Q. et al. Recent advances in convolutional neural network acceleration. *Neurocomputing*, v. 323, p. 37–51, 2019. ISSN 0925-2312. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0925231218311007⟩.

[8] ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: *2017 International Conference on Engineering and Technology (ICET)*. [S.l.: s.n.], 2017. p. 1–6.

[9] SHAHA, M.; PAWAR, M. Transfer learning for image classification. In: *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. [S.l.: s.n.], 2018. p. 656–660.

[10] TAJBAKHSH, N. et al. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, v. 35, n. 5, p. 1299–1312, 2016.

[11] MUKAI, N.; ZHANG, Y.; CHANG, Y. Pet face detection. In: *2018 Nicograph International (NicoInt)*. [S.l.: s.n.], 2018. p. 52–57.

[12] INNOVATION, D. *10,000 Cat Pictures for Science*. 2014. ⟨http://www.datainnovation.org/2014/10000-cat-pictures-for-science/⟩. Acessado em: 8 de maio de 2024.

[13] KHOSLA, A. et al. Dataset for fine-grained image categorization: Stanford dogs. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [s.n.], 2011. p. 1–2. Disponível em: ⟨https://people.csail.mit.edu/khosla/papers/fgvc2011.pdf⟩.

[14] LI, H.; ZHANG, W. Cat face recognition using Siamese network. In: LOSKOT, P. (Ed.). *International Conference on Artificial Intelligence and Intelligent Information Processing (AIIIP 2022)*. SPIE, 2022. v. 12456, p. 124562U. Disponível em: ⟨https://doi.org/10.1117/12.2659645⟩.

[15] BROMLEY, J. et al. Signature verification using a “siamese” time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, v. 07, n. 04, p. 669–688, 1993.

[16] QASSIM, H.; VERMA, A.; FEINZIMER, D. Compressed residual-vgg16 cnn model for big data places image recognition. In: *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.: s.n.], 2018. p. 169–175.

[17] LIN, T.-Y.; KUO, Y.-F. Cat face recognition using deep learning. In: *ASABE 2018 Annual International Meeting*. [s.n.], 2018. Disponível em: ⟨https://api.semanticscholar.org/CorpusID:57554357⟩.

[18] CAHYO, D. D. N.; SUNYOTO, A.; ARIATMANTO, D. Transfer learning and fine-tuning effect analysis on classification of cat breeds using a convolutional neural network. In: *2023 6th International Conference on Information and Communications Technology (ICOIACT)*. [S.l.: s.n.], 2023. p. 488–493.

[19] PARKHI, O. M. et al. Cats and dogs. In: *IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2012.

[20] ULTRALYTICS. *Ultralytics YOLO Docs*. 2024. ⟨https://docs.ultralytics.com/⟩. Acessado em: 24-ago-2024.

[21] LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: FLEET, D. et al. (Ed.). *Computer Vision – ECCV 2014*. Cham: Springer International Publishing, 2014. p. 740–755. ISBN 978-3-319-10602-1.

[22] TRAORE, M. Open Source Dataset, *Cats Dataset*. Roboflow, 2022. ⟨https://universe.roboflow.com/mohamed-traore-2ekkp/cats-n9b87⟩. Visited on 2024-08-24. Disponível em: ⟨https://universe.roboflow.com/mohamed-traore-2ekkp/cats-n9b87⟩.

[23] SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. Disponível em: ⟨http://arxiv.org/abs/1512.00567⟩.

[24] DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2009. p. 248–255.

[25] EL-CHIANG. *CatFace*. 2019. ⟨https://github.com/El-Chiang/CatFace⟩. Acessado em: 8 de maio de 2024.

[26] Kaggle. *Dogs vs. Cats*. 2024. ⟨https://www.kaggle.com/competitions/dogs-vs-cats⟩. Accessed: 2024-06-08.

[27] STEINBACH, M.; TAN, P.-N. knn: k-nearest neighbors. In: WU, X.; KUMAR, V. (Ed.). *The Top Ten Algorithms in Data Mining*. 1st edition. ed. [S.l.]: Chapman and Hall/CRC, 2009. p. 12. ISBN 9780429138423.

[28] HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. Disponível em: ⟨http://arxiv.org/abs/1704.04861⟩.