Sistema computacional para prevenção de ataques em rebanhos via detecção automática de predadores

Giovani Oliveira da Silva

SERVIÇO DE PÓS-GRADUAÇÃO DO FACOM-UFMS

Data de Depósito:

Assinatura:_____

Orientador: *Prof. Dr. Rafael Geraldeli Rossi*
Coorientador: *Dr. José Alexandre Agiova da Costa*

Defesa do Mestrado Profissional em Computação Aplicada da Faculdade de Computação da Universidade Federal do Mato Grosso do Sul, como parte dos requisitos para obtenção do título de Mestre em Computação Aplicada.

UFMS - Campo Grande
Dezembro/2020

Aos meus pais,
Valdemir e Elisabete,

Aos meus irmãos,
Gilverton e Giselli,

Rafael Rossi, José Costa.

Agradecimentos

Agradeço em primeiro lugar a Deus por me dar força e capacitação para desenvolver este projeto. Agradeço grandemente aos meus pais pelo carinho, compreensão e pelo apoio, em especial ao meu pai que estava me motivando no começo deste mestrado e não pode mais está presente em vida no final do mestrado, pois é uma vítima da COVID-19, agradeço ao meu irmão e minha irmã, por todo o incentivo e apoio que me deram.

Agradeço ao meu orientador o Prof. Dr. Rafael Geraldeli Rossi, por todas as orientações, pelas dicas, críticas, sugestões e esclarecimentos que tornaram possível a realização deste trabalho.

Agradeço ao meu coorientador o Dr. José Alexandre Agiova da Costa por estar sempre preocupado e apoiando com parcerias para o projeto.

Agradeço a todos os professores, pela paciência, dedicação e o ótimo trabalho que desempenharam durante todos esses anos, contribuindo muito para a minha formação.

Resumo

A produção de carne ovina é uma das principais frentes de desenvolvimento da agricultura familiar. Ao longo dos anos, há uma exigência maior de qualidade e regularidade na sua oferta. Entretanto, um constante problema para os produtores, e também um problema ambiental, é o ataque de predadores aos rebanhos. As formas de inibir os ataques geralmente se dão pelo recolhimento noturno dos ovinos, e ultimamente por meio de cercas elétricas ou até mesmo vigilância humana, o que muitas vezes se torna ineficiente e acarreta maiores custos de produção, podendo também acarretar na morte dos predadores. Desta forma, neste projeto foi proposto um sistema computacional de baixo custo para reconhecimento automático de predadores, visando assim a sua empregabilidade em qualquer tipo de situação (grandes empresas ou agropecuária familiar). O sistema faz uso de técnicas de visão computacional e comunicação em rede para o reconhecimento de predadores, e geração de alertas para tomada de ações de forma a inibir os ataques. Além disso, a detecção de predadores poderá ajudar tanto o produtor quanto os ambientalistas, uma vez que os ambientalistas poderão capturar e levar o predador para o seu habitat natural e o produtor evitar perdas nas produções de rebanhos. Estudos foram realizados para analisar qual estratégia seria capaz de reconhecer mais acuradamente os predadores. As avaliações das estratégias foram divididas em duas etapas: (i) avaliação de classificação multiclasse para reconhecer 6 diferentes classes de animais (onça pintada, onça parda, cão, ovino, bovino, cavalo); e (ii) avaliação de classificação binária para o reconhecimento se existe ou não um predador na imagem. Para ambas as estratégias foram utilizadas: (i) representações do tipo *Bag of Visual Words (BoVW)* em conjunto com algoritmos de aprendizado de máquina; e (ii) Redes Neurais Convolucionais (CNNs). Na avaliação multiclasse, as melhores acurácias utilizando a representação BOVW foram de 0,5041 enquanto que com o uso de CNNs, a melhor acurácia foi obtida com a arquitetura Xception em conjunto com as técnicas de *transfer learning* e aumento de imagens, proporcionando uma acurácia acurácia média

0,93 (± 0,01) e a acurácia máxima de 0,96. Já na avaliação da classificação binária para reconhecimento da classe predador ou não predador, a melhor acurácia obtida utilizando a representação BOVW foi 0,8743, enquanto que com o uso de CNNs, a melhor acurácia foi obtida pelo modelo Xception em conjunto com as técnicas de *transfer learning* e aumento de imagens, sendo a acurácia média de 0,9926, (± 0,04) e a acurácia máxima 1,00.

Palavras-chave: aprendizado de máquina, visão computacional, processamento de imagens, sistemas embarcados, redes neurais convolucionais, agropecuária, predadores.

Abstract

Sheep meat production is one of the main fronts for the development of family farming. Over the years, there is a greater demand for quality and regularity in its offer. However, a constant problem for producers, and also an environmental problem, is the attack of predators on herds. Ways to inhibit attacks are usually through the gathering of the herd at night, and lately through electric fences or even human surveillance, which is often inefficient and leads to higher production costs, and also can end up with in the death of predators. Thus, in this project, a low-cost computer system for automatic recognition of predators was proposed, aiming at their employability in any type of situation (large companies or family farming). The system makes use of computer vision techniques and network communication for the predators recognition, and generation of alerts for taking actions in order to inhibit attacks. Also, the proposed system can help both producers and environmentalists. Environmentalists will be able to capture and conduct the predator to its natural habitat, and the producer will avoid losses in the livestock production. This project carried out studies to analyze which strategy would be able to accurately recognize predators. The strategy evaluations were divided into two stages: (i) multiclass classification evaluation to recognize 6 different classes of animals (jaguar, puma, dog, sheep, bovine, horse); and (ii) evaluation of binary classification for recognition if there is a predator or not in the imagem. Both strategies makes use of Bag-of-Visual Words (BoVW) representations and machine learning algorithms, and Convolutional Neural Networks. In multiclass classification, the best results obtained using BoVW representations were 0.50. When using CNNs, the best result was obtained with the Xception model with transfer learning and the data augmentation technique, which provided an average accuracy of 0.93 (± 0.01) and a maximum accuracy of 0.96. In the binary classification evaluation, the best results obtained by BoVW representations was 0.8743. When using CNNs, the best result was obtained with the Xception model with transfer learning and the data augmentation technique, which provided an average accuracy of 0.9926 (±

0.04), and maximum accuracy of 1.00.

Keywords: *machine learning*, *computer vision*, *image processing*, *embedded systems*, *convolutional neural networks*, *agriculture*, *predators*.

Sumário

Sumário xv
Lista de Figuras xix
Lista de Tabelas xxi
Lista de Abreviaturas xxiii

1 Introdução **1**
1.1 Contextualização 1
1.2 Objetivos 2
1.3 Organização do Texto 3

2 Conceitos e Trabalhos Relacionados **5**
2.1 Processamento de Imagens 5
2.1.1 Conversão em Escala de Cinza 6
2.1.2 Equalização de Histograma 7
2.1.3 Suavização 7
2.1.4 Técnica de Canny 8
2.1.5 Pontos de Interesse 8
2.2 Representação Estruturada das Imagens 10
2.2.1 Histograma de Cores 10
2.2.2 Bag of Visual Words 13
2.3 Algoritmos de Aprendizado de Máquina 13
2.3.1 *k*-Nearest Neighbors 14
2.3.2 Support Vector Machines 16
2.3.3 Redes Neurais Convolucionais 16
2.3.4 Avaliação da Performance de Classificação 23
2.3.5 Estratégias para Aumentar a Performance de classificação das CNNs 24
2.3.6 Transfer Learning 25
2.4 Aumento de Dados para Aprendizado de Máquina 25
2.5 Medidas de Proximidade 26

2.6 Trabalhos Relacionados 28
2.6.1 Trabalhos que Utilizam CNNs 28
2.6.2 Trabalhos que Utilizam Regiões ou Pontos de Interesse . . 30

3 Proposta: Método para a Detecção Automática de Predadores 33
3.1 Módulo 1: Detecção de Movimento 33
3.2 Módulo 2: Aprendizado de Máquina 35
3.3 Módulo 3: Detecção de Predadores 36

4 Configurações utilizadas nas Avaliações Experimentais 39
4.1 Coleções de Imagens 39
4.2 Configuração Experimental 40
4.2.1 Configurações Utilizadas na Técnica BoVW 40
4.2.2 Configurações Utilizadas nas CNNs 43
4.2.3 Organização dos experimentos 44

5 Resultados e Discussões 51
5.1 Classificação Multiclasse 52
5.1.1 Experimentos com BoVW 52
5.1.2 Experimentos Utilizando Modelos Previamente Treinados . 52
5.1.3 Experimentos com Treinamento Total da Rede 54
5.1.4 Experimentos Com e Sem a Camada Escondida Considerando a Entrada de 128 x 128 pixels 55
5.1.5 Experimentos Com e Sem a Camada Escondida, considerando uma Entrada de 224 x 224 pixels 55
5.1.6 Experimentos com *Transfer Learning* 55
5.1.7 Experimentos com *Transfer Learning*, Com e Sem Camada Escondida, e Considerando uma Entrada 128 x 128 pixels. 55
5.1.8 Experimentos com *Transfer Learning*, Com e Sem a Camada Escondida, e Considerando uma Entrada de 224 x 224 pixels. 57
5.1.9 Experimentos com *Transfer Learning* e Aumento de Dados 58
5.1.10 Comparação Entre os Resultados Obtidos com o Uso de Representações BoVW e CNNs para Classificação Multiclasse 60
5.2 Classificação Binária: Predador x Não Predador 60
5.2.1 Experimentos com BoVW 60
5.2.2 Experimentos com Representações BoVW e Considerando Cão Como Sendo da Classe Predador 61
5.2.3 Experimentos com CNNs 62
5.2.4 Comparação Entre os Resultados Obtidos com BoVW e CNNs 63

5.2.5 Discussão dos Resultados 63

6 Conclusões **67**

Referências Bibliográficas **73**

Lista de Figuras

2.1 (a) Imagem original e (b) imagem após aplicação da conversão em tons de cinza 6
2.2 (a) Imagem tons de cinza e (b) imagem após aplicação da equalização de histograma 7
2.3 (a) Imagem com equalização de histograma e (b) imagem após aplicação do *kernel* gaussiano 8
2.4 (a) Imagem original e (b) imagem após a aplicação da técnica de Canny 9
2.5 Exemplos de aplicação do SURF na imagem original (a), imagem segmentada (b) e imagem com a técnica de Canny 11
2.6 Ilustração da representação do pixel, vetor e imagem 12
2.7 Exemplo de um histograma para a cor vermelha para diferentes animais. 12
2.8 Imagem original (a) e o histograma colorido da imagem (b) 13
2.9 Histograma antes e depois de aplicar a equalização de histograma na imagem em tons de cinza. 14
2.10 Exemplos de *bag-of-visual-words* 15
2.11 Ilustração da extração de características de uma imagem por uma CNN e sua posterior classificação. 17
2.12 Legenda 18
2.13 Ilustração da arquitetura da LeNet-5 18
2.14 Ilustração da arquitetura da AlexNet 19
2.15 Uma arquitetura da rede neural convolucional profunda VGG-16 19
2.16 Ilustração da arquitetura Inception-V1 20
2.17 Uma arquitetura da rede neural convolucional profunda Inception-V3 21
2.18 Ilustração de uma arquitetura ResNet (*skip connections*) 21
2.19 Ilustração de para da arquitetura *InceptionResNet* 22
2.20 Ilustração da arquitetura *Xception* 23

2.21 Imagem original (a), Imagens modificadas de (b) até (f) 27
2.22 Abordagem utilizadas nos trabalhos relacionados 32

3.1 Visão geral da abordagem proposta. 34
3.2 Ilustração dos módulos, componentes dos módulos e interação entre os módulos. 34

4.1 Organização do experimento da BoVW sem a utilização da técnica de Canny. 45
4.2 CNNs com pesos pré treinados e tamanho da entrada 244x244x3. 46
4.3 CNNs com pesos pré treinados e tamanho da entrada 299x299x3. 46
4.4 Legenda de componentes das CNNs. 47
4.5 Experimento das CNNs treinadas sem *transfer learning* 128x128x3. 47
4.6 Experimento das CNNs treinadas sem *transfer learning* 224x224x3. 48
4.7 Experimento das CNNs treinadas com *transfer learning* 128x128x3. 48
4.8 Experimento das CNNs treinadas com *transfer learning* 224x224x3. 49
4.9 Experimento das CNNs treinadas com *transfer learning* 224x224x3, mais aumento de dados. 49

5.1 Comparação das representações BoVW (a) SEM a utilização do filtro de Canny. e (b) COM a utilização do filtro de Canny. 53
5.2 Arquiteturas com pesos pré treinados. 54
5.3 Performances de classificação considerando os modelos SEM camada escondida considerando o tamanho 128 x 128 pixels (a), e COM camada escondida no tamanho 128 x 128 pixels (b) 56
5.4 Performances de classificação considerando os modelos SEM camada escondida considerando o tamanho 224 x 224 (a), e COM camada escondida no tamanho 224 x 224 (b) 57
5.5 Performances de classificação considerando *transfer learning* nos modelos SEM camada escondida, considerando o tamanho 128 x 128 pixels (a), e COM camada escondida no tamanho 128 x 128 pixels (b) 58
5.6 Considerando as convoluções fixa dos modelos nos tamanhos 224 x 224 pixels sem camada escondida (a), e com camada escondida (b) 59
5.7 Performances de classificação considerando somente duas classes "Predador e Não predador"(a) com representações BoVW sem a técnica de Canny no conjunto de imagens; e (b) com representações BoVW aplicando técnica de Canny no conjunto de imagens. 61

5.8 Performances de classificação considerando a espécie Cão da classes Predador no experimento de "Predador e Não predador", (a) considerando a técnica ORB no conjunto imagem sem a técnica de Canny, comparado com a (b) técnica ORB no conjunto imagem depois de aplicado a técnica de Canny 62
5.9 Os melhores resultados em relação a custo-benefício para classificação multiclasse. 64
5.10 Os melhores resultados em relação a custo e benefício para classificação binária. 64

Lista de Tabelas

2.1 Matriz de Confusão para várias classes 23

4.1 Coleção de imagens inicial. 41
4.2 Número de imagens por classe do conjunto de treinamento. . . . 41
4.3 Número de imagens por classe do conjunto de teste. 42

5.1 Matriz de Confusão do modelo Xception. 54

Lista de Abreviaturas

ACC *Accuracy*

AM Aprendizado de Máquina

BoVW *Bag-of-Visual-Words*

CNNs *Convolutional Neural Networks*

DNNs *Deep Neural Networks*

Embrapa Empresa Brasileira de Pesquisa Agropecuária

FAST *Features from Accelerated Segment Test*

GFTT *Good Features to Track*

HTML *HyperText Markup Language*

IA Inteligência Artificial

IP Internet Protocol

IAP Instituto Ambiental do Paraná

IHP Instituto Homem Pantaneiro

RGB *Red*, *Green* and *Blue*

STAR Vem do inglês "destacar em determinadas áreas", estrela

MSER *Maximally Stable Extremal Regions*

BRIEF Binary Robust Independent Elementary

OpenCV *Open Source Computer Vision Library*

ORB *Oriented FAST and Rotated BRIEF*

PIL *Python Imaging Library*

KAZE é uma palavra Japonesa que significa "vento forte".

k-NN k-Nearest Neighbors

SIFT *Scale Invariant Features Transform*

SLIC *Simple Linear Iterative Clustering*

SURF *Speeded Up Robust Features*

SVM *Suppot Vector Machines*

WSGI *Web Server Gateway Interface*

CAPÍTULO

1

Introdução

1.1 Contextualização

A produção de carne ovina no Brasil passou por um processo de transição, começando em cenário de forte informalidade, com abates e comercialização não fiscalizados em direção a um mercado exigente em qualidade e regularidade de oferta, caracterizado por um perfil de consumidores com maior poder aquisitivo e conhecimento gastronômico. Além disso, a cadeia produtiva da carne ovina tem apresentado uma significativa expansão em todas as regiões do Brasil, em função de uma demanda crescente por carne de ovinos nos grandes centros urbanos. A produção teve um aumento regular de 2002 até 2009. Em 2009 foi registrado o recorde de 334,7 mil cabeças abatidas sob inspeção federal. A partir de 2009, o volume de abates fiscalizados iniciou um processo de queda, alcançando no ano de 2016 o menor volume registrado em toda a série histórica 44,7 mil cabeças abatidas, devido a baixa demanda associada a uma queda brusca na oferta de animais a nível nacional (NÓBREGA, 2018).

Um dos problemas que atinge a produção de ovinos e outros rebanhos domésticos no Brasil é a predação[1] (MARCHINI, S. et al., 2011). Além do problema econômico para produtores é também um problema ambiental, pois o ataque de predadores gera conflitos que muitas vezes os levam à morte, mesmo que esse fato se configure em crime ambiental.

Segundo VIDOLIN, G. P. et al (2004) constatou-se em uma fazenda do Pantanal que a onça parda (*Puma concolor*) preda bezerros e ovelhas por serem presas mais fáceis, enquanto que o gado adulto é predado mais pela onça-

[1] O termo predação é utilizado especificamente para ataques de animais selvagens a rebanhos domésticos.

pintada (*Panthera onca*). Um exemplo disso é relatado pelo Instituto Ambiental do Paraná (IAP)[2], o qual tem recebido nos últimos anos várias denúncias de ataques a rebanhos domésticos por onças em áreas rurais, causando prejuízos a grande e pequenos produtores. Após as denúncias, o IAP vistoria as propriedades e propõem soluções para o problema da predação. Vale ressaltar que durante as vistorias, são utilizados procedimentos como a verificação da existência da caça de animais silvestres, levantamento do histórico de predação na propriedade, registro do ataque, verificação da espécie de presa consumida, quantidade de cabeças perdidas (VIDOLIN, G. P. et al, 2004). Também são feitas orientações aos produtores sobre procedimentos preventivos, como a melhoria no manejo das criações, confinamento do rebanho a noite em recinto iluminado, cerca elétrica, disponibilização de cães para afastar os predadores, e em último caso, é utilizada a medida de capturar e o translado do predador para um outro ambiente adequado, preservado assim o mesmo. Porém, estes atendimentos são feitos após as denúncias repassadas tardiamente (VIDOLIN, G. P. et al, 2004).

Outra forma de inibir os ataques de predadores é por meio da vigilância humana. Porém, a vigilância humana pode ser inviável tanto para o pequeno produtor quanto para grande, devido ao custo, que é proporcional ao tamanho da propriedade, ou ainda o custo de instalação de cercas elétricas e iluminação em grande parte da propriedade, o que também incorre em altos custos.

1.2 Objetivos

O objetivo geral foi desenvolver um sistema de monitoramento de ataques via detecção de predadores em imagens, utilizando técnicas de visão computacional. Ao reconhecer um predador, serão gerados alertas para tomada de decisões de ações a inibir os ataques.

Esta solução pode ser aplicada tanto nos grandes quanto nos pequenos produtores devido ao potencial baixo custo da solução proposta. Este sistema também pode auxiliar instituições dedicadas à preservação de animais selvagens a levantar informações referentes aos tipos de predadores e quantidade destes em uma determinada região.

Já como objetivos específicos têm-se:

- Propor uma infraestrutura inicial de (rede e *hardware*) que possibilitem ações rápidas, para evitar a perda de animais domésticos; e
- Estudos de técnicas de processamento de imagens e algoritmos de aprendizado de máquina mais adequados à detecção de predadores.

[2]Instituto Ambiental do Paraná: http://www.iap.pr.gov.br/

1.3 Organização do Texto

O restante da dissertação está dividido em 5 capítulos. No Capítulo 2 são apresentados os conceitos necessários para o entendimento deste projeto e os trabalhos relacionados ao tema. No Capítulo 3 são apresentados os detalhes dos passos do método proposto para a detecção automática de predadores. No Capítulo 4 são apresentadas as configurações utilizadas nas avaliações experimentais sobre reconhecimento de predadores. No Capítulo 5 são apresentados resultados e discussões referentes a detecção de predadores utilizando diferentes técnicas de processamento de imagens e algoritmos de aprendizado de máquina. Por fim, no Capítulo 6 são apresentadas as conclusões sobre este projeto de mestrado.

CAPÍTULO

2

Conceitos e Trabalhos Relacionados

Neste capítulo são apresentados os conceitos necessários para o entendimento do desenvolvimento desse projeto de mestrado, como a aplicação da visão computacional (processamento de imagens e aprendizado de máquina) e a avaliação da performance de classificação, referendados pelos trabalhos relacionados.

Este trabalho está voltado para a área de Visão Computacional, a qual, dentre suas diversas finalidades, visa o reconhecimento de objetos em imagens e sua classificação (MARENGONI and Stringhini, 2009). Para isso, faz-se uso de duas áreas principais: processamento de imagens e aprendizado de máquina. Ambas serão descritas a seguir.

2.1 Processamento de Imagens

Para fazer o reconhecimento de objetos nas imagens de maneira eficiente pelos algoritmos de aprendizado de máquina, as imagens precisam passar por um processo de "limpeza" para eliminar ruídos e informações não relevantes, i.e, ocultam-se ou ressaltam-se padrões, ou ainda padronizam-se as imagens (FREITAS, R. P. da S. , 2016; SZELISKI, 2010). Essa limpeza é realizada por filtros de forma a melhor separar as partes de interesse. O resultado do processamento de imagem é outra imagem, porém, com características relevantes ressaltadas para que se possam extrair características mais discriminativas da imagem e consequentemente aumentar a performance de classificação de algoritmo de aprendizado de máquina. Existem inúmeras técnicas de processamento de imagens que podem ser aplicadas (DAVIES, 2017; SZELISKI, 2010).

Os conceitos que estão nas subseções a seguir são utilizados com o objetivo de extrair características mais discriminativas da imagem.

2.1.1 Conversão em Escala de Cinza

Vale ressaltar que uma vez lida a imagem em formato vermelho, verde e azul, do inglês *Red, Green and Blue* (RGB), esta pode ser convertida em escala de cinza, criando assim um vetor com um único canal para a imagem de saída. A conversão clássica de imagens RGB para escala de cinza é feita utilizando a média ponderada considerando a luminosidade de cada cor, onde cada cor é mais sensível que outra para os olhos humanos (MENEZES, 2010), a conversão é feita através da Equação 2.1 em cada pixel.

$$Grayscale = (3,0 \cdot R) + (59,0 \cdot G) + (11,0 \cdot B) \tag{2.1}$$

Entretanto, vale ressaltar que uma média simples já é o suficiente:

$$Grayscale = (R + G + B)/3 \tag{2.2}$$

Em uma imagem em tons de cinza, os pixels podem assumir valores entre zero, cor negra, até 255 na cor branca. O exemplo da Figura 2.1 (a) apresenta a imagem colorida e a imagem da Figura 2.1 (b) apresenta a imagem convertida para escala de cinza.

Figura 2.1: (a) Imagem original e (b) imagem após aplicação da conversão em tons de cinza

(a) (b)

Fonte: Autoria própria.

2.1.2 Equalização de Histograma

A equalização de histograma é um método que ajusta a intensidade de realce do contraste da imagem, exemplo na Figura 2.2. O contraste de uma imagem é uma medida do espalhamento dos níveis de intensidade que nela ocorrem (CROSTA, 1999).

Figura 2.2: (a) Imagem tons de cinza e (b) imagem após aplicação da equalização de histograma

(a) (b)

Fonte: Autoria própria.

A aplicação da equalização de histograma é feita com o mapeamento da distribuição dos pixels inicial da imagem e após este mapeamento é modificado para outra distribuição de forma mais uniforme dos valores de intensidade dos pixels.

2.1.3 Suavização

A suavização é uma operação de processamento de imagem utilizada principalmente para reduzir ruído, utilizando-se um filtro (CROSTA, 1999). O filtro Gaussiano pode ser aplicado, convocando cada ponto da imagem de entrada com *kernel* gaussiano e após isto são somados todos eles para produzir a imagem de saída. Na Figura 2.3 é apresentada uma ilustração do resultado da aplicação do *kernel* gaussiano em uma imagem.

No processo de suavização, as imagens são modificadas de maneira que reduza a variação da distribuição das cores. Este processo é importante para o aprendizado de máquina, pois melhora a captura dos padrões importantes, deixando de aprender os ruídos que foram retirados (CATRO and FERRARI, 2016).

Figura 2.3: (a) Imagem com equalização de histograma e (b) imagem após aplicação do *kernel* gaussiano

(a) (b)

Fonte: Autoria própria.

2.1.4 Técnica de Canny

O algoritmo de Canny é utilizado para detectar bordas em três etapas (SILVA, J. F. C. et al., 2004):

1. Filtragem: reduz os ruídos presentes na imagem;
2. Realce: calcula magnitude do gradiente;
3. Limiarização: determina quais bordas serão consideradas, pois não é possível saber o valor máximo da magnitude do gradiente que realmente corresponde a uma borda.

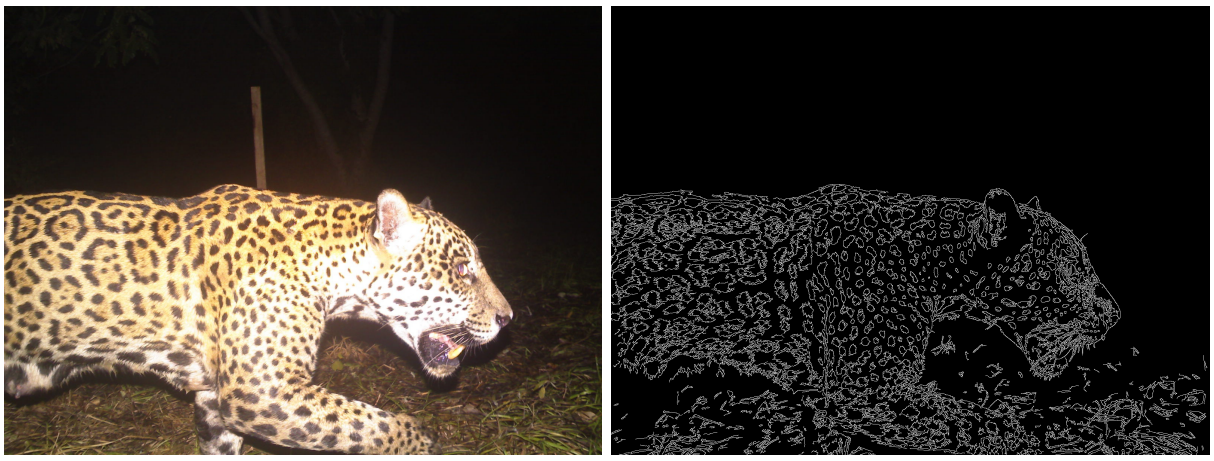
Na Figura 2.4 é apresentada a imagem antes e depois de aplicar a técnica de Canny.

2.1.5 Pontos de Interesse

Os pontos de interesse são pontos com localizações bem definidas e que estão associados às variações de propriedades como, intensidade, cor e textura na imagem. Tais pontos possuem um grande potencial para serem características discriminativas da imagem (NOVAIS, 2016). Além disso, SILVA, F. A. et al. (2013), o propósito dos descritores é que sejam invariantes, a localização, escala, orientação, ou distorções, como borramentos e alterações de iluminação.

A extração dos pontos de interesse possui dois aspectos distintos: detecção dos pontos e descrição dos pontos. A detecção é responsável por identificar

Figura 2.4: (a) Imagem original e (b) imagem após a aplicação da técnica de Canny



(a) (b)

Fonte: Autoria própria.

os locais de interesse de acordo com as variações de propriedades apresentadas acima. Em seguida, é necessário computar uma descrição do ponto de interesse (um vetor de características do ponto).

Os algoritmos de extração de pontos de interesse são divididos em três grupos: os de detecção dos pontos, os de descrição dos pontos, e os que fazem ambos, extraem os pontos e os descritores dos mesmos. Exemplos de técnicas em cada um desses grupos são:

1. Detectores: FAST, STAR, MSER, GFTT;
2. Descritores: BRIEF;
3. Detectores e Descritores: ORB, KAZE, SIFT, SURF;

O algoritmo *Oriented FAST and Rotated BRIEF* (ORB), é uma fusão do detector *Features from Accelerated Segment Test* (FAST) com o descritor *Binary Robust Independent Elementary Features* (BRIEF). O algoritmo FAST encontra pontos notáveis, que são candidatos a pontos de interesse (SANTANA, B. A. S. et al., 2015) e o algoritmo BRIEF computa os descritores de pontos de interesse. Após isso, é aplicado um filtro de cantos Harris para separar os melhores pontos de interesse.

Outros dois algoritmos detectores e descritores frequentemente utilizados na literatura são o *Scale Invariant Features transform* (SIFT) e o *Speeded Up Robust Features* (SURF). No algoritmo (SIFT) os descritores possuem 128 dimensões para descrever uma característica. O algoritmo (SURF) é um algoritmo baseado no SIFT, porém, os descritores do SURF possuem 64 dimensões para descrever uma característica local na imagem.

Na Figura 2.5 são apresentados exemplos de aplicação do algoritmo SURF em uma imagem original, na versão segmentada da imagem original e na imagem original, com aplicação da técnica de Canny. Pode-se observar que diferentes pontos de interesse podem ser gerados de acordo com as diferentes características localizadas nas imagens.

2.2 Representação Estruturada das Imagens

Para que se possam aplicar os algoritmos de aprendizado de máquina ou para fazer algum tipo de processamento nas imagens, é necessário representar as imagens em um formato estruturado. A leitura de uma imagem pode gerar uma representação matricial ou tensorial desta imagem. No caso de imagens branco e preto, ou escalas de cinza, é gerada uma matriz em que cada célula da matriz (pixel), possui o valor 0 ou 1 ou um valor de 0 a 255 para imagens em escala de cinza. Já na leitura considerando um padrão RGB para imagens coloridas, cada pixel é representado por um vetor, o qual contém valores de 0 a 255 para os canais R, G ou B, gerando assim uma representação em formato de tensor. Na Figura 2.6 é apresentada a ilustração da representação do pixel, vetor e imagem. Neste trabalho, cada imagem i será denotada por um vetor $\mathbf{x}_i$. Cada posição do vetor conterá um valor numérico para um determinado tipo de atributo. Diferentes atributos podem ser utilizados para representar uma imagem. Numa das representações mais comuns, cada dimensão k do vetor representa um pixel na posição k da imagem, sendo que cada *pixel* é composto pelos valores de R (*red*), G (*green*) e B (*blue*) que compõem a cor do respectivo *pixel*.

A seguir são descritas as representações de imagens consideradas no desenvolvimento deste trabalho.

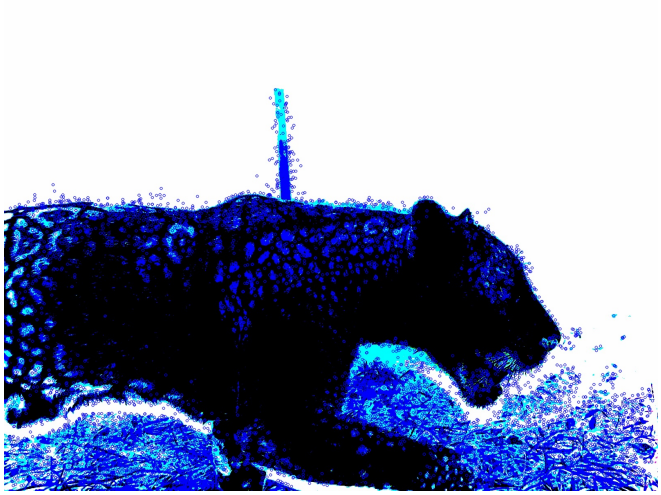
2.2.1 Histograma de Cores

Uma imagem também pode ser representada por histogramas de cores ao invés dos *pixels* individuais. O histograma quantifica a incidência de cada cor, de faixas de cores, ou ainda de faixas de luminosidade na imagem (SZELISKI, 2010).

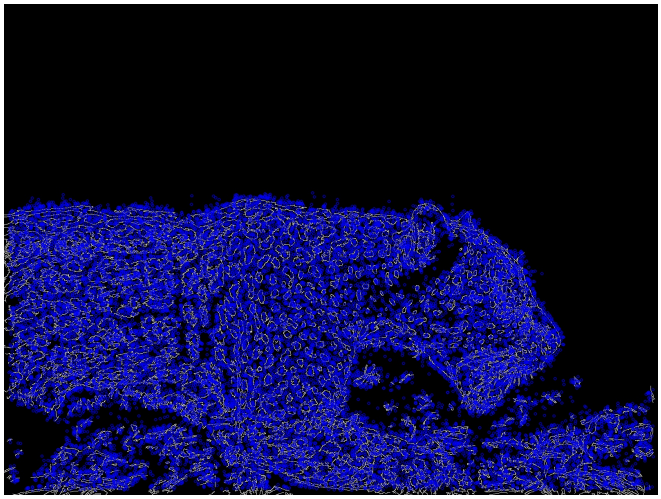
O histograma é uma representação que pode ser visualizada no formato gráfico, é uma representação gráfica de distribuição de intensidade das cores na imagem e que pode ser usado como vetor de características para os algoritmos de aprendizado de máquina, já que imagens de diferentes categorias irão apresentar diferentes histogramas. O histograma de cores no espaço de cores RGB, são amplamente utilizados para imagens digitais. Na Figura 2.7 é apresentado um exemplo de histograma para tons de vermelho de duas imagens:

Figura 2.5: Exemplos de aplicação do SURF na imagem original (a), imagem segmentada (b) e imagem com a técnica de Canny

(a)



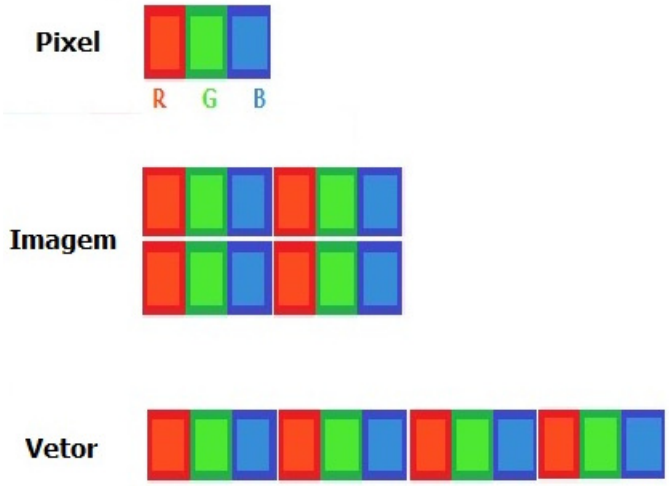
(b)



(c)

Fonte: autoria própria.

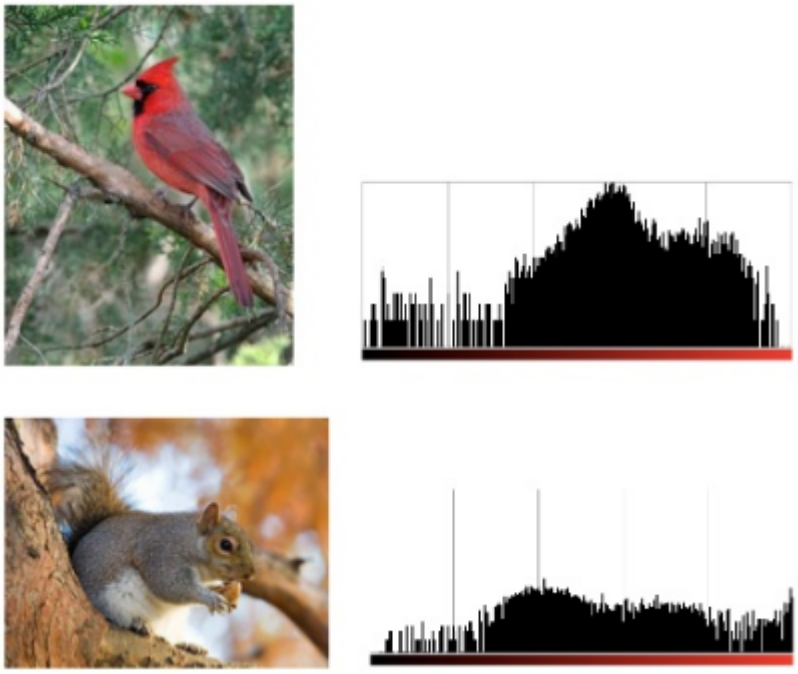
Figura 2.6: Ilustração da representação do pixel, vetor e imagem



Fonte: Autoria própria.

uma contendo um pássaro da espécie Cardial do Norte (*Northern Cardinal*) e outra contendo um esquilo. Já na Figura 2.8 tem o exemplo de como pode ser representado o histograma de cores RGB de uma imagem .

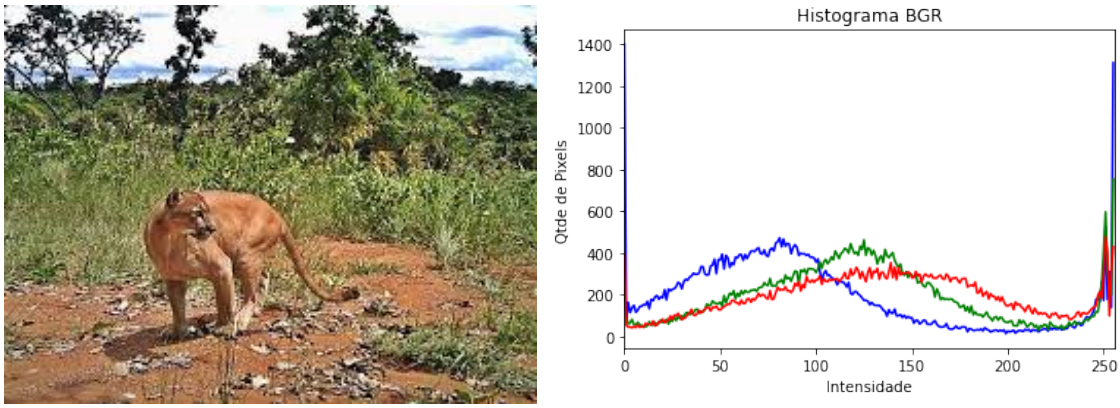
Figura 2.7: Exemplo de um histograma para a cor vermelha para diferentes animais.



Fonte: GRANDIS (2012).

A representação por histograma é uma forma de analisar o antes e depois de aplicar o processamento de imagem visto na Seção 2.1. Na figura 2.9, estão

Figura 2.8: Imagem original (a) e o histograma colorido da imagem (b)



(a) (b)

Fonte: Autoria própria.

exemplos de representação do antes e depois do processamento de imagem com a equalização de histograma (apresentado na Seção 2.1.2).

2.2.2 Bag of Visual Words

Os descritores das imagens podem ser combinados de forma a gerar uma representação estruturada da coleção de imagens. Uma forma comum de se fazer isso é por meio da *bag-of-visual-words* (YANG and NEWSAM, 2010; XU et al., 2009). A *bag-of-visual-words* é semelhante à *bag-of-words* utilizada na Mineração de Textos, a diferença se dá pelo fato que as palavras visuais da *bag-of-visual-words* são representadas pelos descritores dos pontos de interesse semelhantes. Com isso, para gerar a *bag-of-visual-words*, roda-se um algoritmo de agrupamento de dados, como o k-Means (TAN et al., 2005) ou PLSA (HOFMANN, 1999), considerando todos os descritores dos pontos de interesse extraídos de todas as imagens de um conjunto de dados. Cada grupo representará uma palavra visual, e portanto, a dimensionalidade da representação se dará pelo número de grupos. Os valores das dimensões representam o número de descritores de uma imagem que foram atribuídos aos respectivos grupos. Na Figura 2.10 é apresentado o método da *bag-of-visual-words*.

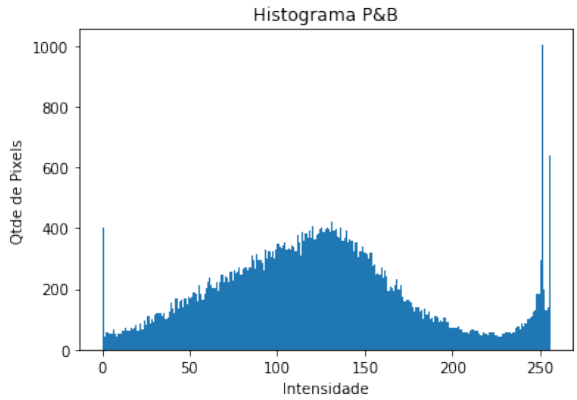
2.3 Algoritmos de Aprendizado de Máquina

Para realizar a classificação automática, podem ser utilizados algoritmos de aprendizado de máquina (AM). O objetivo dos algoritmos de aprendizado de máquina é aprender, generalizar ou ainda extrair padrões ou características das classes das coleções, com base nos valores das características dos

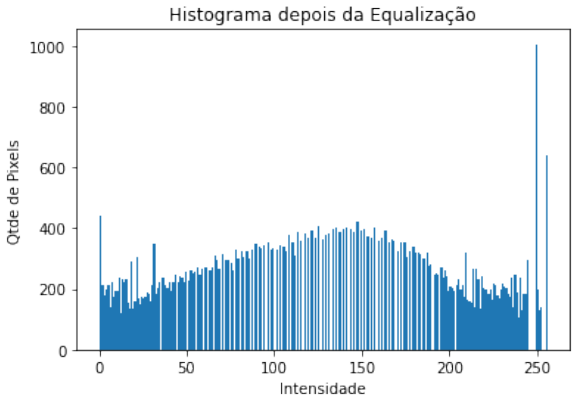
Figura 2.9: Histograma antes e depois de aplicar a equalização de histograma na imagem em tons de cinza.

(a) Imagem tons de cinza

(b) Imagem equalização de histograma



(c) histograma da imagem tons de cinza



(d) histograma da imagem depois da equalização

Fonte: Autoria própria.

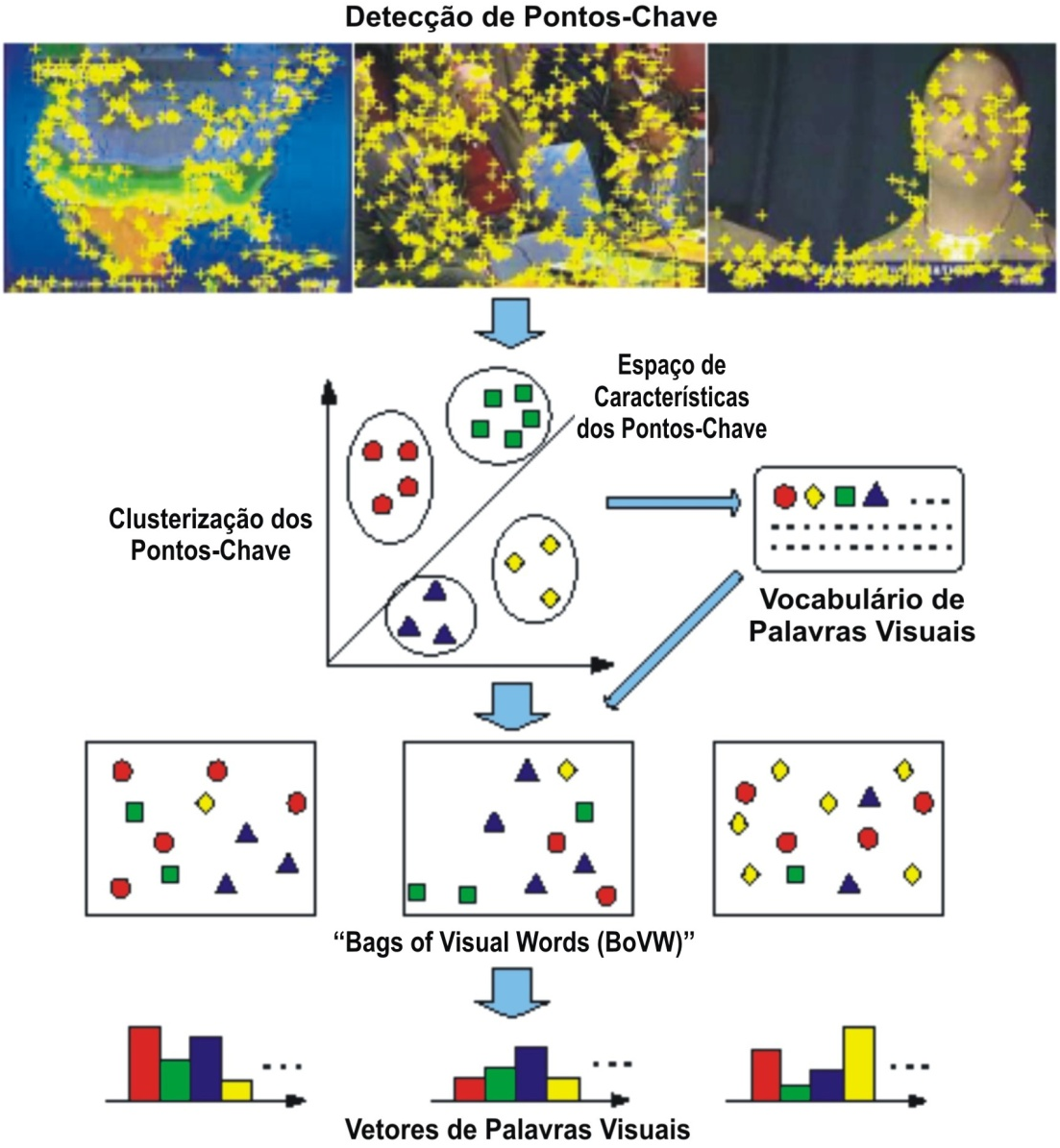
exemplos e, dependendo do tipo do aprendizado, nos respectivos rótulos dos exemplos (TAN et al., 2005). Se o aprendizado for do tipo indutivo, é induzido um modelo de classificação, o qual será capaz de classificar automaticamente novos exemplos.

Neste trabalho serão considerados algoritmos de aprendizado indutivo supervisionado, i.e, aprendem com base apenas em exemplos que possuem rótulos. Foram estudados os algoritmos mais utilizados na classificação de imagens: *k-Nearest Neighbors* (baseado em instâncias), *Support Vector Machine* (estatístico).

2.3.1 k-Nearest Neighbors

O algoritmo *k-Nearest Neighbor* (*k-NN*) é um algoritmo baseado em instâncias, na qual a classificação é dada com base nas classes dos k vizinhos mais próximos. Para isso, é necessário o cálculo de proximidades para a obtenção

Figura 2.10: Exemplos de *bag-of-visual-words*



Fonte: (YANG, J. et al., 2007)

da vizinhança utilizando, por exemplo, as medidas de distância e similaridades apresentadas na Seção 2.5.

A função de classificação do *k-NN* pode ser ponderada ou não ponderada, pela distância ou similaridade dos vizinhos mais próximos. No caso da classificação não ponderada, o voto para cada classe c_j da coleção para a classificação de um exemplo $\mathbf{x}_i$ é dada por:

$$f(c_j, \mathbf{x}_i) = \sum_{d_k \in \mathcal{N}_k^{\mathbf{x}_i}} y_{\mathbf{x}_i, c_j}, \tag{2.3}$$

na qual $y_{\mathbf{x}_i,c_j}$ é igual a 1 se $\mathbf{x}_i$ pertence à classe c_j e 0 caso contrário, e $\mathcal{N}_k^{\mathbf{x}_i}$

representa o conjunto dos k vizinhos mais próximos de $\mathbf{x}_i$.

Já ao utilizar o voto ponderado pela distância dos vizinhos, pode-se utilizar a função apresentada na Equação 2.4 em caso do uso de medidas de similaridade e a Equação 2.5 em caso de medidas de distância (TAN et al., 2005).

$$f(c_j, \mathbf{x}_i) = \sum_{d_k \in \mathcal{N}_k^{\mathbf{x}_i}} \frac{1}{(1 - sim(\mathbf{x}_i, \mathbf{x}_k))} \cdot y_{\mathbf{x}_i, c_j} \tag{2.4}$$

$$f(c_j, \mathbf{x}_i) = \sum_{d_k \in \mathcal{N}_k^{\mathbf{x}_i}} \frac{1}{(dist(\mathbf{x}_{d_i}, \mathbf{x}_{d_j}))} \cdot y_{\mathbf{x}_i, c_j} \tag{2.5}$$

2.3.2 Support Vector Machines

O algoritmo *Support Vector Machines* (SVM) visa obter um hiperplano de separação de margem máxima entre exemplos pertencentes a uma classe positiva (+1) e uma classe negativa (-1) (VAPNIK, 1998).

Portanto, O SVM induz hiperplanos paralelos H_i^+ e H_i^- com máxima distância entre eles visando respeitar as seguintes restrições:

$$H_i^+ : \sum_{k=1}^{m} w_k \cdot x_{i,k} + b \geq 1 - \xi_{\mathbf{x}_i} \text{ se } y_{x_i} = +1 \tag{2.6}$$

e

$$H_i^- : \sum_{k=1}^{m} w_k \cdot x_{i,k} + b \leq -1 + \xi_{\mathbf{x}_i} \text{ se } y_{d_i} = -1, \tag{2.7}$$

na qual $\mathbf{w}$ e b são parâmetros do hiperplano, e tal que $\xi_{\mathbf{x}_i} > 0$ para todo exemplo do conjunto de treinamento. Com isso, pode-se otimizar os hiperplanos mesmo que um exemplo positivo esteja do lado do hiperplano dos exemplos negativos. Porém, um dos termos de otimização do SVM considera $C(\sum_{i=1}^{n} \xi_{\mathbf{x}_i})$, i.e, dependendo do valor do parâmetro C, os erros podem não ser aceitáveis.

2.3.3 Redes Neurais Convolucionais

Uma rede neural realiza o aprendizado por meio de uma estrutura computacional que visa simular o cérebro humano, i.e., uma estrutura de neurônios e conexões entre eles. Basicamente, o aprendizado se dá ajustando os pesos dos neurônios para produzirem uma saída desejada na rede (HAYKIN, 1999).

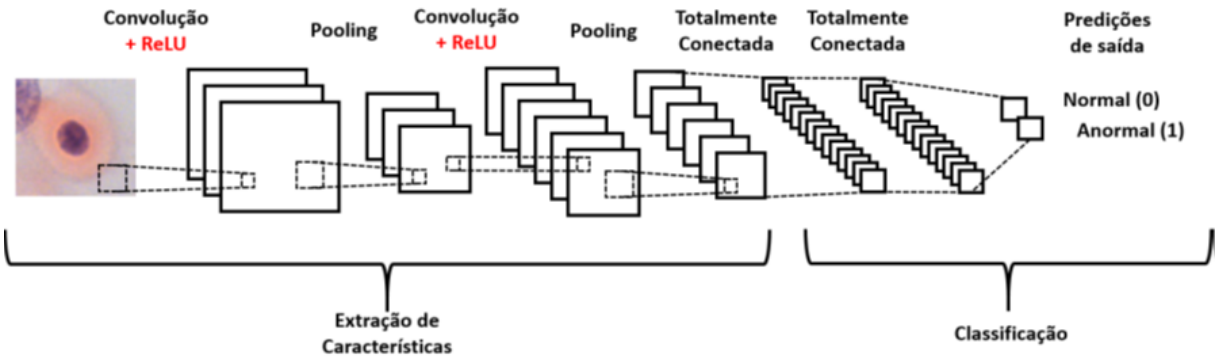
A rede neural pode possuir diferentes arquiteturas, as quais consistem em definir blocos de neurônios organizados em diferentes camadas de forma a otimizar o desempenho da rede neural em uma determinada tarefa. Recentemente, o conceito de aprendizado profundo, que combina camadas ocultas em

uma rede neural com outras funções entre as camadas, têm ganhado destaque, principalmente em tarefas de visão computacional (AGGARWAL, 2018).

Uma das arquiteturas de destaque na área de visão computacional são as Redes Neurais Convolucionais, do inglês *Convolutional Neural Networks* (*CNN*). As CNNs são redes neurais artificiais que utilizam operações de convolução em pelo menos uma camada da rede (AGGARWAL, 2018). Uma operação de convolução corresponde uma operação de produto (·) entre uma matriz de convolução, também denominada *kernel*, e a entrada da rede, de forma que a saída resultará em valores positivos (ou maiores que um limiar de ativação) se os pesos de entrada forem similares aos pesos da matriz de convolução. Portanto, se a matriz de convolução representar um determinado padrão, a entrada da rede terá que conter um padrão similar para que se gerem estímulos para as próximas camadas da rede.

Também são consideradas nas CNNs operações de *pooling*, a qual é capaz de capturar padrões mesmo que haja variação espacial e também para reduzir o tamanho das entradas para processamentos subsequentes na rede, e camadas ou operações denominadas ReLU, que correspondem às funções de ativação de uma rede neural tradicional. Após as camadas de convolução, ReLU e *pooling*, o restante da arquitetura de uma CNN é semelhante ao de uma rede neural multicamadas tradicional, i.e., camadas de neurônios conectados entre si. Na Figura 2.11 é apresentada uma ilustração das camadas de uma CNN.

Figura 2.11: Ilustração da extração de características de uma imagem por uma CNN e sua posterior classificação.



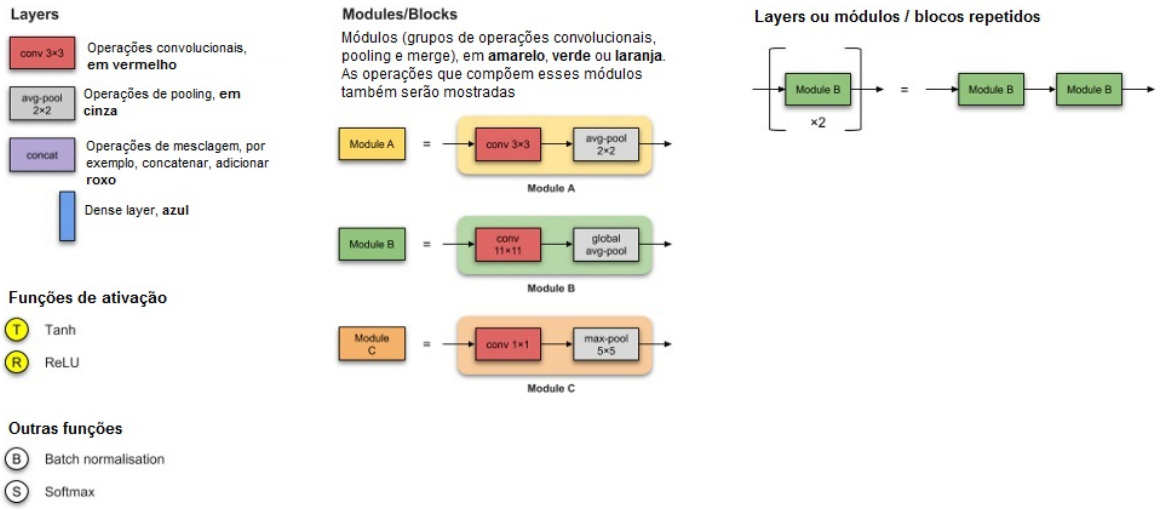
Fonte: ARAÚJO, Flávio H. D. et al (2017).

As CNNs podem ter sua eficiência melhores ou piores de acordo com a sua atividade-fim, na busca de aumentar a performance de classificação (AGGARWAL, 2018). Algumas das arquiteturas mais comuns estão listadas a seguir:

1. **LeNet-5** (LECUN, Y. et al., 1998): utilizada para identificar dígitos ma-

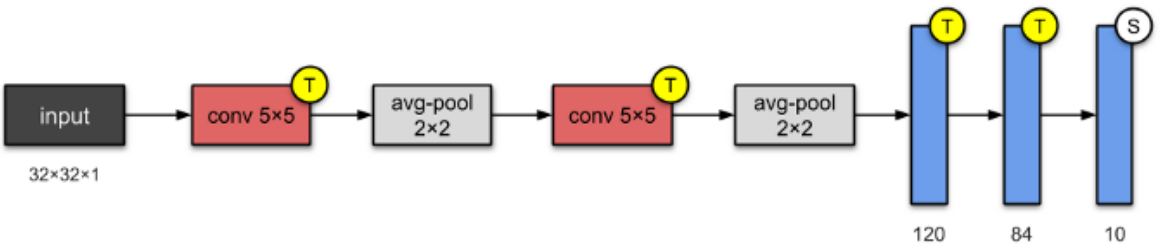
nuscritos e para reconhecimento de código postal. Essa arquitetura serviu como base para outras CNNs. Na Figura 2.13 é apresentada uma ilustração das camadas da LeNet-5.

Figura 2.12: Legenda



Fonte: adaptado de KARIM (2019).

Figura 2.13: Ilustração da arquitetura da LeNet-5

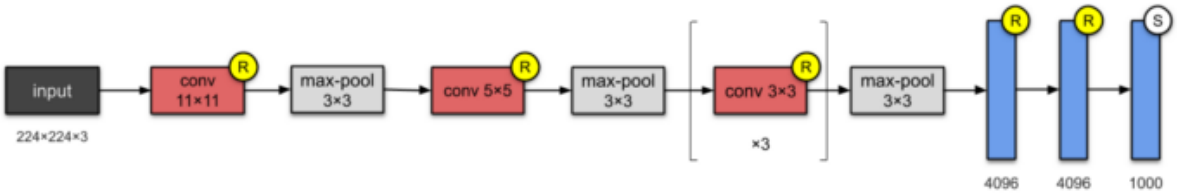


Fonte: KARIM (2019).

2. **AlexNet** (KRIZHEVSKY et al., 2012): desenvolvida pelo grupo KRIZHEVSKY et al. (2012) para competição ImageNet de 2012, ficando em primeiro lugar na referida competição, isto estimulou uma nova linha de pesquisa focada em encontrar redes neurais convolucionais de melhor desempenho. Na Figura 2.14 é apresentada uma ilustração das camadas da AlexNet.

3. **VGG-16** (SIMONYAN and ZISSERMAN, 2014): criada com o objetivo de investigar o efeito da utilização da profundidade na CNNs e sua precisão

Figura 2.14: Ilustração da arquitetura da AlexNet

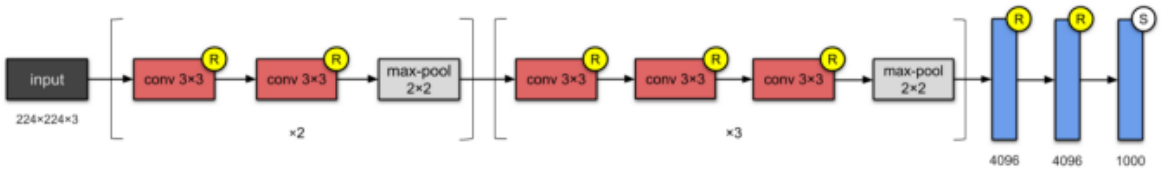


Fonte: KARIM (2019).

no reconhecimento de imagem em larga escala. Uma característica marcante nesse tipo de rede são os filtros de tamanhos reduzidos e a grande profundidade.

Na Figura 2.15. é apresentada uma ilustração das camadas da arquitetura VGG-16.

Figura 2.15: Uma arquitetura da rede neural convolucional profunda VGG-16



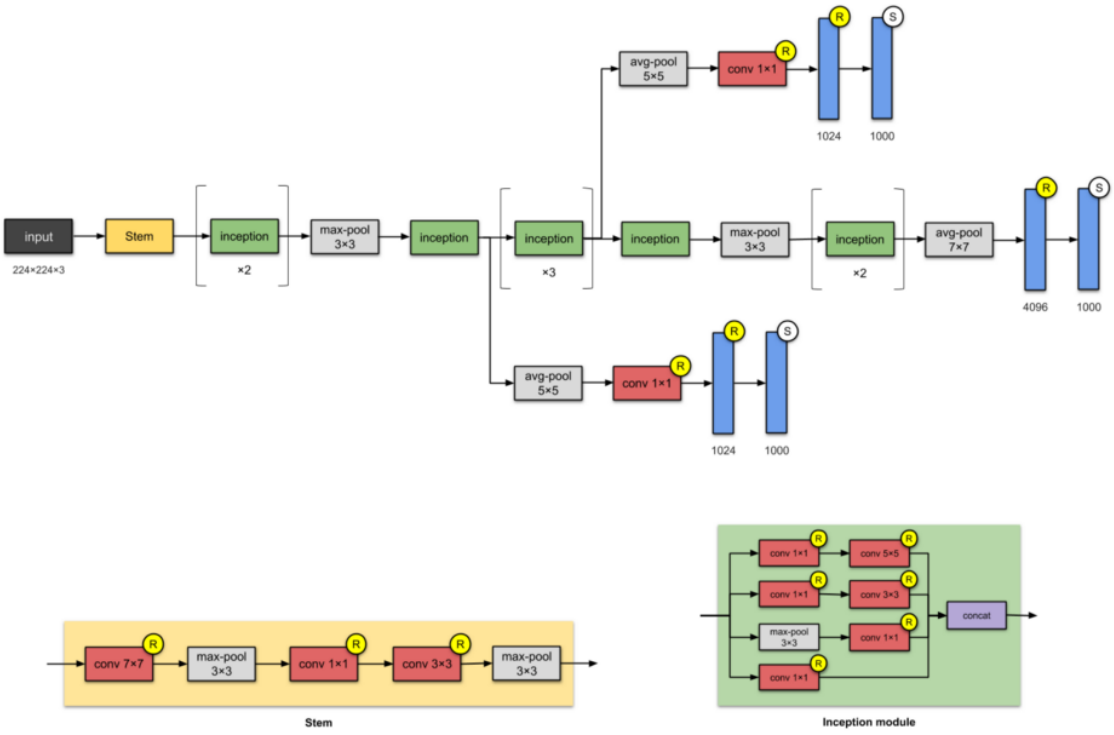
Fonte: KARIM (2019).

4. **Inception:** A arquitetura convolucional profunda *Inception-v1* foi introduzida como *GoogLeNet*, depois foi refinada, adicionando a introdução em lote, tornando-se *Inception-v2* e posteriormente surgiu a ideia de adicionar fatoração na terceira iteração, gerando assim a *Inception-v3* (SZEGEDY et al., 2017).

    - **Inception-v1 conhecida como GoogLeNet** (SZEGEDY, C. et al., 2015): é um modelo que ficou em primeiro lugar na competição *ImageNet Large-Scale Visual Recognition Challenge* 2014. A principal característica desta arquitetura, são as camadas de *inception*. A ideia básica do módulo de *inception* é considerar diferentes tamanhos de filtros para capturar diferentes níveis de detalhe e depois combinar esses filtros. Filtros de tamanho grande capturam detalhes em grandes áreas e filtros de tamanho pequeno capturam padrões em pequenas áreas. Além disso, durante o aprendizado, a rede pode aprender

qual filtro é mais importante para uma determinada tarefa. Especificamente na GoogLeNet, os tamanhos dos filtros são 1×1, 3×3, e 5×5. Na Figura 2.16 é apresentada a arquitetura Inception-V1 e uma ilustração do módulo de *inception* da GoogLeNet.

Figura 2.16: Ilustração da arquitetura Inception-V1

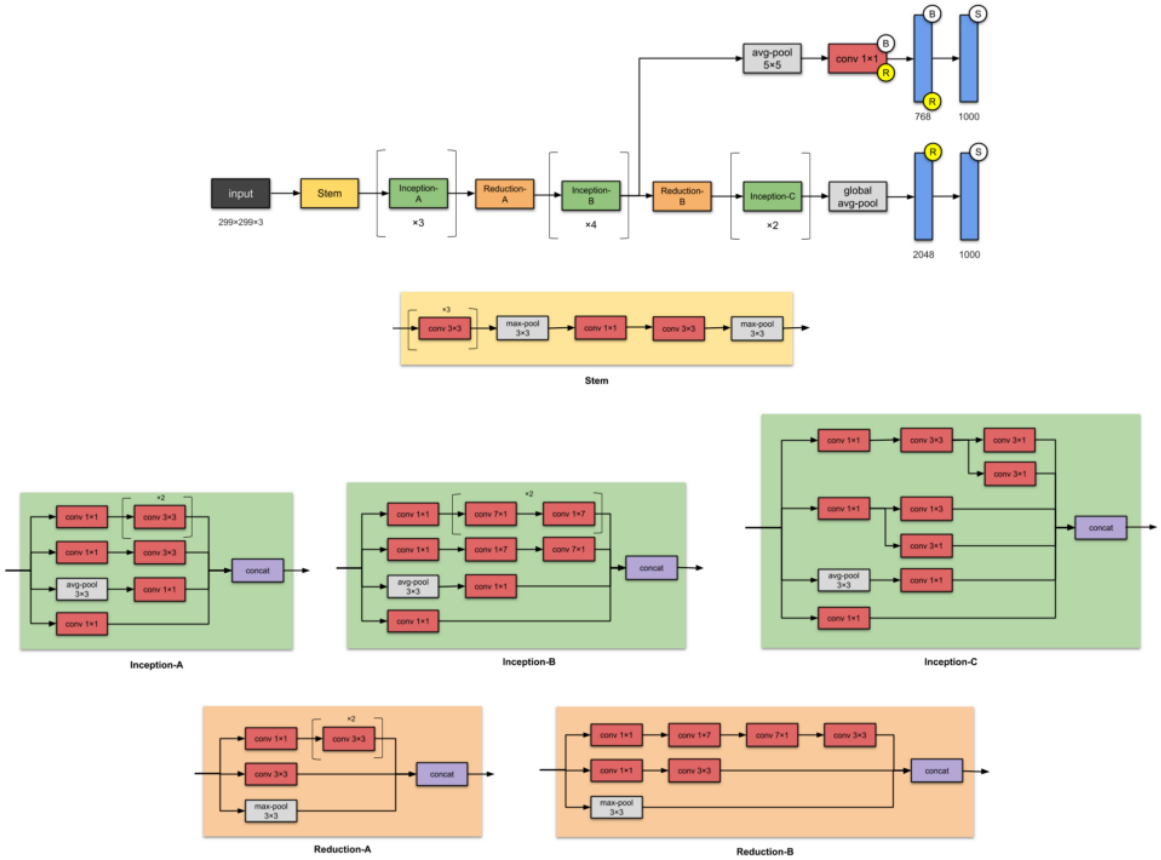


Fonte: KARIM (2019).

- **Incepition-V3** é uma versão melhorada das redes anteriores, que são Incepition-V2 e Incepition-V1. A arquitetura Inception-V3 apresentada na Figura 2.17, demonstrou ter um desempenho bom e custo computacional relativamente baixo. A Incepition-V3 conseguiu melhorar os resultados utilizando convoluções, com fatoração e regularização(SZEGEDY, 2016).

5. **ResNet** (HE, K. et al., 2016): vencedora no *ImageNet Large-Scale Visual Recognition Challenge* 2015, obtendo um erro de apenas 3.6%. Composta por 152 camadas, propôs inovações para viabilizar o seu treinamento. A principal inovação consiste na premissa de que algumas imagens, como um quadrado, requer uma complexidade de abstração menor, e portanto, menos camadas, enquanto que o reconhecimento de uma onça requer uma complexidade maior e, portanto, necessitando mais camadas. Dado isso, a ResNet utiliza conceitos denominados *skortcut connections* com

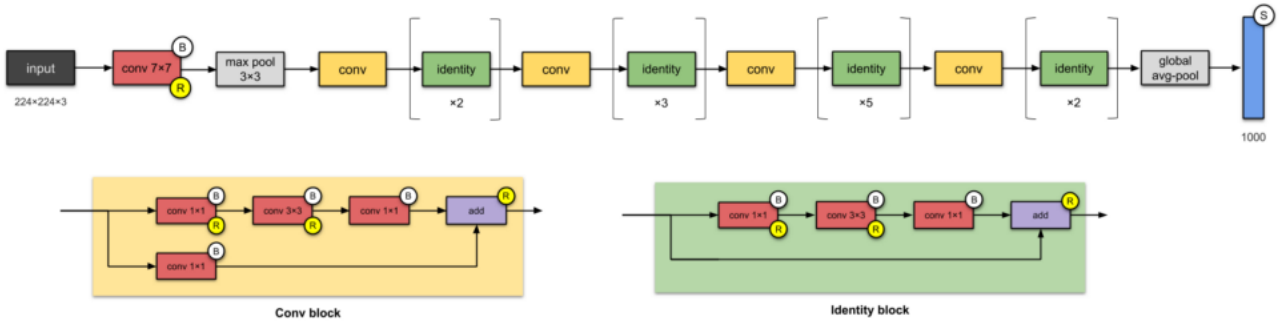
Figura 2.17: Uma arquitetura da rede neural convolucional profunda Inception-V3



Fonte: KARIM (2019).

funções residuais, conhecido também por conexões residuais entre as camadas, os quais são caracterizados pela cópia do conteúdo entre camadas subsequentes para objetos cuja complexidade de aprendizado é mais simples.

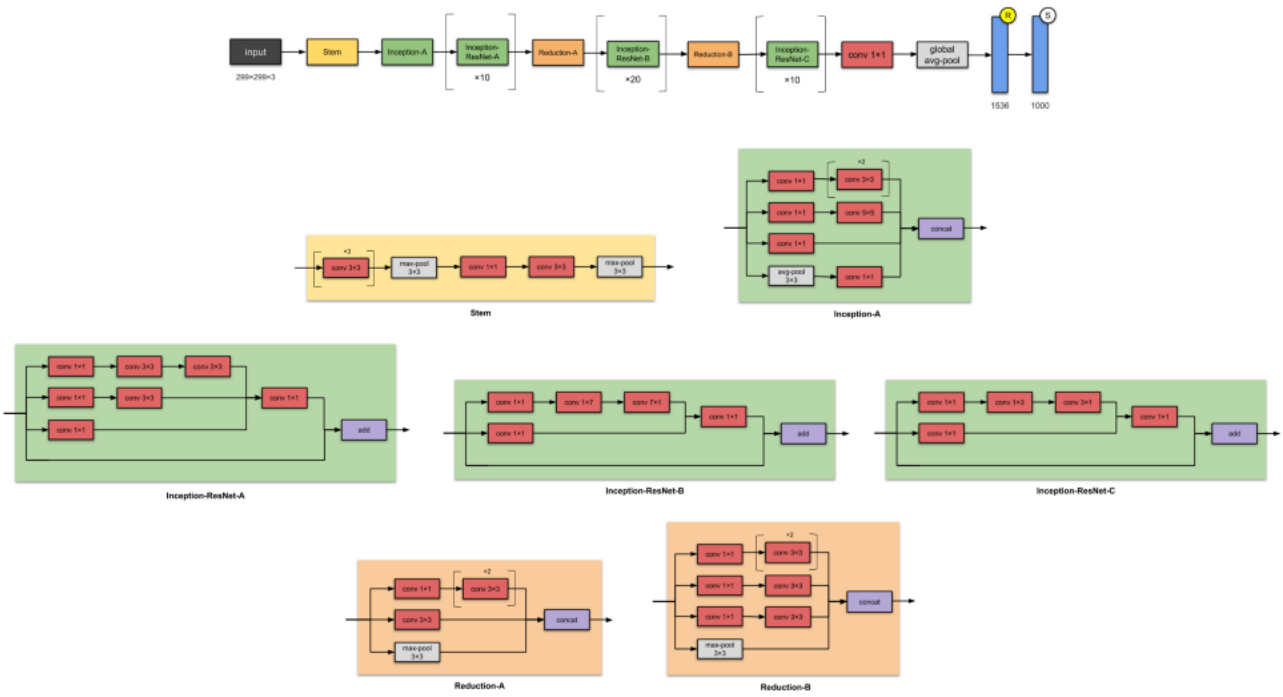
Figura 2.18: Ilustração de uma arquitetura ResNet (*skip connections*)



Fonte: (KARIM, 2019).

6. **InceptionResNet** é a evolução da Inception-V3 e V4, é uma arquitetura capaz de utilizar conexões residuais, da mesma forma que o ResNet, com bastante foco em *(batch normalization)* (SZEGEDY et al., 2017). O *batch normalization* está descrito na Seção 2.3.5. A arquitetura InceptionResNet pode ser vista na Figura 2.19.

Figura 2.19: Ilustração de para da arquitetura *InceptionResNet*

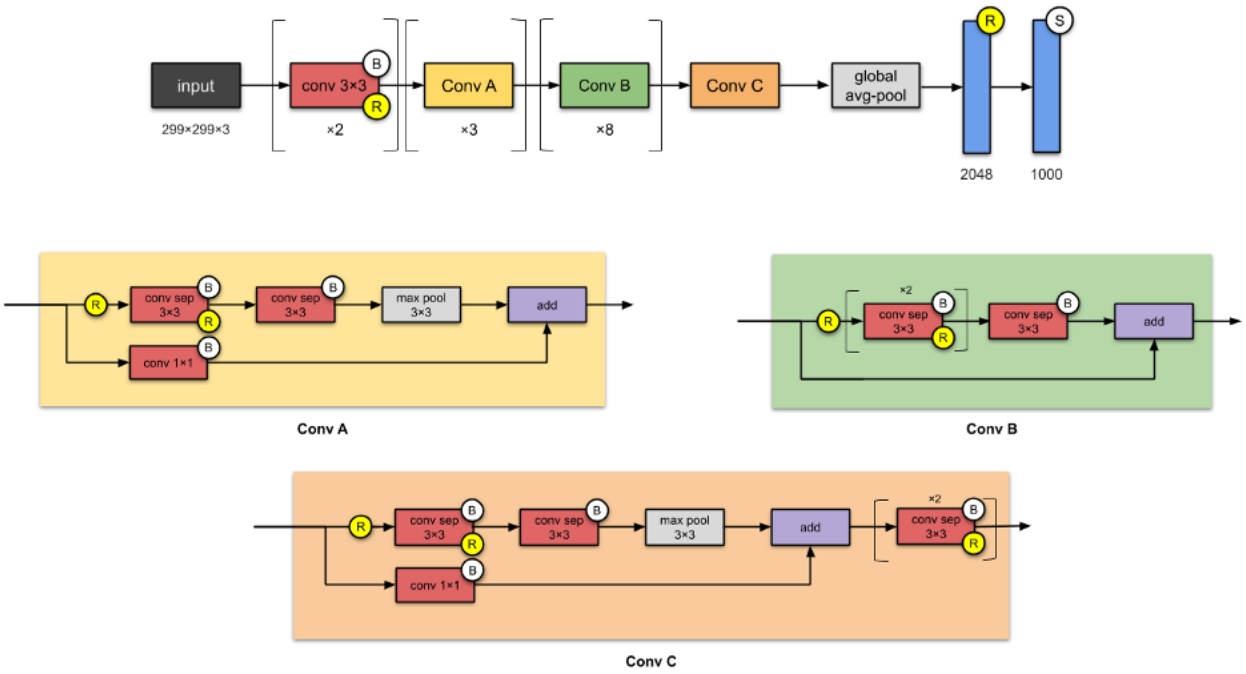


Fonte: (KARIM, 2019).

7. **Xception** é uma rede neural convolucional profunda inspirada no Inception, onde os módulos do Inception foram substituídos por convoluções separáveis em profundidade. A arquitetura Xception da Figura 2.20 possui o mesmo número de parâmetros que o Inception-V3, e mesmo assim conseguiu superar a Inception-V3 no conjunto de dados ImageNet para o qual o Inception-V3 foi projetado para classificar os dados (CHOLLET, 2016).

8. **MobileNetV2** foi desenvolvida para se utilizar em aplicações de visão nos dispositivos móveis. O MobileNet é baseado em uma arquitetura simplificada que utiliza conceito das convoluções separáveis em profundidade para criar uma rede neural pequena e rápida (HOWARD et al., 2017).

Figura 2.20: Ilustração da arquitetura *Xception*



Fonte: (KARIM, 2019).

2.3.4 Avaliação da Performance de Classificação

A maioria das métricas de avaliação de performance de classificação são derivadas de uma matriz de confusão. Segundo CONGALTON and GREEN (1999), a criação de uma matriz de confusão não é simplesmente uma questão de correto ou incorreto no caso binomial, e sim uma questão de qual ou quais categorias de erros estão confusas.

Para avaliar a efetividade de um algoritmo de aprendizado de máquina, normalmente utilizam-se conjuntos disjuntos de exemplos para treinar o modelo, i.e., um para o aprendizado do modelo e outro para testar o modelo.

Tabela 2.1: Matriz de Confusão para várias classes

Classes reais	Classe atribuídas pelo classificador			
	Classe 1	**Classe 2**	···	**Classe N**
Classe 1	n_{11}	n_{12}	···	n_{1N}
Classe 2	n_{21}	n_{22}	···	n_{2N}
···	···	···	···	···
Classe N	n_{N1}	n_{N2}	···	n_{NN}

Fonte: Adaptado de (SIMÕES and COSTA, 2007).

Na matriz de confusão apresentada na Tabela 2.1, as classes atribuídas

pelo modelo classificador estão em relação às classes previstas. A Tabela 2.1 é uma matriz de confusão para múltiplas classes a sua configuração está da seguinte forma: as classificações corretas estão representadas como $[n_{11}]$, $[n_{22}]$, ..., $[n_{NN}]$. Portanto, as classificações corretas são representadas nas linhas e colunas quando o número da linha for igual ao número da coluna. As classificadas erradas são as classes representadas nas linhas e colunas quando os números da linha forem diferentes como as $[n_{12}]$, $[n_{21}]$. Um exemplo de classificação errada seria $[n_{12}]$, que representa o número de animais da classe 1 que foram classificados como da classe 2 pelo classificador.

A partir da matriz de confusão é possível calcular métricas de Avaliação para a Classificação como Acurácia (ACC), que indica, dentre todas as classificações, quantas o modelo classificou corretamente. A acurácia é uma medida intuitiva e adequada para avaliação em cenários onde há um balanceamento de classes. Segundo CATRO and FERRARI (2016), a taxa global de sucesso do algoritmo, conhecida como a acurácia, é o número de classificação corretas dividido pelo número total de classificação, exemplo na equação 2.8.

$$ACC = \frac{n_c}{n_t}, \tag{2.8}$$

na qual n_t é número total de instâncias para o teste e n_c é o número de instância de teste classificada corretamente. O valor da acurácia varia de 0 a 1 (0 a 100%). Conforme os valores se aproximam de 1, maior será a confiança na estimativa da classificação, consequentemente, menor o risco de erro a partir do uso da informação.

2.3.5 Estratégias para Aumentar a Performance de classificação das CNNs

Segundo SHORTEN and KHOSHGOFTAAR (2019) utiliza-se de estratégias para aumentar o desempenho da generalização, estas técnicas se concentram na própria arquitetura dos modelos das CNNs. Estratégias como *Dropout regularization*, *Batch normalization*, *Transfer learning*. Podem ser utilizadas para aumentar a performance de classificação das CNNs.

O *Dropout* é uma técnica de regularização que zera os valores de ativação de neurônios escolhidos, durante o treinamento. Essa restrição força a rede a aprender recursos mais robustos em vez de confiar na capacidade preditiva de um pequeno subconjunto de neurônios na rede.

O *Batch normalization* é outra técnica de regularização que normaliza o conjunto de ativações em uma camada. A normalização funciona subtraindo a média do lote de cada ativação e dividindo pelo desvio padrão do lote. Essa técnica de normalização, juntamente com a padronização, é uma técnica pa-

drão no pré-processamento de valores de pixel.

O *Transfer Learning* é uma técnica que utiliza os pesos de uma rede treinada em um grande conjunto de dados, como o ImageNet e depois utiliza esses pesos como pesos iniciais em uma nova tarefa de classificação. Normalmente, apenas os pesos nas camadas convolucionais são copiados, em vez de toda a rede, incluindo camadas totalmente conectadas (SHORTEN and KHOSHGOFTAAR, 2019).

2.3.6 Transfer Learning

A *Transfer Learning* permite o treinar um modelo de classificação acuradamente utilizando uma menor quantidade de dados de treinamento. Isso é possível fazendo uso e refinando os padrões aprendidos por um modelo pré-treinado em um outro domínio, que não o domínio em questão. Porém, vale ressaltar que deve haver semelhanças entre os dois domínios de aplicação, e caso não exista, a *Transfer Learning* pode degradar a performance de classificação (AGGARWAL, 2015).

No caso das redes neurais convolucionais, normalmente o ajuste fino é feito nas camadas densas, ou seja, as camadas convolucionais são aprendidas em um outro domínio de aplicação, por exemplo, a ImageNet, e seus pesos são congelados. Apenas as conexões das camadas densas até a camada de saída são aprendidas (YOSINSKI, J et al., 2014).

Segundo OLIVEIRA and PRATI (2013) a *Transfer Learning* é uma alternativa para minimizar retrabalho com a grande quantidade de configuração de parâmetros que demanda muito tempo computacional, com testes para gerar bons resultados. A *Transfer Learning* só é útil, quando o modelo de Aprendizado de Máquina, que já foi treinado, é utilizado para transferir o conhecimento para outro modelo que contêm poucos dados, os dois modelos tem que ter o contexto próximo na vida real, se não tiverem o contexto próximo pode prejudicar o desempenho do modelo (SILVA, 2017).

2.4 Aumento de Dados para Aprendizado de Máquina

Nesta seção são apresentados conceitos utilizados para o aumento do conjunto de imagens para aprendizado de máquina. O conjunto de dados é importante para o desempenho do modelo gerado no aprendizado de máquina, e para melhorar o desempenho do modelo é possível aumentar os dados que já se tem, utilizando bibliotecas que fazem processamento de imagens. São exemplos de bibliotecas a imgaug, Keras e OpenCV (DEIS, 2019).

No artigo de SHORTEN and KHOSHGOFTAAR (2019), é descrito aumento de dados de imagem para o aprendizado profundo em redes neurais convo-

lucionais (CNNs). A pesquisa relata a importância da grande quantidade de dados para o bom desempenho da rede e assim evitar problemas como *overfitting*, que é o fenômeno quando o modelo aprende sinais e ruídos nos dados de treinamento, mas não tem um bom desempenho em novos dados sobre os quais o modelo não foi treinado.

O aumento de dados abrange um conjunto de técnicas que aprimoram o tamanho e a qualidade dos conjuntos de dados de treinamento de modo a melhorar o desempenho da rede neural. Alguns exemplos de transformações de imagens como rotação, deslocamento lateral e altura, inverter o lado da figura e dimensão ou escala, são apresentados na Figura 2.21.

2.5 Medidas de Proximidade

Medidas de similaridade ou dissimilaridade são utilizadas para calcular a proximidade entre dois objetos. Várias aplicações dentro da visão computacional podem fazer uso das medidas de proximidades, como o cálculo de grupos para a geração das *bag-of-visual-words*, na classificação baseada nos vizinhos mais próximos (apresentado a seguir), ou para detecção de cenas (DAVIES, 2017).

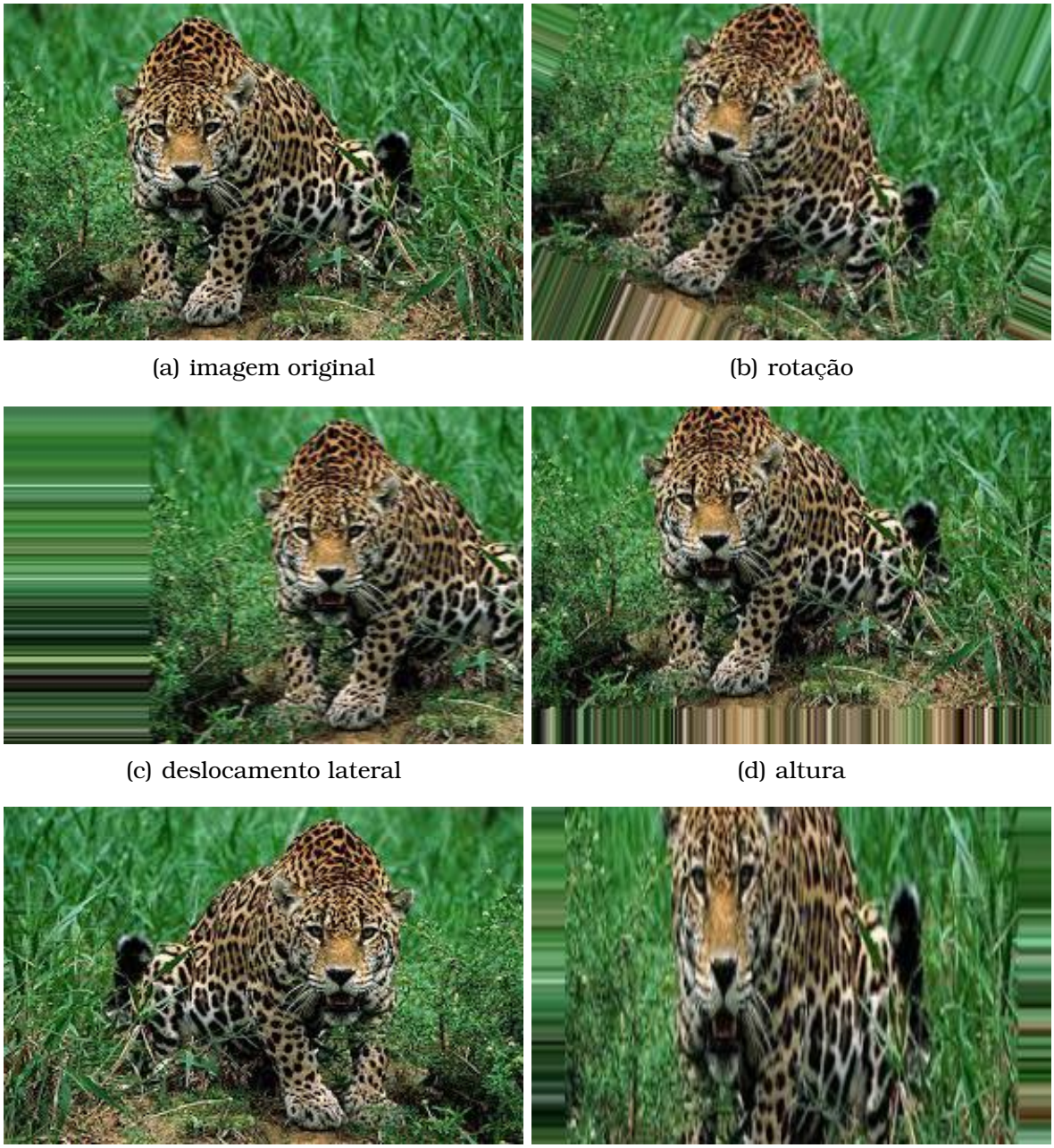
As medidas de proximidade aconselháveis para dados numéricos são a similaridade de Cosseno, distância Euclidiana e a correlação de Pearson (TAN et al., 2005). As imagens representadas por vetores podem ter a similaridade do Cosseno encontrada em pelo menos dois vetores, calculando o ângulo entre eles. A equação de similaridade do Cosseno (2.9), quando os vetores são semelhantes e o ângulo dos vetores é pequeno e aponta na mesma direção, representando assim a semelhança entre eles. Contudo, se o ângulo for muito grande, os vetores divergem indicando que não são semelhantes. O resultado sempre estará entre 1 e -1, quando for 1 é completamente semelhante e quando -1 completamente diferentes.

$$cos(\mathbf{x}_i, \mathbf{x}_j) = \frac{x_i \cdot x_j}{||x_i||||x_j||} = \frac{\sum_{k=1}^{m} x_{i,k} \cdot x_{j,k}}{\sqrt{\sum_{k=1}^{m} x_{i,k}^2} \cdot \sqrt{\sum_{k=1}^{m} x_{j,k}^2}} \tag{2.9}$$

A distância Euclidiana (2.10) é considerada uma medida de dissimilaridade e corresponde à distância em linha reta entre dois objetos em um espaço m-dimensional. A distância entre dois vetores que representam duas imagens (ou ponto de interesse), $\mathbf{x}_i$ e $\mathbf{x}_j$, respectivamente, é dada por:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{m} (x_{i,k} - x_{j,k})^2}, \tag{2.10}$$

Figura 2.21: Imagem original (a), Imagens modificadas de (b) até (f)



(a) imagem original (b) rotação

(c) deslocamento lateral (d) altura

(e) inverter o lado da figura (f) dimensão

Fonte: Autoria própria.

na qual $x_{i,k}$ e $x_{j,k}$ representam respectivamente o valor do k-ésimo, atributo das imagens i e j respectivamente.

Já a correlação de Pearson (2.11) mede a correlação linear entre os valores dos atributos dos objetos. Quanto maior a correlação, mais similares são os objetos. A interpretação dos coeficientes são: (i) $p = 1$ é de que existe correlação linear positiva perfeita entre os valores dos vetores $\mathbf{x}_i$ e $\mathbf{x}_j$; (ii); $p = -1$ representa um correlação linear negativa perfeita; e (iii) $p = 0$ significa que não existe correlação linear (LIRA, 2004). A correlação de Pearson para dois vetores $\mathbf{x}_i$ e

$\mathbf{x}_j$ é dada por:

$$p(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{k=1}^{m} (x_{i,k} - \overline{\mathbf{x}_i}) \cdot (x_{j,k} - \overline{\mathbf{x}_j})}{\sqrt{\sum_{k=1}^{m} (x_{i,k} - \overline{\mathbf{x}_i})^2} \cdot \sqrt{\sum_{k=1}^{m} (x_{j,k} - \overline{\mathbf{x}_j})^2}}, \tag{2.11}$$

na qual $\overline{\mathbf{x}_i}$ e $\overline{\mathbf{x}_j}$ representam respectivamente a média dos valores dos vetores $\mathbf{x}_i$ e $\mathbf{x}_k$

2.6 Trabalhos Relacionados

Nesta seção são apresentados os trabalhos encontrados na literatura que tratam do uso de visão computacional para o reconhecimento de animais.

2.6.1 Trabalhos que Utilizam CNNs

O trabalho de ANTONIO (2019) tem a proposta de detecção de animais em rodovias utilizando câmeras e aprendizado de máquina supervisionado. As câmeras são instaladas a partir de dados levantados, em pontos já identificados de travessia de animais. O motivo de colocar as câmeras em pontos determinados, é o risco de acidentes nas rodovias e a grande quantidade de atropelamento de animais silvestres, estimativas apontam que 475 milhões de animais selvagens são atropelados anualmente no Brasil.

A primeira etapa da proposta teve como função ensinar o algoritmo, a partir de características extraídas de um conjunto de imagens de treinamento selecionado, juntamente com o uso de uma técnica de aprendizado de máquina supervisionado escolhida, que no final gerou um modelo para classificação. Na segunda etapa, ao invés de um conjunto de treinamento, é utilizado um conjunto de imagens de teste, que é processado com o modelo resultante, obtido da etapa anterior, para que se obtenha a classificação.

O conjunto de dados foi criado com 20 imagens produzidas de maneira manual, através de programa de edição de imagem, mais dois conjuntos de imagens, um de imagens noturnas capturadas por uma câmera trap, embora não fosse especificado a quantidade de imagens utilizadas, e outro conjunto de imagens disponível na internet (AFKHAM, H. et al., 2008). Foram treinados com dois algoritmos de aprendizado de máquina supervisionados o k-Nearest Neighbors e Random Forest. O melhor resultado obtido foi da medida F de 0,6243 na combinação do espaço de cor GRAY com o algoritmo de segmentação de imagem ENTROPY, usando o algoritmo de AM k-Nearest Neighbors. A medida F é uma métrica de avaliação da performance de classificação, que corresponde à uma média harmônica simples das medidas precisão e revoca-

ção.

O artigo de NOROUZZADEH, M. S. et al. (2018) tem o objetivo de auxiliar a identificação dos animais, para ajudar os ecologistas nos estudos do tamanho e distribuição de suas populações. Basearam o estudo no conjunto de imagens do projeto Snapshot Serengeti, um projeto de coleta de imagens por armadilhas fotográficas (câmeras-trap), com 225 câmaras que funcionam continuamente no Parque Nacional Serengeti, em Tanzânia no continente Africano, desde 2011. Neste artigo, os autores ressaltam a importância da automatização do reconhecimento das espécies, descrevendo como é grande a economia em tempo de análise de imagens em relação à análise de forma manual.

A solução utilizada pelos autores para detectar imagens que contêm animais, foi baseada em nove arquiteturas diferentes para encontrar as redes com melhor desempenho e comparar os resultados. O treinamento de cada modelo foi feito uma única vez, porque isso é computacionalmente caro e as evidências teóricas e empíricas sugerem que redes neurais profundas *deep neural networks* (DNNs) diferentes, treinadas com a mesma arquitetura, mas iniciadas de maneira diferente geralmente convergem para níveis de desempenho semelhantes.

O conjunto de treinamento continha 1,4 milhão de imagens e o conjunto de teste continha 105 mil imagens. Foram treinadas as seguintes DNNs: AlexNet, Network in Network, VGG, GoogleNet, ResNet-18, ResNet-34, ResNet-50, ResNet-101 e ResNet-152, para identificar 48 espécies no conjunto de dados Snapshot Serengeti[1], composto por 3,2 milhões de imagens usadas para estudar ecossistemas. Todas as arquiteturas alcançaram uma precisão de classificação maior de 95,8% de acurácia, o modelo VGG alcançou a melhor precisão de 96,8% de acurácia para detectar se tem animal ou não na imagem. Já o melhor modelo para identificar espécies foi o ResNet-152 que obteve 93,8% acurácia com top 1 e 98,8% de acurácia com top 5, na qual top 1 significa ser a classe melhor classificado e o top 5 significa estar entre as cinco melhores classificadas. Com esta solução as equipes de voluntários teriam uma economia de 8,4 anos de esforço de rotulagem humana no conjunto 3,2 milhões de imagens. Em NOOR, A. et al. (2020) é apresentado um trabalho que tem o objetivo de identificar quais ovelhas estão com dor e assim descobrir quais ovelhas estão com a saúde afetadas, que atrapalha o seu crescimento por algum tipo de doença. A proposta para resolver este problema, foi desenvolver um conjunto de dados para estimar o nível de dor a partir das expressões faciais das ovelhas. Foi feito uma pesquisa para gerar um modelo classificador que pudesse classificar de forma eficiente quando a imagem da face da ovelha está normal (expressão sem dor) e face anormal (expressão com dor), para

[1]Snapshot Serengeti: `https://www.zooniverse.org/projects/zooniverse/snapshot-serengeti`

isso utilizou aprendizado de máquina com a técnica de *transfer learning*. Foi desenvolvido uma base de dados utilizando a técnica de aumento de dados (Data Augmentation), a base de dados com 1650 imagens para treinamento, 350 imagens para validação. Entre este conjunto de dados, 1400 imagens de ovelhas saudáveis e 900 eram imagens de ovelhas com expressão de dor. Diferentes arquiteturas foram testadas com a técnica de *transfer learning*, como AlexNet, VGG-16, GoogleNet, DenceNet-201, Inception-V3, ResNet-50 e DarkNet para classificação binária das expressões dos rostos das ovelhas. O melhor desempenho foi de 100% e 98% de acurácia para o conjunto de dados do treinamento com 99,69% e 97,8% de acurácia para o conjunto de dados de validação (Com as arquiteturas VGG16 e ResNet-50 respectivamente).

2.6.2 Trabalhos que Utilizam Regiões ou Pontos de Interesse

No trabalho de ARRUDA (2018) é apresentada uma abordagem para detectar e identificar as espécies de animais do pantanal usando imagens térmicas e CNNs. O objetivo do trabalho foi auxiliar os pesquisadores a proteger a biodiversidade e espécies ameaçadas nos seus habitats naturais, dentro do Pantanal. As imagens térmicas captam a radiação infravermelha emitida pelos objetos com uma temperatura acima do zero absoluto, podem ser utilizadas tanto de dia quanto à noite, e permitem identificar o animal em diferentes condições de iluminação. Na etapa de treinamento para classificação dos animais, foram utilizadas as regiões de interesse obtidas pelas saídas do algoritmo *Simple Linear Iterative Clustering*-(SLIC), que agrupa os pixels da imagem em regiões que possuam similaridade baseada na emissão de calor dos animais. E o superpixel que permite usar as características locais dos pixels da imagem, é uma abordagem que verifica se a temperatura de um superpixel é maior que a temperatura média do ambiente. Os experimentos foram feitos com uma abordagem que contém CNN bem similar ao Fast R-CNN, porém utiliza o conceito SLIC em um conjunto de dados de 1.600 imagens. As arquiteturas de redes convolucionais utilizadas no trabalho foram: AlexNet, VGGNet e GoogleNet. A avaliação da abordagem foi feita pela comparação Fast-CNN, utilizando duas métricas: precisão média e medida F para todas as classes de animais. Os resultados obtidos tiveram um ganho médio entre 6% e 10% quando comparado ao método Fast R-CNN.

O problema abordado no artigo de BORTH, M. R. et al. (2013) refere-se ao estudo de extração de características de imagens de peixes para automatização do reconhecimento de espécies. Foi aplicada a técnica SURF e foi analisada a correspondência dos pontos de interesse em peixes de mesma espécie considerando variações nas imagens, principalmente a angulação. O objetivo do trabalho é ajudar ecologistas e estudiosos do meio ambiente e órgãos go-

vernamentais e empresas de pescados. Podem ser criadas soluções através da extração de informações visuais como, formas, cor e textura, com objetivo de identificar características das imagens para a utilização em um sistema de reconhecimento de padrões com SURF, que extrai pontos de interesse de imagem e posteriormente compara esses pontos de interesse com os pontos de interesse de outra imagem. Se os pontos de interesse forem semelhantes, significa que o objeto da primeira imagem provavelmente está na segunda imagem. Para calcular a semelhança entre os pontos de interesse foi utilizado o cálculo de distância de Mahalanobis ou Euclideana. Foi feito o teste entre a semelhança dos pontos de interesse e verificou-se erros no casamento de pontos de interesse entre imagens diferentes, mesmo quando o peixe era da mesma espécie. Foi necessário reduzir o *threshold* para quase zero, para encontrar o casamento entres os pontos de interesse. Por outro lado, conseguiu-se bons resultados de identificação quando comparada a imagem com qualquer uma das rotações de (0º a 90º). O teste foi feito com o algoritmo SURF na linguagem Java por meio do projeto Jopensurf.

O trabalho SIVIC and ZISSERMAN (2003) tem o objetivo de encontrar um objeto específico com facilidade, velocidade e precisão que o Google recupera documentos de texto em páginas *Web*. O problema é que um objeto pode ter uma variação muito grande dependendo do ponto de vista. A solução utilizada pelo autor foi a abordagem da *Bag of Visual Words (BoVW)*. Extraindo pontos de interesse nas regiões invariantes, no trabalho foi utilizado o algoritmo SIFT para extrair as características e representá-las por um vetor de 128 dimensões através do descritor SIFT. Segundo o autor, é inviável agrupar todos os descritores do filme, por isto foram selecionadas apenas 48 fotos, ainda assim, teriam em média 200 mil descritores. Foi criada uma função de distância para agrupar, sendo utilizada a distância Mahalanobis, para fazer a filtragem nos descritores. Para poder recuperar as imagens foi feita a analogia com a recuperação de texto, com computação da correspondência dos descritores, sendo necessária a construção de um vocabulário de palavras visuais. Isto foi feito com histograma das frequências dos descritores agrupados, que foram as palavras visuais para a recuperação e o histograma de frequências foi criado por meio do algoritmo k-Means.

Apesar de cada um dos trabalhos relacionados terem utilizados pelo menos uma abordagem e no máximo três das abordagens que também foram abordadas neste trabalho o nosso trabalho se diferencia, pois analisa as abordagem de uso da BoVW, com uso de dois algoritmos de Aprendizado de Máquina (AM), Redes Neurais Convolucionais (CNNs), treinamento com *transfer learning* e a técnica de aumento de dados, para identificar quais das abordagens geram os melhores resultados para detectar se tem um predador na imagem.

Além disso, os trabalhos BORTH, M. R. et al. (2013) e o SIVIC and ZISSERMAN (2003), utilizaram algoritmos de extração de pontos de interesses mas não utilizaram as abordagens das CNNs. Na Figura 2.22 é apresentado uma análise das abordagens que também foram utilizadas nos trabalhos relacionados:

Figura 2.22: Abordagem utilizadas nos trabalhos relacionados

Estratégias utilizadas nos trabalhos relacionados					
Trabalhos relacionados	**Uso da BoVW**	**Algoritmos de Aprendizado de Máquina (AM)**	**Redes Neurais Convolucionais (CNNs)**	**Transfer learning**	**Aumento de dados**
T1	Não	Sim	Não	Não	Não
T2	Não	Não	Sim	Não	Não
T3	Não	Não	Sim	Sim	Sim
T4	Não	Não	Sim	Não	Não
T5	Não	Não	Não	Não	Não
T6	Sim	Sim	Não	Não	Não

Fonte: Autoria própria.

Legenda com os apelidos utilizados para os trabalhos relacionados na Figura 2.22:

- Trabalho (T1) (ANTONIO, 2019);
- Trabalho (T2) (NOROUZZADEH, M. S. et al., 2018);
- Trabalho (T3) (NOOR, A. et al., 2020);
- Trabalho (T4) (ARRUDA, 2018);
- Trabalho (T5) (BORTH, M. R. et al., 2013);
- Trabalho (T6) (SIVIC and ZISSERMAN, 2003).

CAPÍTULO

3

Proposta: Método para a Detecção Automática de Predadores

Neste capítulo são descritas as tecnologias, a arquitetura que foi utilizada no desenvolvimento da proposta, bem como as bibliotecas e *hardwares* utilizados. Uma visão geral da arquitetura proposta é apresentada na Figura 3.1. Basicamente, o sistema proposto irá capturar imagens, as quais serão transmitidas a um servidor, o qual, por sua vez, reconhecerá um predador na imagem, emitirá alertas em páginas *web* ou dispositivos *mobile*, e também armazenará as informações pertinentes em um banco de dados.

A arquitetura proposta está dividida em módulos, os quais são apresentados na Figura 3.2. A descrições detalhadas destes módulos são apresentadas nas próximas seções

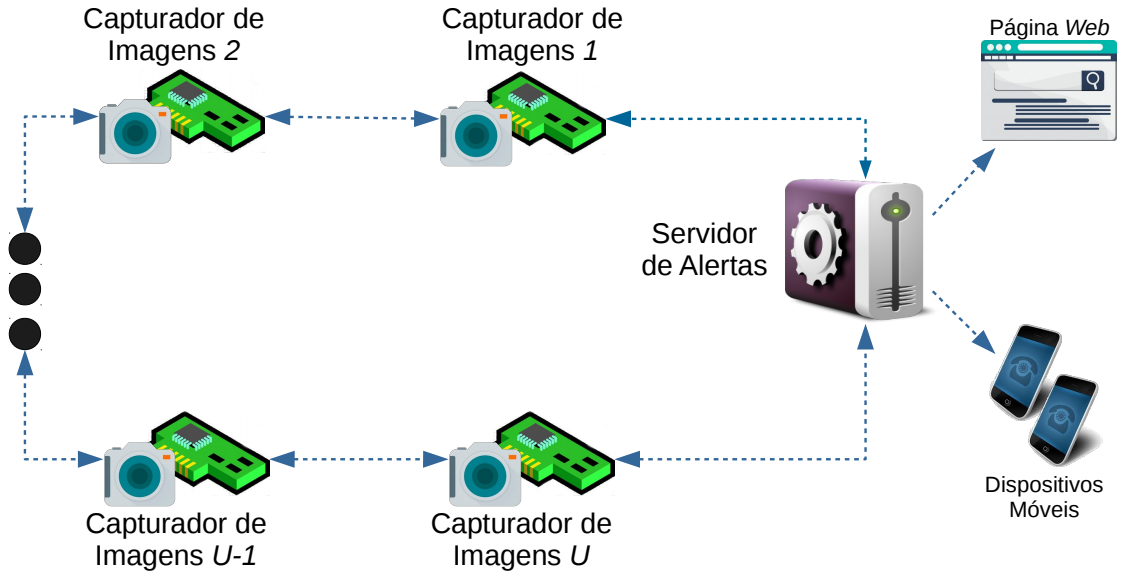
3.1 Módulo 1: Detecção de Movimento

O módulo Detecção de Movimento tem o objetivo de enviar a imagem para o servidor de alertas. Porém, visando diminuir o tráfego de rede, só serão enviadas imagens para o servidor quando forem detectados objetos novos na imagem, de maneira que o próprio *hardware* utilizado para a captura da imagem possa detectá-los.

O *hardware* escolhido para manter este módulo de detecção de movimento do projeto é o microcontrolador *Raspberry Pi 3B* em conjunto com uma Câmera OmniVision-OV5647 de visão noturna[1][2]. O *Raspberry Pi* possibilita a

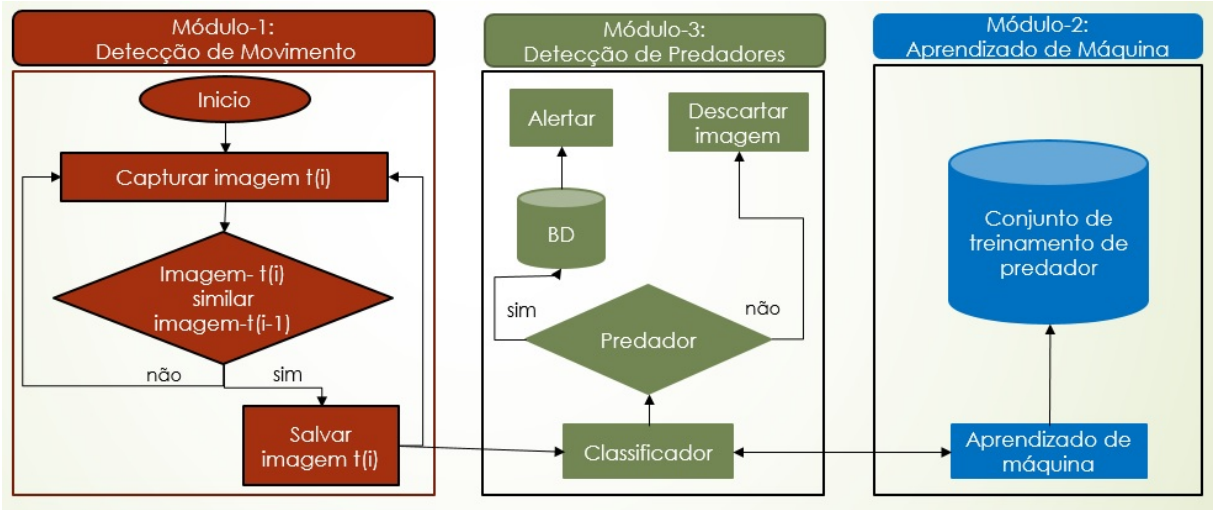
[1]Raspberry:https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md
[2]Descrição da câmera OmniVision Ov5647 Megapixel:http://pdf1.alldatasheet.com/datasheet-

Figura 3.1: Visão geral da abordagem proposta.



Fonte: Autoria própria.

Figura 3.2: Ilustração dos módulos, componentes dos módulos e interação entre os módulos.



Fonte: Autoria própria.

instalação de diversos sistemas operacionais e consequentemente a utilização de diversas linguagens de programação.

Este módulo foi desenvolvido na linguagem de programação Java, e foi utilizada a biblioteca *Open Source Computer Vision Library* (OpenCV)[3], a qual é uma biblioteca de visão computacional e aprendizado de máquina. Neste módulo foi utilizado a função para receber a imagem e armazenar em um objeto

pdf/view/587045/OMNIVISION/OV5647.html

[3]OpenCV - https://opencv.org/about/

"Mat", também do OpenCV, que disponibiliza a possibilidade de manipular os valores do RGB. Além disso, foi utilizada uma função do OpenCV para salvar a imagem, depois de submetida ao cálculo de proximidade utilizando a correlação de Pearson (usada para comparar as imagens conforme mencionado na Seção 2.5), e assim enviá-la ao servidor de alertas.

O sistema consiste em um *loop* que captura imagens em determinados períodos de tempo, para comparar as imagens, de forma a verificar se algum evento ocorreu. Também é necessário o uso de um limiar no processo, para avaliar se a imagem vai ser descartada ou utilizada. Foram testadas as medidas de proximidade descritas na Seção 2.5, comparando a similaridade ou dissimilaridade entre as duas imagens subsequentes. Caso a similaridade for menor que o limiar, é considerado que um novo objeto surgiu na imagem, caso a dissimilaridade estiver acima de um limiar, também é considerado que um novo objeto surgiu na imagem.

Após analisar resultados gerados por ambas as medidas e em diferentes situações, optou-se pelo uso da correlação de Pearson, principalmente devido à definição de limiares, já que o intervalo de valores desta medida é bem definido ([-1,1]). Com base em testes empíricos, foi estabelecido o limiar de 0,95.

Ao considerar que um novo objeto surgiu na imagem, essa deverá ser enviada ao servidor de alertas. Para o envio da imagem do Módulo 1 para o Módulo 2, foi utilizada uma instancia da classe Runtime da linguagem Java, para que se possa fazer uso do aplicativo `curl` para fazer a submissão da imagem através do método `POST`. Além disso, para facilitar a comunicação, diminuir esforço e custos, da infraestrutura de comunicação, uma possibilidade é utilizar redes *wireless* do tipo Mesh, na qual cada nó da rede pode se comportar como um transmissor de informações (AKYILDIZ and WANG, 2009). Com isso, dispensa-se a infraestrutura de cabos necessária e pode-se fazer um monitoramento em grandes áreas, desde que os dispositivos de coletas de imagens sejam capazes de alcançar uns aos outros na comunicação sem fio.

3.2 Módulo 2: Aprendizado de Máquina

O módulo de aprendizado de máquina é responsável por aprender um modelo de classificação. Este modelo é responsável por reconhecer predadores em imagens e está sendo utilizado no Módulo-3, Reconhecimento de Predadores.

Para tanto, foi necessário avaliar qual é a melhor estratégia que proverá um reconhecimento mais acurado dos predadores, que considerou os algoritmos de aprendizado de máquina e as técnicas de processamento de imagens, descritos na Seção 2, em uma base de dados composta por 2.458 imagens de

6 diferentes tipos de animais.

Para o aprendizado de máquina, foram consideradas as bibliotecas TensorFlow[4], Scikit-Learn[5] e a Keras[6].

A biblioteca OpenCV também foi utilizada para o pré-processamento das imagens para a geração de represetações do tipo BoVW. Foram utilizadas funcionalidades para conversão para escala de cinza, eliminação de ruídos da imagens, equalização de histogramas de cores, aplicação da técnica de Canny, e para encontrar pontos de interesse nas imagens. Já a biblioteca Scikit-Learn foram utilizadas as funcionalidades para gerar as matrizes de confusão e computar as métricas de performance de classificação, para gerar as palavras visuais das representações BoVW utilizando o algoritmo *k-Means*, para calcular os histogramas de frequências de palavras das imagens utilizando o algoritmo *k-Nearest Neighbors*, além do algoritmos de aprendizado de máquina para a construção de modelos de classificação *k-Nearest Neighbors* e *Support Vector Machines*. Já a biblioteca Keras foi utilizada para criar e treinar as CNNs. Os resultados das performances de classificação dos algoritmos de aprendizado de máquina são apresentados na Seção 5.

3.3 Módulo 3: Detecção de Predadores

O módulo de detecção de predadores tem o objetivo de gerar alertas para que sejam tomadas medidas preventivas ao ataque de predadores. Cada predador detectado será armazenado em um banco de dados para gerar futuras estatísticas ou mesmo retreinar os modelos de classificação. Também será gerada a entrada em um serviço *web*, o qual poderá alimentar sites ou ainda enviar mensagens a dispositivos móveis cadastrados, de forma que o responsável pelo local esteja ciente da iminência de um ataque.

O módulo de detecção de predadores foi implementado utilizando o módulo Flask na linguagem Python. O Flask é um *framework* para a criação de microserviços baseado em Werkzeug e Jinja 2. O Werkzeug é uma biblioteca de utilitários para Interface de Porta de Entrada do Servidor Web do inglês *Web Server Gateway Interface* (WSGI). O WSGI é um protocolo que faz a comunicação entre a aplicação Web e o servidor Web. Já o Jinja é um mecanismo de templates para Python que facilita a tarefa de usar HTML dentro da aplicação (OLIVEIRA, 2019).

O Módulo 3 - Detecção de Predadores fica aguardando imagens enviadas pelo Módulo 1 - Detecção de Movimento, e utiliza o modelo de classificação gerado pelo Módulo 2 - Aprendizado de Máquina. De acordo com o resultado

[4] https://www.tensorflow.org/
[5] https://scikit-learn.org/
[6] https://keras.io/

da classificação, será gerado um alerta e a imagem poderá ser descartada caso não haja uma espécie predadora na imagem. Foi utilizado o método Image da biblioteca Python Imaging Library - (PIL) para ler os metadados da imagem recebida, antes de passar o mesmo para o modelo de classificação e prever a qual classe a imagem pertence.

CAPÍTULO

4

Configurações utilizadas nas Avaliações Experimentais

Neste capítulo são apresentados as configurações utilizadas nas avaliações experimentais para o reconhecimento automático de predadores considerando diferentes técnicas de processamento de imagens, algoritmos de aprendizado de máquina e dois conjuntos de dados com diferentes classes. Os detalhes sobre algoritmos usados, parâmetros, esquemas de avaliação, técnicas de pré-processamento, conjuntos de dados e os resultados obtidos são apresentados nas próximas seções.

4.1 Coleções de Imagens

Uma parte das imagens utilizadas no processamento de imagens foram disponibilizadas através de uma parceria entre a Embrapa e o Instituto Homem Pantaneiro. A base é composta por imagens e vídeos de 24 espécies de animais. Devido ao número de imagens por categoria ser considerado baixo, também foram extraídas imagens de frames dos vídeos, gerando assim a quantidade inicial de imagens por categoria apresentada na Tabela 4.1.

Mesmo com a extração das imagens dos vídeos, a quantidade de imagens por categoria foi considerada baixa, principalmente para o treinamento de algoritmos de aprendizado profundo. Foram então coletadas imagens para as 6 categorias do conjunto de dados, obtidas por resultados de busca no site Google Imagens[1], para aumentar a quantidade também foram adicionadas as

[1] https://www.google.com/imghp?hl=pt-BR

imagens de cães e cavalos que foram encontradas no banco de dados "Animal-Dataset"no site Kaggle [2].

Estabeleceu-se com a orientação da Embrapa, um conjunto de categorias de animais que seria de interesse para testar os modelos de classificação. As categorias definidas foram: onça pintada, onça parda, cão, ovino, boi e cavalo; Estas classes são consideradas nos experimentos dos modelos com classificação multiclasse e como predador são consideradas as seguintes classes: onça pintada, onça parda e cão. Vale ressaltar que os resultados que serão reportados a seguir consideram essas 2.458 imagens descritas na Tabela 4.2 para treinamento e 240 imagens descritas na Tabela 4.3 para validação. Vale ressaltar que as imagens do conjunto de validação são de fonte independente das imagens do conjunto de treinamento.

No modelo binário, só foram consideradas as classes Predador x Não Predador. A categoria Predador foi gerada utilizando as espécies Onça pintada e Onça parda. Já para a categoria Não Predador foram utilizadas as espécies Ovino, Boi, Cão e Cavalo. A base de dados para treinamento totalizou 590 imagens para Predador e para Não Predador 1.868 imagens. Já a base de dados de validação ficou com um total de 80 imagens para Predador e 160 imagens para Não Predador. No modelo multiclasses, cada classe foi identificada individualmente. A base de dados para treinamento foi de 2.458 imagens e a base de dados para validação ficou 240 imagens.

4.2 Configuração Experimental

Para apresentar as configurações utilizadas nos experimentos, esta seção está organizada em duas partes. Na primeira parte são apresentadas as configurações utilizadas na abordagem da BoVW, e na segunda as configurações utilizadas nas CNNs.

4.2.1 Configurações Utilizadas na Técnica BoVW

Todas as imagens, com os respectivos pré-processamentos listados a seguir, foram submetidas aos algoritmos ORB para a detecção de pontos de interesse. Uma vez extraídos os pontos de interesse, foram geradas diferentes quantidades de grupos para gerar, consequentemente, diferentes quantidades de palavras visuais (dicionário). Para gerar o dicionário de palavras da técnica BoVW foi utilizado o algoritmo k-Means da biblioteca Scikit-Learn, e o algoritmo k-Nearest Neighbor na para atribuir uma palavra a um grupo de palavras e consequentemente gerar o histograma de frequências de palavras visuais. Com isso, foram geradas representações BoVW com as seguintes

[2] https://www.kaggle.com/navneetsurana/animaldataset

Tabela 4.1: Coleção de imagens inicial.

Coleção de imagens inicial			
Classe	Vídeo	Imagem	**Total por classe**
1-Onça pintada	21	3	24
2-Ovino	0	12	12
3-Boi	0	12	12
4-Anta	9	3	12
5-Capivara	4	0	4
6-Cateto	1	5	6
7-Cutia	4	0	4
8-Gato Mourisco	3	4	7
9-Irara	6	3	9
10-Jaguatirica	15	14	29
11-Lobinho	5	0	5
12-Macaco prego	3	0	3
13-Mão pelada	0	1	1
14-Mutum	4	0	4
15-Onça parda	12	14	26
16-Quati	1	0	1
17-Queixada	18	6	24
18-Seriema	1	0	1
19-Tamanduá bandeira	6	0	6
20-Tapiti	2	0	2
21-Tatu canastra	2	0	2
22-Tatu galinha	2	0	2
23-Veado catingueiro	2	2	4
24-Veado mateiro	6	6	12
Total	127	85	212

Fonte: Autoria própria.

Tabela 4.2: Número de imagens por classe do conjunto de treinamento.

Base de dados usada no treinamento durante os experimentos	
Classe	**Total de imagens por classe**
1-Onça pintada	290
2-Ovino	109
3-Boi	576
4-Onça parda	300
5-Cão	443
6-Cavalo	740
Total de imagens	**2.458**

Fonte: Autoria própria.

quantidades de atributos: 50, 100, 200, 500, 1.000, 5.000, 10.000, 25.000 e 50.000.

Foram então aplicados os seguintes algoritmos e respectivos parâmetros

Tabela 4.3: Número de imagens por classe do conjunto de teste.

Base de dados usada na validação com imagens diferente do treinamento	
Classe	**Total de imagens por classe**
1-Onça pintada	40
2-Ovino	40
3-Boi	40
4-Onça parda	40
5-Cão	40
6-Cavalo	40
Total de imagens	**240**

Fonte: Autoria própria.

nas representações BoVW:

1. **SVM**: foram considerados os dois tipos de *kernel* mais comuns, *linear* e *radial basis function*, sendo que o *kernel linear* é o mais aconselhável para representações com alta dimensionalidade (CARUANA and NICULESCU-MIZIL, 2006). Como C é um parâmetro real, positivo e sem limite, o valor desse parâmetro é normalmente definido por 10^x para representações com alta dimensionalidade. Para cada tipo de *kernel* foram utilizados valores de $C \in \{10^{-1}; 10^0; 10^1\}$ conforme (CARUANA and NICULESCU-MIZIL, 2006).

2. ***k-NN***: foi utilizado o algoritmo *k-NN* com e sem voto ponderado pela distância. Foi considerado $k \in \{1; 5; 10; 15; 20\}$. A métrica utilizada é a similaridade cosseno.

Todas as imagens foram submetidas aos seguintes pré-processamentos:

- Conversão de RGB para escala de cinza;
- Equalização de histogramas;
- Filtro gaussiano (foram utilizado os parâmetros de altura igual a 9 e largura igual a 9 para o kernel gaussiano);

Além das técnicas de pré-processamento apresentadas acima, também foi utilizada a técnica de Canny. Esta técnica possui dois parâmetros: valor mínimo igual a 100, e valor máximo 220, sendo que arestas com valor de gradiente menor que *100* são descartadas e arestas com valor maior que o parâmetro *220* são consideradas. Aresta entre esses dois limiares são consideradas como bordas, caso estejam conectadas a outras bordas detectadas.

Os resultados da BoVW, apresentados na Seção 5, são avaliados a partir da acurácia gerada pelos modelos treinados com AM, que variam conforme a quantidade de atributos da representação, e de acordo com a aplicação ou não da técnica de Canny.

4.2.2 Configurações Utilizadas nas CNNs

Para os experimentos foi utilizada a linguagem de programação Python, com vistas ao treinamento e teste das CNNs. Os modelos CNNs utilizados foram MobileNetV2, InceptionResNetV2, InceptionV3, ResNet50V2, VGG16 e Xception, sendo estas CNNs os modelos criados no treinamento do conjunto de dados ImageNet, um grande conjunto de dados que consiste de 1,4 milhões de imagens e 1 mil classes.

Foi utilizada a classe ImageDataGenerator[3] da biblioteca para padronizar os valores dos pixels das imagens entre 0 e 1 (por questões de convergência). O ImageDataGenerator também foi utilizado para o aumento de dados, conforme mencionado na Seção 2.4, possibilitando assim modificar e aumentar a quantidade de imagens. Foram atribuídos os seguintes parâmetros para o ImageDataGenerator:

- `rotation_range` = 40: para rotação da imagem em 40 graus;
- `width_shift_range = 0.2`: deslocamento lateral;
- `height_shift_range = 0.2`: deslocamento na altura;
- `zoom_range = 0.2`: quantidade de zoom aplicado na imagem;
- `horizontal_flip = True`: inverter a imagem no eixo horizontal;
- `vertical_flip = True`: inverter imagem no eixo vertical;
- `fill_mode = 'nearest'`: utilização do algoritmo *Nearest* para a interpolação da imagem.

As configurações utilizadas nas arquiteturas das *CNN*s foram:

- `weights='imagenet'`: utilizado para carregar os pesos dos modelos pré-treinado no conjunto de dados ImageNet.
- `include_top = False`: retira as camadas densas (intermediárias e de saída) da *CNN*.
- `layer.trainable = False`: com esta configuração é possível congelar as camadas convolucionais e o treinamento é realizado apenas nas camadas densas.
- `tf.keras.layers.GlobalAveragePooling2D`: utilizado para realizar o *flattening* dos filtros de *pooling* para gerar dados de entrada para a parte densa da rede.

[3]https://keras.io/api/preprocessing/image/

- `tf.keras.layers.Dense(1024,activation='relu')`: camada intermediária densa com 1024 neurônios e função de ativação *Relu*

- `tf.keras.layers.Dropout(0.2)`: é a técnica que foi apresentada na Seção 2.3.5, que é utilizada para restringir e forçar a rede a aprender recursos mais robustos.

- `tf.keras.layers.Dense(6,activation='softmax')`: camada de saída com o número de neurônios equivalente ao número de classes no cenário de classificação multiclasse, e função de ativação *Softmax* para gerar as probabilidades das categorias nas camadas de saída.

- `tf.keras.layers.Dense(1,activation='sigmoid')`: camada de saída com o número de neurônios equivalente ao número de classes no cenário de classificação binária, e função de ativação *Sigmoid* para gerar valores de saída entre 0 e 1, sendo que valores ≤ 0 significam que a imagem não contém um predador e valore > 0 significam que a imagem contém um predador.

Os resultados das CNNs, que serão apresentados na Seção 5.1.3 e 5.1.6, correspondem a uma média de dez inicializações aleatórias dos pesos das conexões de cada CNN. Também são apresentadas a acurácia média e máxima considerando todas as inicializações de pesos.

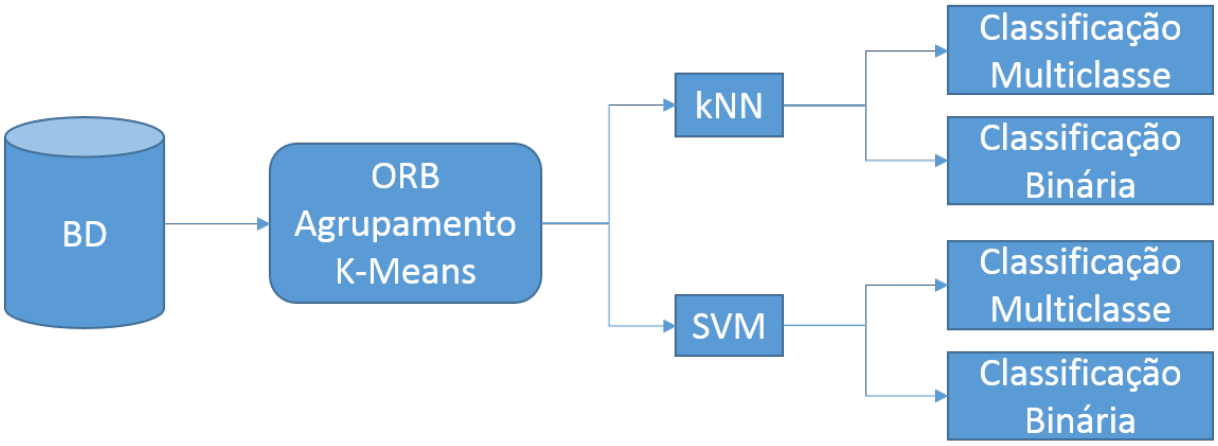
4.2.3 Organização dos experimentos

Os experimentos foram organizados em classificação multiclasse e classificação binária.

- Nos experimentos utilizando representações BoVW, foram geradas BoVWs com diferentes quantidades de dimensões. Também foram geradas BoVWs com e sem a aplicação da técnica de Canny. Na Figura 4.1 é apresentado a organização do experimento BoVW sem e com a utilizar da técnica de Canny.

- A organização dos experimentos utilizando redes pré-treinadas é apresentada na Figura 4.2 e na Figura 4.3. Para poder utilizar os pesos pré-treinados das CNNs MobiliNetV2, ResNet50V2 e VGG16, o tamanho de entrada da imagem deve ser de 244x244x3 e nas CNNs Inception-ResNetV2, InceptionV3 e Xception, o tamanho da imagem deve ser de 299x299x3. Vale ressaltar que neste experimento foi feito um mapeamento para as 1000 classes do ImageNet. O mapeamento foi feito da seguinte forma:

Figura 4.1: Organização do experimento da BoVW sem a utilização da técnica de Canny.



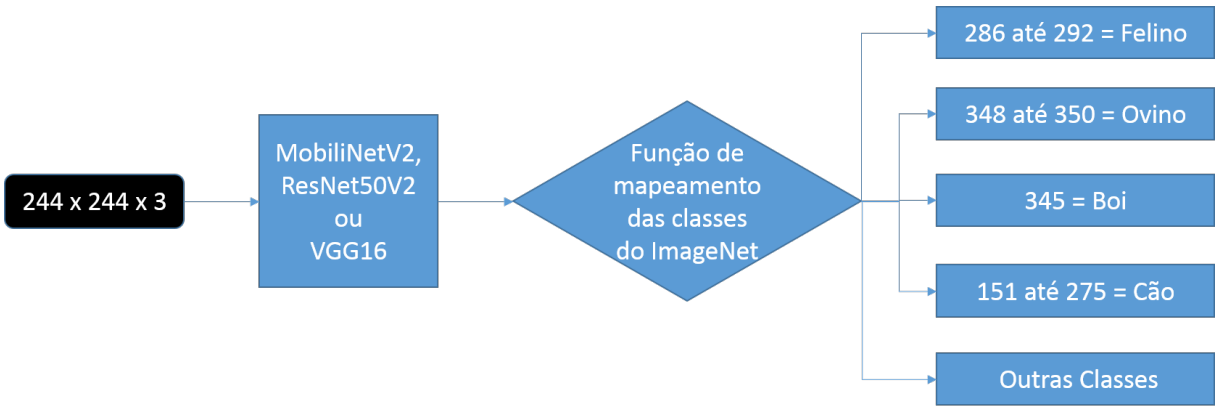
Fonte: autoria própria.

- Foi utilizado uma lista provida do GitHub [4] que mostra quais são os ID e nomes das classes da ImageNet
- As classes com valores entre 286 a 292 são considerados da classe `Felino`;
- As classes com valores entre com valores entre 348 e 350 são considerados `Ovino`;
- A classe com valor 345 for considerada `Boi`;
- As classes com valores entre com valores entre 151 e 275 são considerados `Cão`;
- As demais classes foram consideradas como `Outros`.

- Na Figura 4.4 é apresentada uma legenda para o melhor entendimento sobre os experimentos utilizando CNNs sem *transfer learning* e experimentos de CNNs com *Transfer learning*;
- O experimento de treinamento sem *transfer learning* foi feito com dois tamanhos de entrada de imagens. Na Figura 4.5 é apresentado com entrada 128x128x3 e na Figura 4.6 a entrada 224x224x3;
- O experimento de treinamento com *transfer learning* foi feito com dois tamanhos de entrada de imagens. Na Figura 4.7 é apresentado com entrada de 128x128x3 e na Figura 4.8 a entrada de 224x224x3;

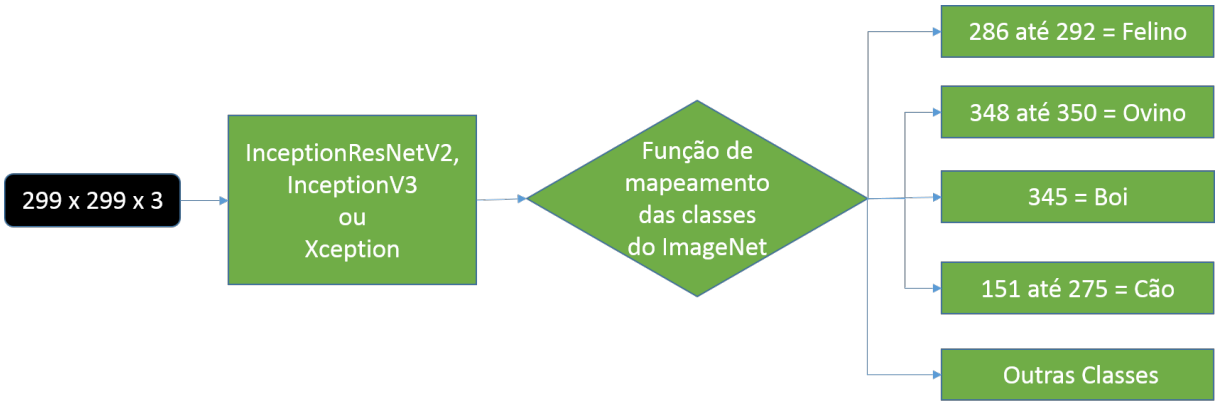
[4] `https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a`

Figura 4.2: CNNs com pesos pré treinados e tamanho da entrada 244x244x3.



Fonte: autoria própria.

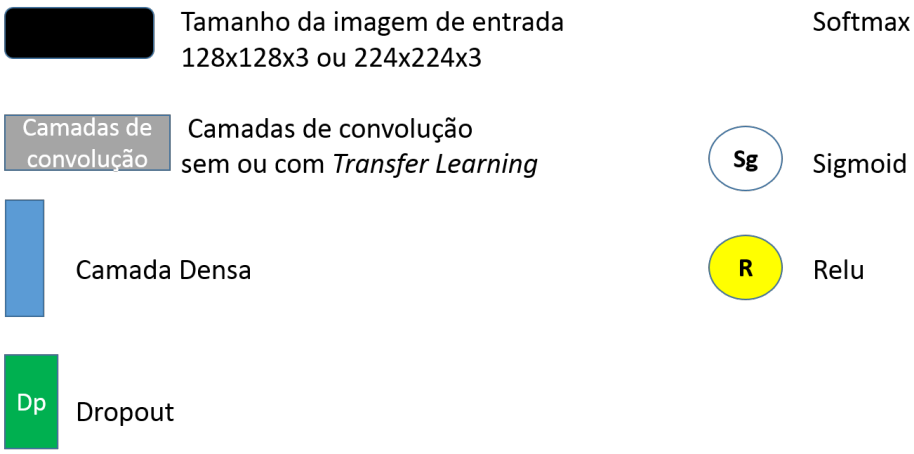
Figura 4.3: CNNs com pesos pré treinados e tamanho da entrada 299x299x3.



Fonte: autoria própria.

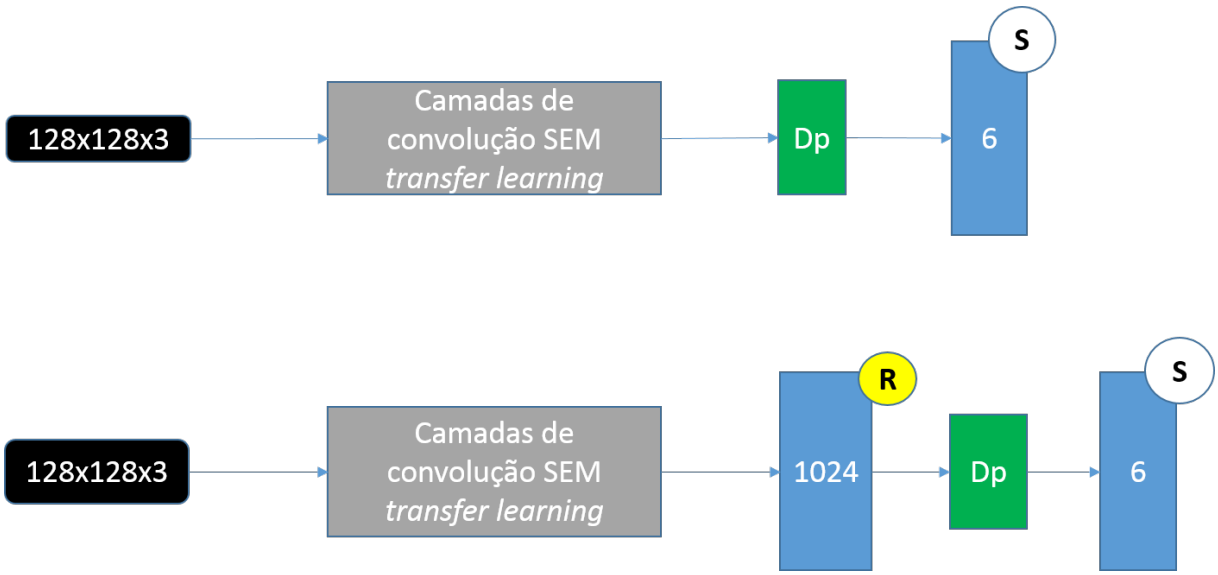
- Na Figura 4.9 é apresentado o treinamento com *transfer learning* mais aumento de dados. Foram analisadas as CNNs Xception, InceptionResNet50V2 e InceptionV3;

Figura 4.4: Legenda de componentes das CNNs.



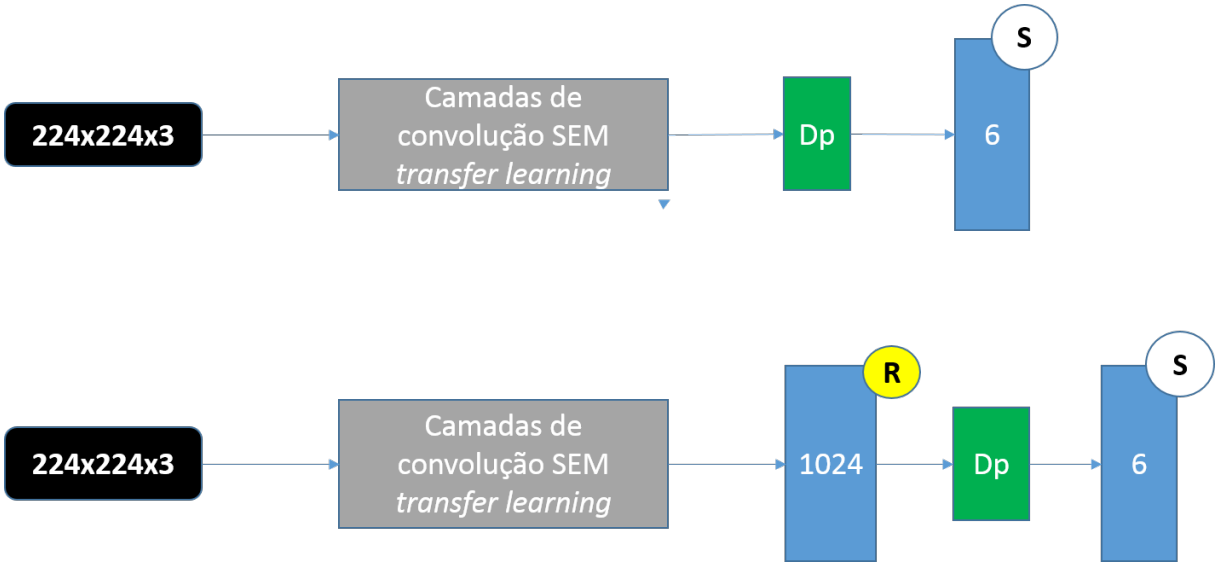
Fonte: autoria própria.

Figura 4.5: Experimento das CNNs treinadas sem *transfer learning* 128x128x3.



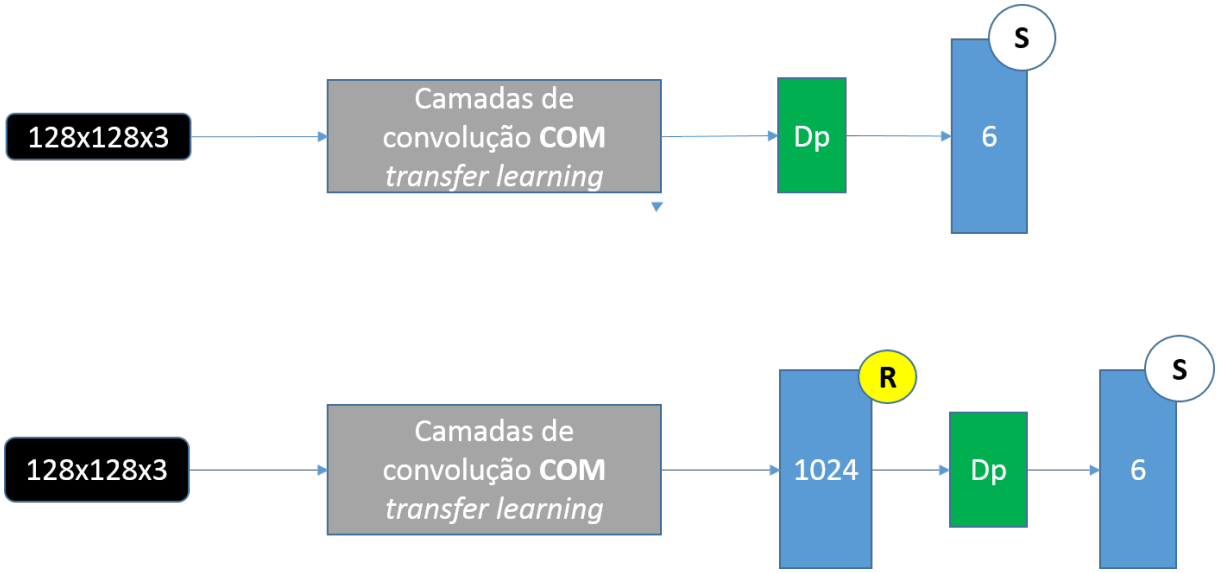
Fonte: autoria própria.

Figura 4.6: Experimento das CNNs treinadas sem *transfer learning* 224x224x3.



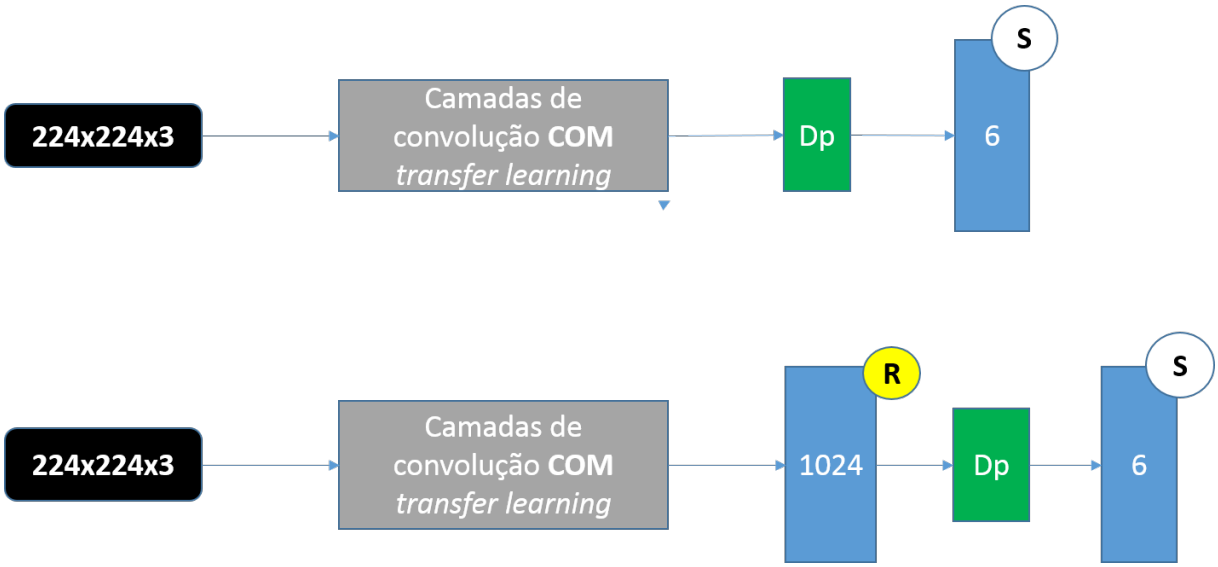
Fonte: autoria própria.

Figura 4.7: Experimento das CNNs treinadas com *transfer learning* 128x128x3.



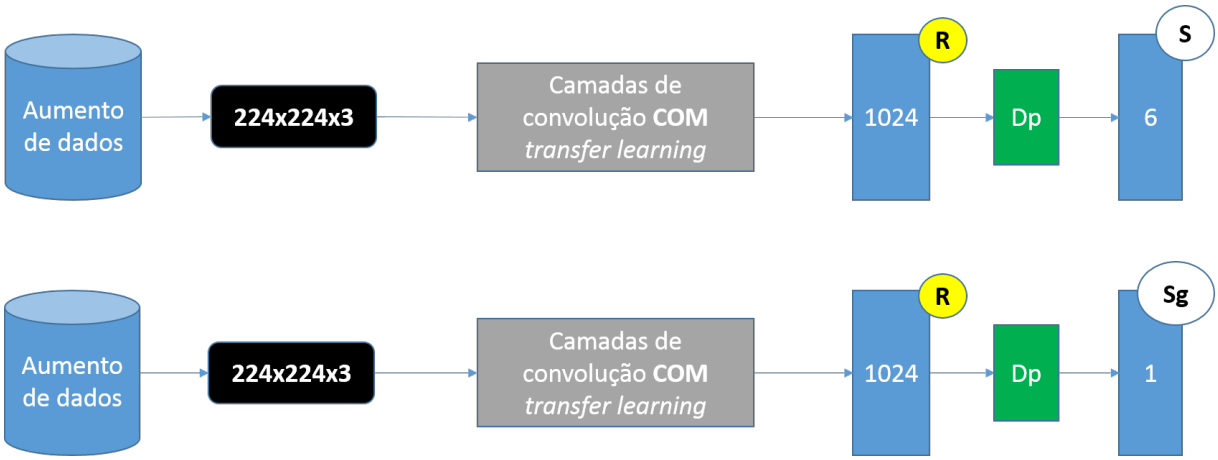
Fonte: autoria própria.

Figura 4.8: Experimento das CNNs treinadas com *transfer learning* 224x224x3.



Fonte: autoria própria.

Figura 4.9: Experimento das CNNs treinadas com *transfer learning* 224x224x3, mais aumento de dados.



Fonte: autoria própria.

CAPÍTULO

5

Resultados e Discussões

Nesta seção são apresentadas as comparações entre os resultados de performances de classificação utilizando a abordagem BoVW em conjunto com os algoritmos de AM k-NN e SVM, disponíveis na biblioteca Scikit-Learn, e as CNNs MobileNetV2, InceptionResNetV2, InceptionV3, ResNet50V2, VGG16 e Xception, todas disponíveis na biblioteca Keras.

A seção está organizada em dois grupos de experimentos:

- **Classificação Multiclasse**
  - Experimentos com o uso de representações BoVW;
  - Experimentos com o uso de CNNs pré-treinadas;
  - Experimentos com o uso de de CNNs sem *transfer learning*;
  - Experimentos com o uso de de CNNs com *transfer learning*;
  - Experimentos com o uso de de CNNs com *transfer learning* e aumento de dados;
  - Comparação entre os resultados da BoVW e os resultados obtidos pelas CNNs.
- **Predador x Não Predador (Classificação Binária)**
  - Experimentos com o uso de representações BoVW;
  - Experimentos com o uso de de CNNs com *transfer learning* e aumento de dados;
  - Comparação entre os resultados da BoVW e os resultados obtidos pelas CNNs.

5.1 Classificação Multiclasse

Nesta seção são apresentados os resultados para classificação multiclasse utilizando representações BoVW em conjunto com algoritmos de aprendizado de máquina, e CNNs. A seguir são apresentados os resultados utilizando as divisões apresentadas na seção anterior.

5.1.1 Experimentos com BoVW

Nesta seção é apresentada a comparação das representações BoVW com e sem a aplicação da técnica de Canny, considerando as diferentes quantidades de palavras visuais, e considerando os algoritmos *k-NN* e SVM. Na Figura 5.1 são apresentados os resultados das variantes da BoVW para os diferentes algoritmos. Analisando o comportamento geral dos dados, a acurácia tende a diminuir quando se aumenta a quantidade de atributos acima de mil atributos. A técnica de Canny aplicada no conjunto de imagens obteve melhor resultado apenas com a quantidade de 500 atributos.

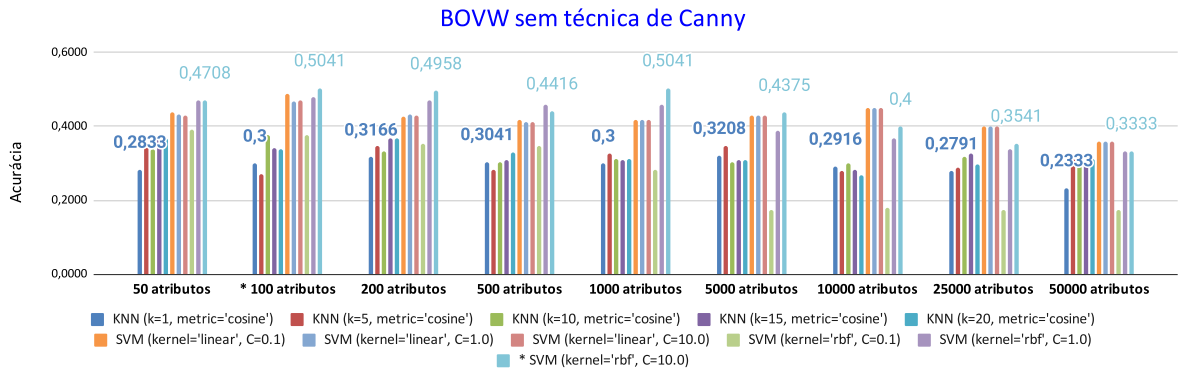
Comparando os resultados gerados com BoVW para todos animais, o melhor foi sem a técnica de Canny, que obteve o resultado de acurácia máxima de 0.5041 para o algoritmo SVM com os parâmetros kernel='rbf' e C=10.0, na quantidade de 100 atributos. A partir dos resultados obtidos neste experimento, infere-se que se perdem informações importantes na imagem quando se aplica a técnica de Canny e a extração acima de 10 mil atributos prejudica a qualidade dos resultados dos algoritmos de aprendizado de máquina.

5.1.2 Experimentos Utilizando Modelos Previamente Treinados

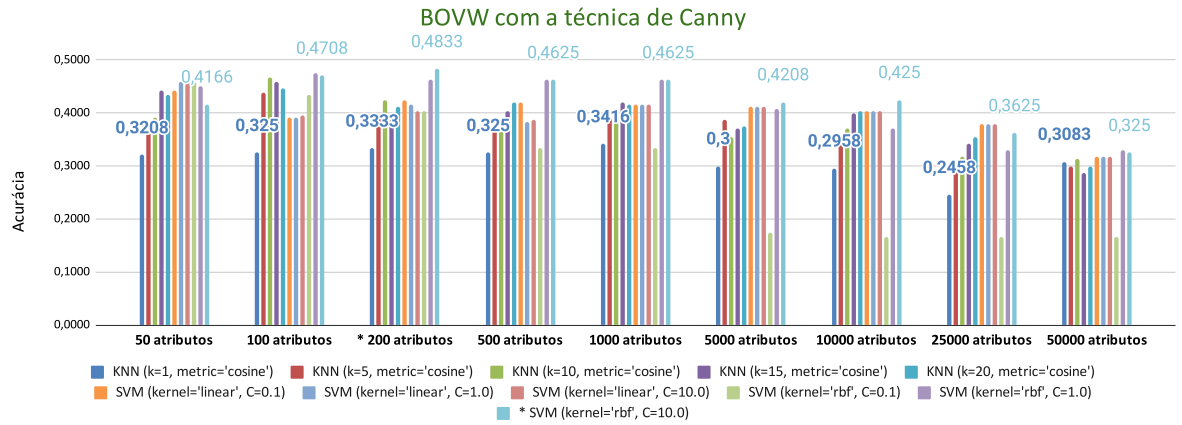
O experimento com uso de modelos treinados consideram o uso integral e sem modificação dos pesos de arquiteturas de CNNs treinadas na base ImageNet. Na Figura 5.2 é apresentado o gráfico com as acurácias de cada rede convolucional com os seus pesos pré-treinados. Porém, vale ressaltar que foi necessário utilizar o tamanho de entrada que cada CNN determina por padrão. As CNNs e seus respectivos tamanhos de entrada são:

- MobileNetV2: entrada igual a 244 X 244 pixels;
- InceptionResNetV2: entrada igual a 299 X 299 pixels;
- InceptionV3: entrada igual a 299 X 299 pixels;
- ResNet50V2: entrada igual a 244 X 244 pixels;
- VGG16: entrada igual a 244 X 244 pixels;

Figura 5.1: Comparação das representações BoVW (a) SEM a utilização do filtro de Canny. e (b) COM a utilização do filtro de Canny.



(a)



(b)

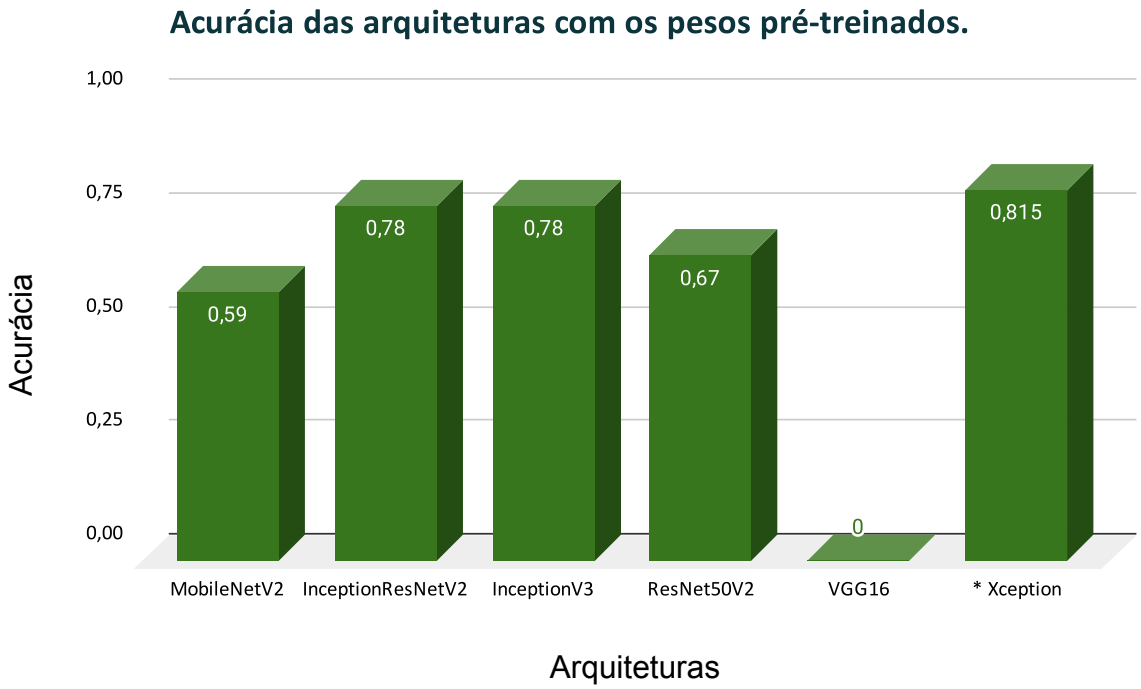
Fonte: autoria própria.

- Xception: entrada igual a 299 X 299 pixels;

Neste experimento, a CNN Xception obteve o melhor resultado com acurácia de 0,815 enquanto as CNNs InceptionResNetV2 e InceptionV3 obtiveram o segundo melhor resultado de 0,78 de acurácia. O pior resultado foi o VGG16 com 0,0 de acurácia. Na Tabela 5.1 é apresentada a matriz de confusão gerada com os resultados obtidos com o modelo Xception.

Conforme a matriz de confusão da Tabela 5.1 o modelo Xception acertou a previsão das 40 imagens de Felino e não errou nenhuma previsão para a classe Felinos, acertou a previsão de 24 imagens de 40 para Ovino, acerto 32 de 40 imagens da classe Boi, acertou 39 da classe Cão, errando somente a previsão de uma imagem.

Figura 5.2: Arquiteturas com pesos pré treinados.



Fonte: autoria própria.

Tabela 5.1: Matriz de Confusão do modelo Xception.

Matriz de Confusão para as classes detectadas no projeto predador					
Classes reais	Classes atribuídas pelo classificador				
	1-Felino	**2-Ovino**	**3-Boi**	**4-Cão**	**5-Outros**
1-Felino	40	0	0	0	0
2-Ovino	0	24	0	9	7
3-Boi	0	0	32	3	5
4-Cão	0	0	0	39	1
5-Outros	0	0	0	0	0

Fonte: autoria própria.

5.1.3 *Experimentos com Treinamento Total da Rede*

O experimento de classificação de modelos treinados sem *transfer learning* considera o treinamento total da rede. Foram feitos experimentos com as CNNs e comparações dos resultados das CNNs sem e com a camada escondida de 1024 neurônios adicionada, considerando a entrada de tamanho 128 x 128 pixels. Também foi feita comparação considerando a entrada de tamanho 244 x 244 pixels.

5.1.4 Experimentos Com e Sem a Camada Escondida Considerando a Entrada de 128 x 128 pixels

Nesta seção são apresentados os resultados obtidos por modelos CNNs considerando pesos iniciais gerados aleatoriamente e com com aprendizado de todos os pesos durante o treinamento, e entrada de dimensões 128 x 128 pixels. Na Figura 5.3 (a) são apresentados os resultados das arquiteturas sem camada escondida, e na Na Figura 5.3 (b) são apresentados os resultados com camada escondida.

O melhor resultado obtido com os modelos sem camada escondida foi o do modelo Xception, que obteve uma média de 0.779 e a maior acurácia com 0.866. Já considerando a camada escondida na arquitetura, o melhor modelo foi a ResNet50V2 que obteve uma média de 0.874, sendo a acurácia máxima de 0.9.

5.1.5 Experimentos Com e Sem a Camada Escondida, considerando uma Entrada de 224 x 224 pixels

Nesta seção são apresentados os resultados obtidos com modelos pré-treinados na Imagenet. Na Figura 5.4 (a) é apresentado um gráfico com os resultados das performances de classificação considerando os modelos sem camada escondida. Dentre os resultados, a melhor média foi obtida pelo modelo InceptionResNetV2 com uma média de 0.847. Já a maior acurácia foi obtida com o modelo Xception com valor de 0.9291.

Na Figura 5.4 (b) são apresentados os resultados são dos modelos utilizando a camada escondida. A maior média foi obtida pelo modelo MobileNetV2 sendo o valor de 0,9075. Já a acurácia máxima também foi obtida pelo modelo MobileNetV2, com 0,9208.

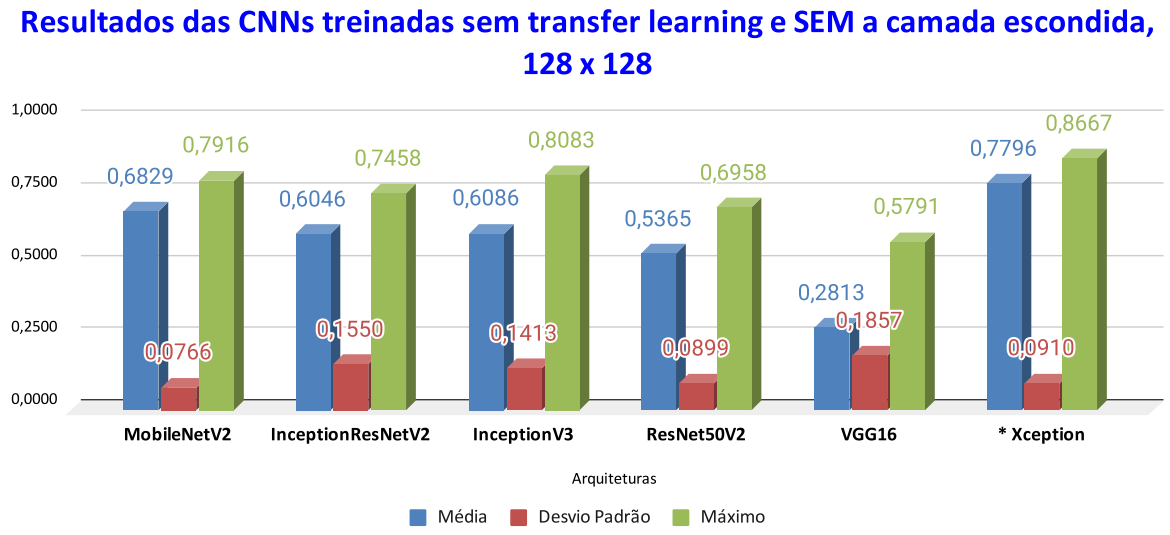
5.1.6 Experimentos com Transfer Learning

Nesta seção são apresentados os resultados obtidos com o uso de *transfer learning* utilizando pelos obtidos com o treinamento na base Imagenet. Vale ressaltar que foram congelados os pesos das camadas convolucionais e foi realizado o treinamento apenas nas camadas densas da rede.

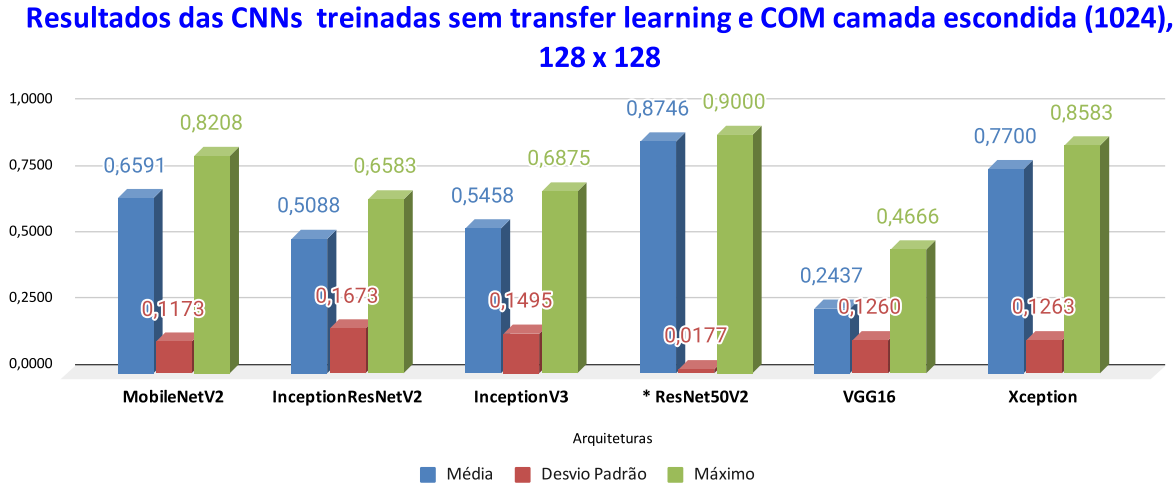
5.1.7 Experimentos com Transfer Learning, Com e Sem Camada Escondida, e Considerando uma Entrada 128 x 128 pixels.

O melhor resultado obtido no experimento com *transfer learning* sem camada escondida foi utilizando o modelo Xception, o qual obteve a média de

Figura 5.3: Performances de classificação considerando os modelos SEM camada escondida considerando o tamanho 128 x 128 pixels (a), e COM camada escondida no tamanho 128 x 128 pixels (b)



(a)

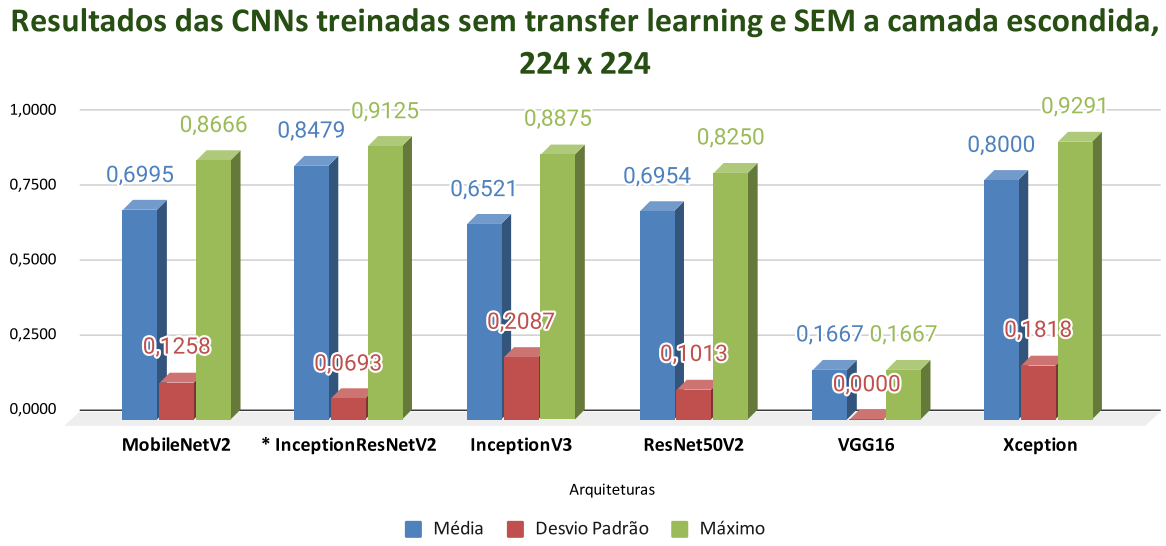


(b)

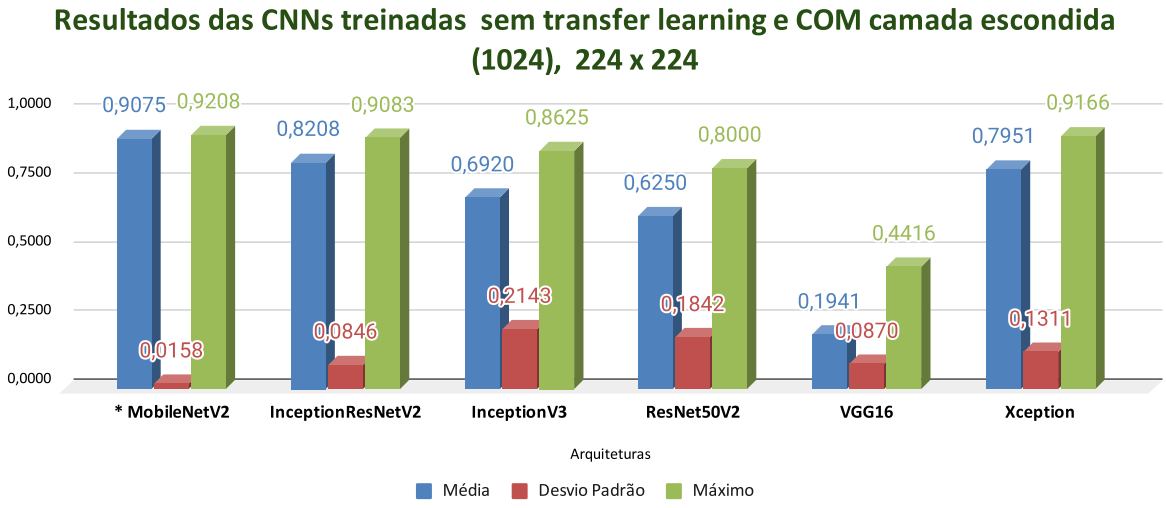
Fonte: autoria própria.

0.7433 (Figura 5.5 (a)). Porém, a acurácia máxima foi obtida com o modelo Inception, sendo o valor de 0,8291. Já nos experimentos utilizando a camada escondida (Figura 5.5 (b)), o melhor resultado, tanto em termos de média quanto valor máximo foi obtido pelo modelo Xception, sendo a acurácia média de 0,8771 e a acurácia máxima de 0,9083.

Figura 5.4: Performances de classificação considerando os modelos SEM camada escondida considerando o tamanho 224 x 224 (a), e COM camada escondida no tamanho 224 x 224 (b)



(a)



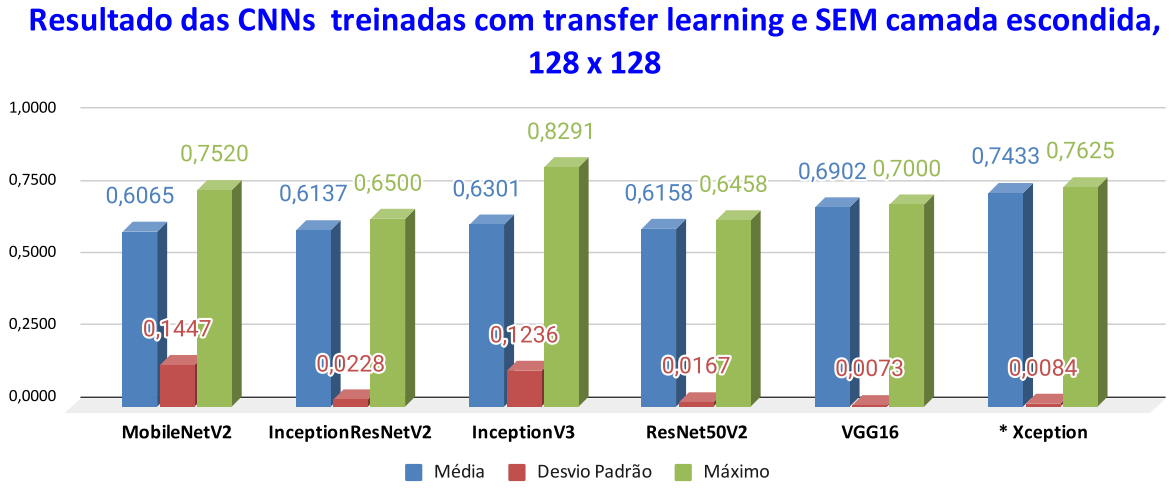
(b)

Fonte: autoria própria.

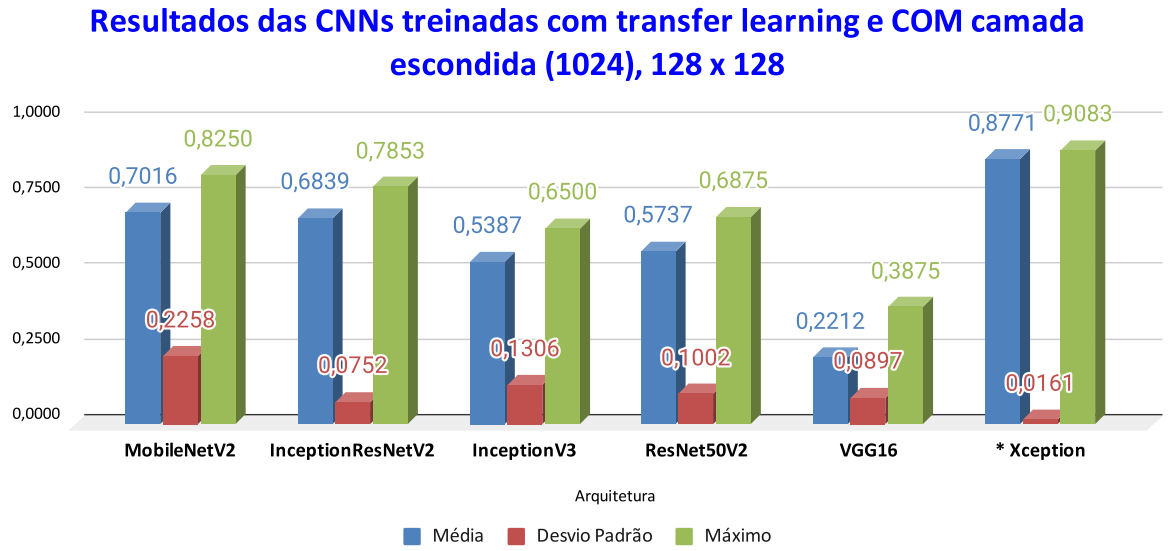
5.1.8 *Experimentos com Transfer Learning, Com e Sem a Camada Escondida, e Considerando uma Entrada de 224 x 224 pixels.*

Neste experimento, o melhor resultado sem a utilização da camada escondida foi obtida pela arquitetura InceptionResNetV2, com média de 0,9033. Porém, a acurácia máxima foi obtida pelo modelo Xception, com o valor de 0,9208. Já o melhor resultado gerado com camada escondida foi o do modelo

Figura 5.5: Performances de classificação considerando *transfer learning* nos modelos SEM camada escondida, considerando o tamanho 128 x 128 pixels (a), e COM camada escondida no tamanho 128 x 128 pixels (b)



(a)



(b)

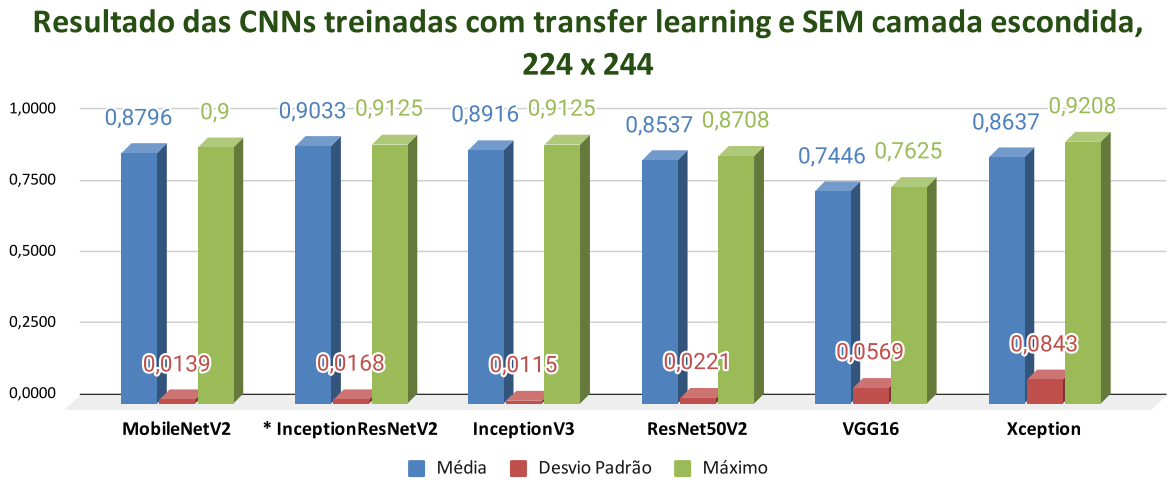
Fonte: autoria própria.

InceptionResNetV2, com média de 0.8955. Contudo, a maior acurácia foi do modelo Xception com 0.9250. Na Figura 5.6 são apresentados os resultados para todos com modelos com e sem a camada escondida.

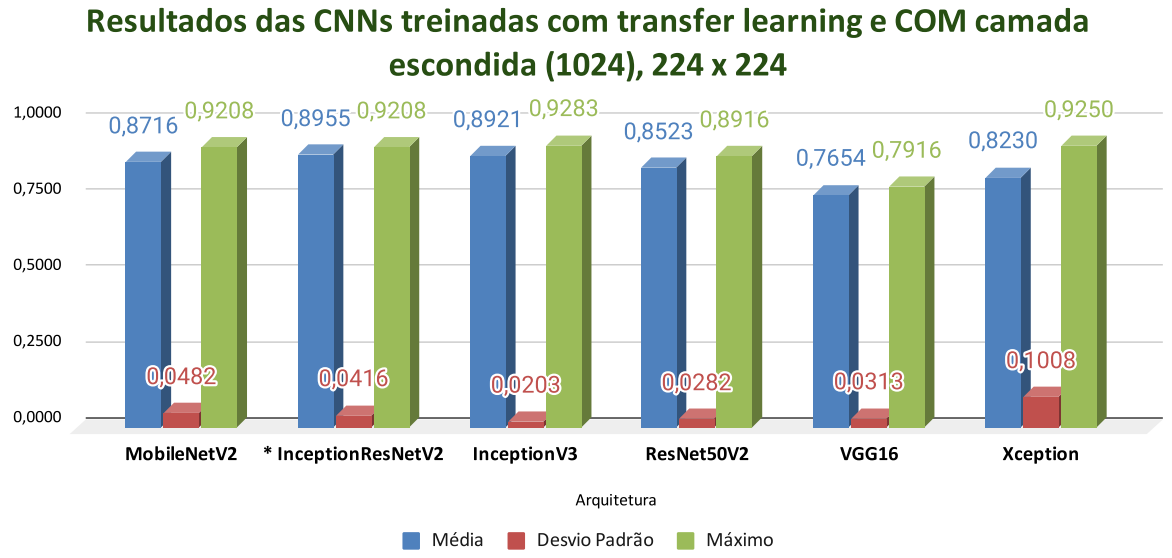
5.1.9 Experimentos com Transfer Learning e Aumento de Dados

Com base nos resultados dos experimentos anteriores, onde os melhores resultados foram obtidos com o uso de *transfer learning* e entrada de 224 x

Figura 5.6: Considerando as convoluções fixa dos modelos nos tamanhos 224 x 224 pixels sem camada escondida (a), e com camada escondida (b)



(a)



(b)

Fonte: autoria própria.

224 pixels, foi aplicado a técnica de aumento de dados visando o aumento da acurácia. Foram utilizadas aqui 3 arquiteturas com os melhores resultados obtidos nas seções anteriores: Xception, InceptionV3 e InceptionResNetV2. Ao aplicar a técnica de aumento de dados, foi obtida uma acurácia média de 0.9329, acurácia máxima de 0,9625, e um desvio padrão de 0.0138. O aumento de dados proporcionou uma melhora no resultados do modelo Xception com a camada escondida, aumentando em 0,1099 acurácia média e 0,0375 a acurácia máxima.

A InceptionV3 obteve a média 0,9183 com desvio padrão 0,0123 chegando

à acurácia máxima de 0,9334. Já o InceptionResNetV2 obteve a média 0,9241 com desvio padrão 0,0151 chegando à acurácia máxima de 0,9458. Portanto, o modelo Xception obteve resultados melhores que os dos modelos InceptionV3 e InceptionResNetV2. Porém, vale ressaltar que para todas as arquiteturas houve um aumento na acurácia média e uma diminuição no desvio padrão.

5.1.10 Comparação Entre os Resultados Obtidos com o Uso de Representações BoVW e CNNs para Classificação Multiclasse

O melhor resultado, dentre os experimentos de Classificação Multiclasse, foi obtido com o uso da CNN Xception, em que se obteve o resultado da acurácia máxima de 0,9625 no experimento *transfer learning* com a camada escondida, entrada de tamanho 224 x 224 pixels, e em conjunto com a técnica de aumento de dados. Esse resultado foi superior ao melhor resultado obtido utilizando uma representação BoVW, que obteve a acurácia máxima de 0,5041.

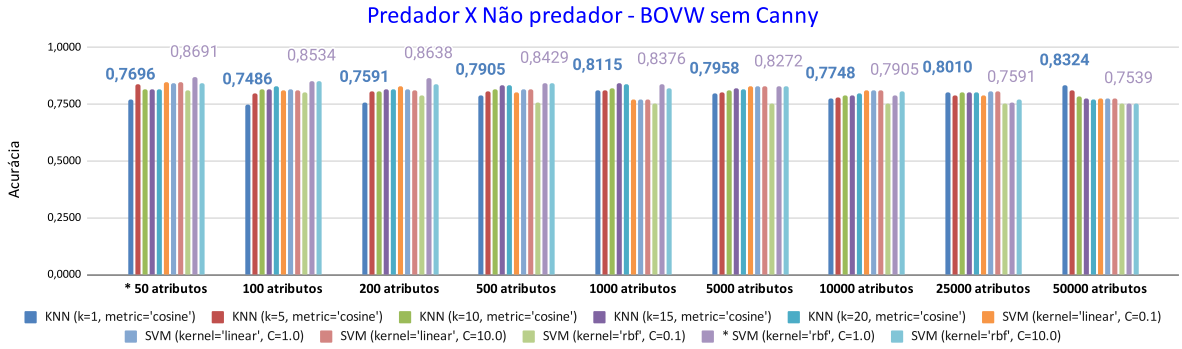
5.2 Classificação Binária: Predador x Não Predador

Baseado nos resultados obtidos nos experimentos multiclasse, foram considerados as CNNs Xception, InceptionResNetV2 e InceptionV3 nos experimentos de classificação binária (Predador x Não Predador). Foram realizados dois experimentos para a análise de performance da classificação: (i) experimento com as CNNs, em conjunto com *transfer learning* e aumento de dados; e (ii) experimento utilizando representações BoVW e algoritmos de aprendizado de máquina.

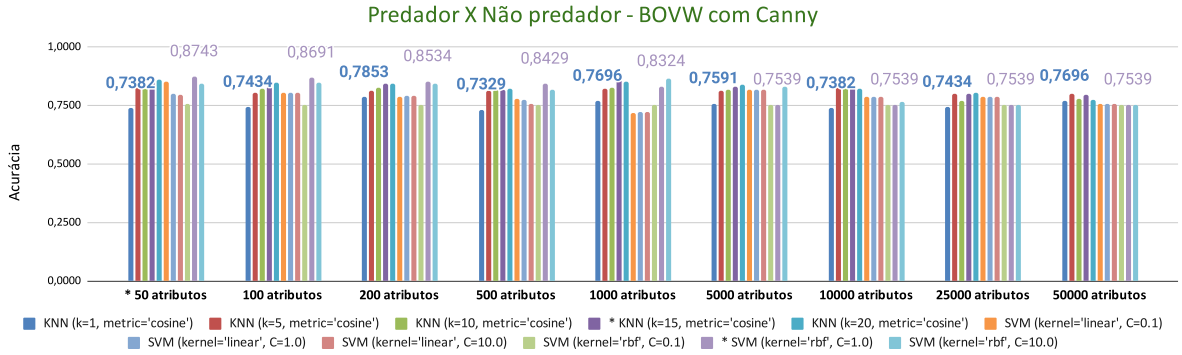
5.2.1 Experimentos com BoVW

Na Figura 5.7 são apresentados os resultados utilizando representações BoVW e algoritmos de aprendizado de máquina. O melhor resultado foi com uma BOVW de 50 atributos extraído do conjunto de imagens com a técnica de Canny, e utilizando o algoritmo SVM com os parâmetros do *kernel*='rbf' e $C = 1.0$. A acurácia obtida para esta combinação foi de 0,8743. Pode-se observar que, em geral, a quantidade de atributos foi inversamente proporcional à performance de classificação.

Figura 5.7: Performances de classificação considerando somente duas classes "Predador e Não predador"(a) com representações BoVW sem a técnica de Canny no conjunto de imagens; e (b) com representações BoVW aplicando técnica de Canny no conjunto de imagens.



(a)



(b)

Fonte: autoria própria.

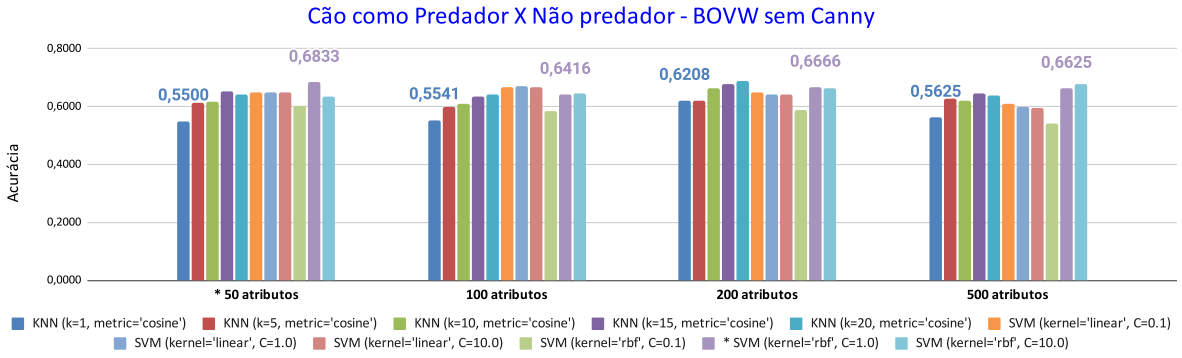
5.2.2 *Experimentos com Representações BoVW e Considerando Cão Como Sendo da Classe Predador*

A diferença deste experimento para o experimento anterior foi que a base de dados foi modificada, retirando as imagens da classe Cão como pertencentes à classe Não Predador e incluindo essas imagens na classe Predador. Isso foi feito pelo motivo que existem casos de cães selvagens que também predam ovinos.

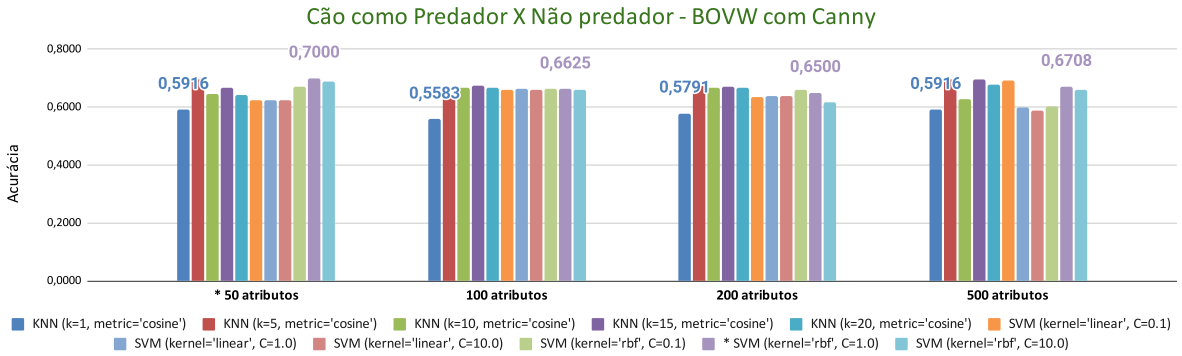
Neste experimento também foram consideradas representações BoVW com 50, 100, 200, e 500 atributos, uma vez que verificou-se que um número maior de atributos não implica no aumento da performance de classificação. Na Figura 5.8 são apresentados os resultados deste experimento.

Considerando os resultados apresentados na Figura 5.8, tanto com quanto sem a utilização da técnica de Canny, o melhor resultado foi obtido com com 50 atributos na representação BOW, uso da técnica de Canny, e com o algo-

Figura 5.8: Performances de classificação considerando a espécie Cão da classes Predador no experimento de "Predador e Não predador", (a) considerando a técnica ORB no conjunto imagem sem a técnica de Canny, comparado com a (b) técnica ORB no conjunto imagem depois de aplicado a técnica de Canny



(a)



(b)

Fonte: autoria própria.

ritmo SVM com parâmetros $C = 1$ e *kernel='rbf'*.

5.2.3 Experimentos com CNNs

Neste experimento foi utilizada a arquitetura Xception em conjunto com as técnicas de *transfer learning* e aumento de dados, uma vez que essa combinação obteve os melhores resultados nos experimentos anteriores. O resultado obtido foi uma média de 0.9926, desvio padrão de 0.036, e um valor de acurácia máxima de 1,0.

Neste mesmo experimento, apenas modificando a base de dados para considerar imagens de cães como sendo da classe Predador, a acurácia média obtida foi de 0,91, com desvio padrão de 0.0259, e a acurácia máxima foi de 0,9416.

5.2.4 Comparação Entre os Resultados Obtidos com BoVW e CNNs

A acurácia máxima obtida pela CNN Xception foi de 1,0, valor máximo possível para essa medida, enquanto que a acurácia máxima obtida utilizando representações BoVW foi 0,8743. Se comparada a acurácia média, a CNN Xception também obteve resultados superiores ao uso de BoVW (0,9926). Portanto, comparando estes dois resultados para classificação binária, o melhor resultado obtido para classificação de Predador x Não Predador foi com o modelo da CNN Xception utilizado *transfer learning* e aumento de imagens.

5.2.5 Discussão dos Resultados

Em geral, a proposta obteve resultados satisfatórios para capturar imagens automaticamente e prever de forma eficiente em ambos contextos, análise multiclasse e binária. No contexto de previsão multiclasse neste projeto, a acurácia máxima obtida entre todos os experimentos foi obtida com o modelo Xception, utilizando o método de Aprendizagem de Máquina mais *transfer learning* com aumento de dados que resultou na acurácia 0,9625. Já no contexto de classificação binária, foi possível obter a acurácia máxima (1,0). Novamente, essa acurácia foi obtida utilizando a arquitetura Xception com as técnicas de *transfer learning* e aumento de dados.

Para fazer uma análise de custo-benefício, isto é, apresentar diferentes abordagens utilizadas que consequentemente possuem diferentes custos computacionais. Para isso, foram considerados os melhores resultados utilizando representações BoVW com algoritmos de aprendizado de máquina, os quais possuem o menor custo computacional, em seguidas CNNs com entradas 128 x 128 pixels, e por fim, CNNs com entradas 224 x 224, e CNNs com entradas 224 x 224 e considerando uma base com aumento de imagens, as quais possuem o maior custo computacional dos experimentos. Vale ressaltar também que são apresentados os custos com e sem o uso de *transfer learning*, o que também impacta o custo computacional uma vez que com o uso de *transfer learning* não foi preciso aprender os pesos das camadas de convolução.

Nas Figuras 5.9 e 5.10 são apresentados os melhores resultados para classificação multiclasse e binária, respectivamente, desta análise de custo-benefício. Pode-se analisar pela 5.9 que a abordagem de menor custo computacional (BoVW) não apresenta performance de classificação satisfatória (0,5). Já ao utilizar CNNs, sem considerar *transfer learning*, a variação da entrada de 128 x 128 para 224 x 224 foi capaz de aumentar a acurácia média em 0,04 e a a acurácia máxima em 0,02. Já ao utilizar *transfer learning*, a variação da acurácia média ao aumentar as dimensões da entrada foi de 0,02

Figura 5.9: Os melhores resultados em relação a custo-benefício para classificação multiclasse.

Classificação Multiclasse

BoVW	Sem Canny	Com Canny
Classificação (AM)	SVM	SVM
Quantidade de Atributos	100	100
Acurácia	0,50	0,48

Treinamento **sem** *transfer learning* 128x128	Camada escondida	Acurácia **média**	Acurácia **máxima**	Desvio padrão
Xception	Sem	0,78	0,87	0,09
Resnet50V2	Com	0,87	0,90	0,02

Treinamento **com** *transfer learning* 128x128	Camada escondida	Acurácia **média**	Acurácia **máxima**	Desvio padrão
Xception	Sem	0,74	0,76	0,01
Xception	Com	0,88	0,91	0,02

Treinamento **sem** *transfer learning* 224x224	Camada escondida	Acurácia **média**	Acurácia **máxima**	Desvio padrão
InceptionResnetV2	Sem	0,85	0,91	0,07
MobiletV2	Com	0,91	0,92	0,02

Treinamento **com** *transfer learning* 224x224	Camada escondida	Acurácia **média**	Acurácia **máxima**	Desvio padrão
InceptionResnetV2	Sem	0,90	0,92	0,02
InceptionResnetV2	Com	0,90	0,92	0,04

Treinamento **com** *transfer learning* + **aumento de dados** 224x224	Camada escondida	Acurácia **média**	Acurácia **máxima**	Desvio padrão
Xception	Com	0,93	0,96	0,01
InceptionResnetV2	Com	0,92	0,94	0,01
InceptionV3	Com	0,91	0,93	0,01

Fonte: autoria própria.

Figura 5.10: Os melhores resultados em relação a custo e benefício para classificação binária.

Classificação binária

Cão não predador		
BoVW	Sem Canny	Com Canny
Classificação (AM)	SVM	SVM
Quantidade de Atributos	50	50
Acurácia	0,8691	0,8743

Cão como predador		
BoVW	Sem Canny	Com Canny
Classificação (AM)	SVM	SVM
Quantidade de Atributos	50	50
Acurácia	0,6833	0,7000

Treinamento com transfer learning + aumento de dados, com tamanho 224x224

CNN	Cão	Acurácia **média**	Acurácia **máxima**	Desvio padão
Xception	Não predador	0,9926	1,0000	0,0037
Xception	predador	0,9100	0,9416	0,0259

Fonte: autoria própria.

na acurácia média e 0,01 na acurácia máxima. Portanto, o uso de *transfer learning* diminui a variação dos resultados ao aumentar a entrada. Isso pode

ser interessante em situações práticas, pois nesta situação, além da entrada menor, apenas a parte densa da rede precisa ser treinada. Por fim, ao utilizar o aumento de dados, *transfer learning* e entrada de 224 x 224 pixels, pode-se notar um aumento na acurácia média de 0,03 e acurácia máxima de 0,04.

Já ao analisar o custo benefício da classificação binária (Figura 5.10), e considerando Cão como sendo da classe Predador, observa-se um aumento na performance de classificação de 0,12 ao considerar uma CNN Xception com *transfer learning* e aumento de dados em relação ao uso de representações BoVW. Já ao considerar Cão como sendo da classe Predador, a aumento na performance ao utilizar a CNN Xception ao invés da BoVW é de 0,21. Portanto, na classificação binária considerando como Não Predador, a representação BoVW apresenta um custo-benefício satisfatório. Porém, ao considerar cão como sendo da classe predador, o decréscimo na acurácia é de 0,17, tornando essa abordagem menos vantajosa para o uso em situações práticas. Por fim, novamente, a abordagem utilizando CNN Xception, *transfer learning*, aumento de dados e entrada de 224 x 224 pixels, possui custo computacional maior, porém, apresentou as maiores performances de classificação.

CAPÍTULO

6

Conclusões

Neste projeto de mestrado foi desenvolvido um sistema de alertas a ataques de predadores a rebanhos via reconhecimento automático de predadores em imagens. O sistema possui uma fase de captura e detecção de movimento ou surgimento de novos objetos nas imagens. Ao realizar tal detecção, as imagens são submetidas à técnicas de visão computacional, as quais irão reportar se uma imagem contém ou não um predador, ou ainda, qual a espécie do animal detectado na imagem.

Além da arquitetura do sistema, neste projeto também foram apontados quais abordagens apresentam os melhores resultados para o reconhecimento de predadores ou espécies de animais nas imagens, além de uma análise de custo-benefício das abordagens. Para a classificação binária, isto é, Predador e Não Predador, os modelos CNNs usando o *transfer learning*, com camada escondida e entrada de tamanho 224 x 224 pixels, tiveram melhor desempenho que o uso de representações BoVW e algoritmos de aprendizado de máquina. Destaca-se aqui a abordagem Xception com o uso das técnicas de *transfer learning* e aumento de dados, a qual obteve uma acurácia média de 0,99 e acurácia máxima de 1,0 (valor máximo para essa medida), obtendo portanto um desempenho satisfatório.

Para a classificação multiclasse considerando 6 diferentes espécies de animais como categorias, as arquiteturas CNNs em conjunto com as técnicas de *transfer learning* e aumento de dados obtiveram as maiores acurácias além de superarem os resultados obtidos utilizando representações BoVW. Vale ressaltar que as arquiteturas com melhores performances foram, respectivamente, Xception, InceptionResNetV2 e InceptionV3. Por exemplo, a arquitetura Xception em conjunto com as técnicas mencionadas anteriormente obteve uma

acurácia média de 0,9329, com desvio padrão de 0,0138, e acurácia máxima de 0,9625. Portanto, essas abordagens são indicadas para uso de identificação da espécie predadora, atendendo não somente o escopo do projeto, mas podendo ser utilizado para fins ambientais, como o censo e preservação de espécies selvagens.

Neste projeto buscou-se criar um sistema inicial que possibilitasse ações para evitar a perda dos animais domésticos. Como trabalhos futuros serão feitos testes de alcance, isto é, qual a distância máxima em que é possível fazer o reconhecimento do predador na imagem além de testar os melhores formas de comunicação dos dados, isto é, testar a capacidade de transmissão dos dados para analisar se é possível a transmissão das imagens para o seu reconhecimento em um servidor central, ou se é necessária a classificação da imagem no próprio dispositivo de captura e transmissão apenas da informação da categoria detectada.

Referências Bibliográficas

AFKHAM, H. et al. (2008). **Joint visual vocabulary for animal classification**. In *Proceedings of the International Conference on Pattern Recognition*, pginas 1–4. IEEE. Citado na página 28.

AGGARWAL, C. (2015). ***Data Classification: Algorithms and Applications***. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press. Citado na página 25.

AGGARWAL, C. (2018). ***Neural Networks and Deep Learning:*** *A Textbook*. Springer International Publishing. Citado na página 17.

AKYILDIZ, I. e WANG, X. (2009). ***Wireless Mesh Networks***. Advanced Texts in Communications and Networking. Wiley. Citado na página 35.

ANTONIO, W. H. S. (2019). **Detecção de Animais em Rodovias utilizando Câmeras e Aprendizado Supervisionado**. Citado nas páginas 28 e 32.

ARAÚJO, Flávio H. D. et al (2017). **Redes Neurais Convolucionais com Tensorflow:** Teoria e Prtica. *Sociedade Brasileira de Computação. III Escola Regional de Informática do Piauí. Livro Anais-Artigos e Minicursos*, 1:382–406. Citado na página 17.

ARRUDA, M. S. (2018). **Identificação de Espécies de Animais do Pantanal usando Imagens Térmicas e Redes Neurais Convolucionais**. Citado nas páginas 30 e 32.

BORTH, M. R. et al. (2013). **Análise da Extração da Extração de Atributos do Algoritmo SURF em Espécies de Peixes**. Citado nas páginas 30 e 32.

CARUANA, R. e NICULESCU-MIZIL, A. (2006). **An empirical comparison of supervised learning algorithms**. In *Proceedings of the International Conference on Machine Learning*, pginas 161–168. ACM. Citado na página 42.

CATRO, L. N. e FERRARI, D. G. (2016). ***Introdução a mineração de dados:*** *conceitos básicos, algoritmos e aplicações*. Saraiva. Citado nas páginas 7 e 24.

CHOLLET, F. (2016). **Xception: Deep Learning with Depthwise Separable Convolutions**. *CoRR*, abs/1610.02357. Citado na página 22.

CONGALTON, R. e GREEN, K. (1999). **Basic analysis techniques**. *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*, pginas 47–68. Citado na página 23.

CROSTA, A. P. (1999). ***Processamento digital de imagens de sensoriamento remoto***. UNICAMP/Instituto de Geociências. Citado na página 7.

DAVIES, E. (2017). ***Computer Vision:*** *Principles, Algorithms, Applications, Learning*. Elsevier Science. Citado nas páginas 5 e 26.

DEIS, A. (2019). **Data Augmentation for Deep Learning**. Disponível em: `https://towardsdatascience.com/data-augmentation-for-deep-learning-4fe21d1a4eb9`. Acesso em: 17 abril. 2020. Citado na página 25.

FREITAS, R. P. da S. (2016). **Sistema para detecção de bordas em exames clínicos**. Citado na página 5.

GRANDIS, K. (2012). **PyCon 2012: Militarizing Your Backyard:** computer vision and the squirrel hordes. Citado na página 12.

HAYKIN, S. (1999). ***Neural Networks:*** *A Comprehensive Foundation*. Prentice Hall. Citado na página 16.

HE, K. et al. (2016). **Deep residual learning for image recognition**. In *Proceedings of the International IEEE Conference on Computer Vision and Pattern Recognition*, pginas 770–778. Citado na página 20.

HOFMANN, T. (1999). **Probabilistic Latent Semantic Analysis**. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, pginas 289–296. Morgan Kaufmann Publishers Inc. Citado na página 13.

HOWARD, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., e Adam, H. (2017). **MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications**. *CoRR*, abs/1704.04861. Citado na página 22.

KARIM, R. (2019). **Illustrated: 10 cnn architectures**. Citado nas páginas 18, 19, 20, 21, 22, e 23.

KRIZHEVSKY, A., SUTSKEVER, I., e HINTON, G. E. (2012). **Imagenet classification with deep convolutional neural networks**. In *Proceedings of the International of advances in neural information processing systems*, pginas 1097–1105. Citado na página 18.

LECUN, Y. et al. (1998). **Gradient-based learning applied to document recognition**. *Proceedings of the IEEE*, 86(11):2278–2324. Citado na página 17.

LIRA, S. A. (2004). **Análise de correlação: abordagem teórica e de construção dos coeficientes com aplicações**. *Setores de Ciências Exatas e de*, pagina 209. Citado na página 27.

MARCHINI, S. et al. (2011). **Predadores silvestres e animais domésticos:** guia prático de convivência. *Brasília: Instituto Chico Mendes de Conservação da Biodiversidade*. Citado na página 1.

MARENGONI, M. e Stringhini, S. (2009). **Tutorial: Introdução à visão computacional usando opencv**. *Revista de Informática Teórica e Aplicada*, 16(1):125–160. Citado na página 5.

MENEZES, L. F. J. (2010). **Processamento de imagens digitais para visão robótica**. *INTELLECTUS. Revista do Grupo Polis Educacional*, pagina 39. Citado na página 6.

NOOR, A. et al. (2020). **Automated sheep facial expression classification using deep transfer learning**. *Computers and Electronics in Agriculture*, 175:105528. Citado nas páginas 29 e 32.

NOROUZZADEH, M. S. et al. (2018). **Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning**. *Proceedings of the National Academy of Sciences*, 115(25):E5716–E5725. Citado nas páginas 29 e 32.

NOVAIS, J. P. (2016). **Aplicação dos Algoritmos SIFT e SURF na Classificação de Sub-Imagens por Discriminação de Textura**. Citado na página 8.

NÓBREGA, A. (2018). **Novo Censo Agropecuário mostra crescimento de efetivo de caprinos e ovinos no Nordeste**. Citado na página 1.

OLIVEIRA, G. d. e PRATI, R. C. (2013). **Ajuste de parâmetros em algoritmos de aprendizado de máquina utilizando transferência de aprendizado**. *X Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*. Citado na página 25.

OLIVEIRA, K. F. S. (2019). **Conhecendo o Jinja2:** um mecanismo para templates no flask. Disponível em: https://imasters.com.br/desenvolvimento/conhecendo-o-jinja2-um-mecanismo-para-templates-no-flask. Acesso em: 19 abril. 2020. Citado na página 36.

SANTANA, B. A. S. et al. (2015). **Análise de desempenho de algoritmos detectores de keypoints para um sistema de navegação visual de robôs baseados em smartphones**. Citado na página 9.

SHORTEN, C. e KHOSHGOFTAAR, T. M. (2019). **A survey on image data augmentation for deep learning**. *Journal of Big Data*, 6(1):60. Citado nas páginas 24 e 25.

SILVA, L. C. F. D. (2017). **Modelo de Transferência de Aprendizagem baseado em Regressão Linear Regularizada**. Citado na página 25.

SILVA, F. A. et al. (2013). **Evaluation of Keypoint Detectors and Descriptors**. *Workshop de visão computacional - WVC*. Citado na página 8.

SILVA, J. F. C. et al. (2004). **Avaliação da qualidade da detecção de bordas em uma sequência de imagens de ruas e rodovias**. *Revista Brasileira de Cartografia*, 56(2). Citado na página 8.

SIMONYAN, K. e ZISSERMAN, A. (2014). **Very deep convolutional networks for large-scale image recognition**. *arXiv preprint arXiv:1409.1556*. Citado na página 18.

SIMÕES, A. d. S. e COSTA, A. H. R. (2007). **Aprendizado não-supervisionado em redes neurais pulsadas de base radial: um estudo da capacidade de agrupamento para a classificação de pixels**. *SBA: Controle & Automação Sociedade Brasileira de Automatica*, 18(2):251–264. Citado na página 23.

SIVIC, J. e ZISSERMAN, A. (2003). **Video Google:** a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pginas 1470–1477. IEEE. Citado nas páginas 31 e 32.

SZEGEDY, C., Ioffe, S., Vanhoucke, V., e Alemi, A. A. (2017). **Inception-v4, inception-resnet and the impact of residual connections on learning**. In *Proceedings of the International Thirty-first AAAI Conference on Artificial Intelligence*, pagina 4278–4284. Citado nas páginas 19 e 22.

SZEGEDY, C. e. a. (2016). **Rethinking the inception architecture for computer vision**. In *Proceedings of the International IEEE Conference on Computer Vision and Pattern Recognition*, pginas 2818–2826. Citado na página 20.

SZEGEDY, C. et al. (2015). **Going deeper with convolutions**. In *Proceedings of the International of the IEEE conference on computer vision and pattern recognition*, pginas 1–9. Citado na página 19.

SZELISKI, R. (2010). ***Computer Vision**: Algorithms and Applications*. Springer London. Citado nas páginas 5 e 10.

TAN, P. N., STEINBACH, M., e KUMAR, V. (2005). ***Introduction to Data Mining-Pearson***. Citado nas páginas 13, 14, 16, e 26.

VAPNIK, V. (1998). ***Statistical learning theory***. Adaptive and learning systems for signal processing, communications, and control. Wiley. Citado na página 16.

VIDOLIN, G. P. et al (2004). **Avaliação da predação a animais domésticos por felinos de grande porte no Estado do Paraná:** implicações e estratégias conservacionistas. *Cad. Biodivers*, 2:50–58. Citado nas páginas 1 e 2.

XU, S., FANG, T., LI, D., e WANG, S. (2009). **Object classification of aerial images with bag-of-visual words**. *IEEE Geoscience and Remote Sensing Letters*, 7(2):366–370. Citado na página 13.

YANG, Y. e NEWSAM, S. (2010). **Bag-of-visual-words and Spatial Extensions for Land-use Classification**. In *Proceedings of the International Conference on Advances in Geographic Information Systems*, pginas 270–279. Citado na página 13.

YANG, J. et al. (2007). **Evaluating bag-of-visual-words representations in scene classification**. In *Proceedings of the International Workshop on multimedia information retrieval*, pginas 197–206. Citado na página 15.

YOSINSKI, J et al. (2014). **How transferable are features in deep neural networks?** In *Proceedings of the International Advances in neural information processing systems*, pginas 3320–3328. Citado na página 25.