

**MINISTÉRIO DA DEFESA  
EXÉRCITO BRASILEIRO  
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA  
INSTITUTO MILITAR DE ENGENHARIA  
PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS E COMPUTAÇÃO**

**ALEXANDRA MIGUEL RAIBOLT DA SILVA**

**DESCNET — REDE NEURAL CONVOLUCIONAL DE DESCRITORES: UMA  
ABORDAGEM PARA O FECHAMENTO DE *LOOP* EM *VISUAL SLAM***

**RIO DE JANEIRO  
2021**

**ALEXANDRA MIGUEL RAIBOLT DA SILVA**

**DESCNET — REDE NEURAL CONVOLUCIONAL DE  
DESCRITORES: UMA ABORDAGEM PARA O FECHAMENTO DE  
*LOOP* EM *VISUAL SLAM***

Dissertação apresentada ao Programa de Pós-graduação em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciência em Sistemas e Computação.

Orientador: Paulo Fernando Ferreira Rosa - Ph.D.

Coorientador: Alberto Torres Angonese - D.Sc.

Rio de Janeiro

2021

©2021

INSTITUTO MILITAR DE ENGENHARIA

Praça General Tibúrcio, 80 – Praia Vermelha

Rio de Janeiro – RJ CEP: 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmар ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

da Silva, Alexandra Miguel Raibolt

DescNet — Rede Neural Convolutacional de Descritores: uma abordagem para o fechamento de *loop* em *Visual SLAM* / Alexandra Miguel Raibolt da Silva. – Rio de Janeiro, 2021.

130 f.

Orientador: Paulo Fernando Ferreira Rosa.

Coorientador: Alberto Torres Angonese.

Dissertação (mestrado) – Instituto Militar de Engenharia, Sistemas e Computação, 2021.

1. Detecção e Descrição de Características. 2. Saco de Características Visuais. 3. Perceptron Multicamadas. 4. Rede Neural Convolutacional. 5. Rede Neural Convolutacional de Descritores. 6. Detecção de Fechamento de *Loop*. 7. *Visual SLAM*. 8. Jetson Nano. I. Fernando Ferreira Rosa, Paulo, orient. II. Torres Angonese, Alberto, coorient. III. Título

**ALEXANDRA MIGUEL RAIBOLT DA SILVA**

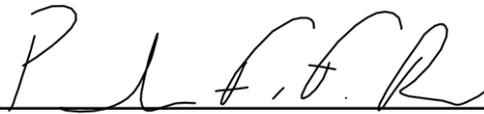
**DescNet — Rede Neural Convolutacional de Descritores:  
uma abordagem para o fechamento de *loop* em *Visual  
SLAM***

Dissertação apresentada ao Programa de Pós-graduação em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciência em Sistemas e Computação.

Orientador: Paulo Fernando Ferreira Rosa

Coorientador: Alberto Torres Angonese

Aprovado em Rio de Janeiro, 2 de dezembro de 2021, pela seguinte banca examinadora:



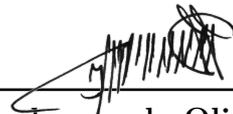
---

Prof. **Paulo Fernando Ferreira Rosa** - Ph.D. do IME - Presidente



---

Prof. **Alberto Torres Angonese** - D.Sc. da FAETERJ-Petrópolis



---

Prof. **Jauvane Cavalcante de Oliveira** - Ph.D do LNCC



---

Prof. **Hebert Azevedo Sá** - Ph.D do IME



---

Prof. **Luiz Satoru Ochi** - D.Sc. da UFF

Rio de Janeiro  
2021

*Dedico esta ao meu companheiro, Diego.*

# AGRADECIMENTOS

Em Eclesiastes 3:1, é dito que: “*Tudo tem o seu tempo determinado, e há tempo para todo o propósito debaixo do céu*”. Portanto, agradeço a Deus, pelo tempo que me foi dado, permitindo-me chegar onde estou. O tempo é valioso, e pude usá-lo de acordo neste ciclo. Entretanto, não sozinha, sendo necessário fazer alguns agradecimentos. Primeiramente aos meus queridos pais, Fátima Miguel e Alexandre Raibolt, e ao meu companheiro, Diego Gonzalez, pelo amor, pelo incentivo e pelo apoio constante, além da compreensão ao suportar centenas de horas de ausência em virtude da dedicação a este trabalho. Agradeço a Marcelo Braun (in memoriam), meu tio, cuja lembrança é de carinho e ensinamentos, sei que onde estiver, estará orgulhoso de mim.

Ao Prof. Paulo Rosa, por quem tive a honra de ter sido orientada durante o mestrado; agradeço pelos ensinamentos, incentivos, confiança depositada e discussões ao longo deste ciclo. Para mim, você é uma inspiração do que verdadeiramente significa ser professor; a mais nobre das artes, um dom, uma missão diária, é deixar um legado. Ao Prof. Alberto Angonese, meu coorientador, por me auxiliar durante o ciclo da graduação, pelo incentivo à realização do mestrado, pelos ensinamentos e discussões ao longo deste ciclo, e pela amizade. Desejo agradecer ainda a todas as pessoas que contribuíram com o desenvolvimento desta Dissertação de mestrado, tenha sido por meio de críticas, ideias, apoio, incentivo ou qualquer outra forma de auxílio. Aos colegas dos Programas de Pós-graduação em Sistemas e Computação (PPgSC) e em Engenharia de Defesa (PPgED), além dos colegas do Laboratório de Robótica e Inteligência Computacional (RoboLAB). Desejo agradecer em especial, às pessoas citadas a seguir, Alexandre Boente, Ana Lúcia Barreto, Antonio Doneda, Cláudio Vasconcelos, Gustavo Casqueiro, Patrícia do Amorim, Taise Grazielle, Thiago Baldivieso, e Viviane Sofiste.

Aos colegas da Federação das Indústrias do Estado do Rio de Janeiro (Firjan) e da Fu2re Smart Solutions, que me proporcionaram apoio e dias de tranquilidade necessários para a dedicação aos estudos e finalização deste trabalho. Aos funcionários, direção, administração e todo corpo docente do Programa de Pós-graduação em Sistemas e Computação (PPgSC) e à Seção de Engenharia da Computação (SE/9) do Instituto Militar de Engenharia (IME). E por fim, à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pela bolsa de mestrado do Programa de Demanda Social (DS).

Hoje chega o tempo de desfrutar das recompensas do meu trabalho, e isto, é um dom de Deus.

*Alexandra Miguel Raibolt da Silva*

*“Anti-intellectualism has been a constant thread winding its way through our political and cultural life, nurtured by the false notion that democracy means that “my ignorance is just as good as your knowledge.””*  
*(Isaac Asimov)*

# RESUMO

A presente Dissertação de mestrado apresenta uma nova abordagem para explorar filtros convolucionais esparsos e binários em Redes Neurais Convolucionais (em inglês, *Convolutional Neural Network* — CNN) tradicionais, denominada — Rede Neural Convolucional de Descritores (em inglês, *Descriptor Convolutional Neural Network* — DescNet). Recentes avanços impulsionados pela crescente demanda para integração de aplicações de Aprendizado Profundo (em inglês, *Deep Learning* — DL) em particular na Robótica Móvel, têm motivado diversas pesquisas para a superação dos desafios relacionados às limitações de recursos computacionais. Um dos maiores desafios da área, que motiva o presente estudo, é o desenvolvimento de aplicações complexas integradas a sistemas de Localização e Mapeamento Simultâneos (em inglês, *Simultaneous Localization and Mapping* — SLAM) de plataformas robóticas móveis. Para tal aplicação, se faz necessário o uso de poder computacional exaustivo. Portanto, economizar recursos provenientes de modelos de Redes Neurais Convolucionais, potencializa a integração e uso desta, e de outras ferramentas integradas. Assim, propomos a reformulação das camadas convolucionais por meio de Descritores Binários Locais (em inglês, *Local Binary Descriptors* — LBD). Com isto, apresentamos duas novas camadas para arquiteturas profundas: (i) Detecção de Características Locais (em inglês, *Local Feature Detection* — LFD), e (ii) Convolução de Descritor Local (em inglês, *Local Descriptor Convolution* — LDC), como uma alternativa viável e eficiente às camadas convolucionais. Descritores Binários Locais inspiram os fundamentos do *design* destas duas camadas. A camada de Detecção de Características Locais é responsável por produzir o processo de Detecção e Descrição de Características por meio do Descritor Binário Local escolhido. Já a camada de Convolução de Descritor Local consiste em um conjunto de filtros convolucionais esparsos e binários reformulados por meio do Descritor Binário Local, seguidos de uma função de ativação não linear. Implementando uma ResNet-152 com blocos DescNet, alcançamos bons resultados sobre a camada convolucional tradicional em conjuntos de dados visuais competitivos (MNIST e CIFAR-10), enquanto permite economia no número de parâmetros aprendíveis do modelo e, conseqüentemente, economia computacional significativa, tornando a DescNet um modelo aplicável em ambientes reais, onde há recursos escassos e limitados. Apresentamos ainda, como proposta de trabalho futuro, a integração de um modelo híbrido de arquitetura de Rede Neural Artificial (em inglês, *Artificial Neural Networks* — ANN) — a Rede Convolucional Recorrente de Longo Prazo (em inglês, *Long-term Recurrent Convolutional Networks* — LRCN) com blocos DescNet com um sistema *Visual SLAM* — VSLAM, capaz de solucionar o problema de Detecção de Fechamento de *Loop*. Um estudo teórico e experimentos apoiam a abordagem proposta.

**Palavras-chave:** Detecção e Descrição de Características. Saco de Características Visuais. Perceptron Multicamadas. Rede Neural Convolucional. Rede Neural Convolucional de Descritores. Detecção de Fechamento de *Loop*. *Visual SLAM*. Jetson Nano.

# ABSTRACT

This Master's Thesis presents a new approach to explore sparse and binary convolutional filters in traditional Convolutional Neural Networks (CNN), called — Descriptor Convolutional Neural Network (DescNet). Recent advances driven by the growing demand for the integration of Deep Learning (DL) applications, particularly in Mobile Robotics, has motivated several researches to overcome the challenges related to the limitations of computational resources. One of the biggest challenges in the area, which motivates this study, is the development of complex applications integrated to Simultaneous Localization and Mapping (SLAM) systems of mobile robotic platforms. For such an application, it is necessary to use exhaustive computational power. Therefore, saving resources from Convolutional Neural Networks models enhances the integration and use of this and other integrated tools. Thus, we propose the reformulation of the convolutional layers through Local Binary Descriptors (LBD). With this, we introduce two new layers for deep models: (i) Local Feature Detection (LFD), and (ii) Local Descriptor Convolution (LDC), as a viable and efficient alternative to convolutional layers. Local Binary Descriptors inspire the design fundamentals of these two layers. The Local Feature Detection layer is responsible for producing the Feature Detection and Description process through the chosen Local Binary Descriptor. The Local Descriptor Convolution layer, on the other hand, consists of a set of sparse and binary convolutional filters reformulated through the Local Binary Descriptor, followed by a non-linear activation function. By implementing a ResNet-152 with DescNet blocks, we achieved good results over the traditional convolutional layer in competitive visual datasets (MNIST and CIFAR-10), while allowing savings the number of learnable parameters from the model and, consequently, significant computational savings, making DescNet applicable in real environments, where resources are scarce and limited. We also present, as a proposal for future work, the integration of a hybrid model of Artificial Neural Networks (ANN) architecture — the Long-term Recurrent Convolutional Network (LRCN) with DescNet blocks with a Visual SLAM (VSLAM) system, able to solve the Loop Closure Detection problem. A theoretical study and experiments support the proposed approach.

**Keywords:** Feature Detection and Description. Bag of Visual Features. Multilayer Perceptron. Convolutional Neural Network. Descriptor Convolutional Neural Network. Loop Closure Detection. Visual SLAM. Jetson Nano.

## LISTA DE ILUSTRAÇÕES

Figura 1 – Influência da incerteza na localização e mapeamento. Fonte: (GUIZILINI, 2008) apud (THRUN; BURGARD; FOX, 2005). . . . .	20
Figura 2 – Visão esquemática do problema de Detecção de Fechamento de <i>Loop</i> . Fonte: (MOREIRA, 2017). . . . .	22
Figura 3 – Importância da Detecção de Fechamento de <i>Loop</i> . Fonte: Adaptado de (CLEMENTE et al., 2007). . . . .	23
Figura 4 – Pirâmide escala-espço da imagem estabelecida pelo algoritmo SIFT. Fonte: Adaptado de (LOWE et al., 1999). . . . .	35
Figura 5 – Detecção de extremos no espaço de escala gaussiana estabelecida pelo algoritmo SIFT. Fonte: Adaptado de (LOWE et al., 1999). . . . .	36
Figura 6 – Aproximações de núcleo Gaussiano estabelecida pelo algoritmo SURF. Fonte: Adaptado de (BAY; TUYTELAARS; GOOL, 2006). . . . .	38
Figura 7 – Comparação entre Escala Gaussiana e Escala de Difusão Não Linear. Fonte: Adaptado de (ALCANTARILLA; BARTOLI; DAVISON, 2012). . . . .	40
Figura 8 – Padrão de 128 pares de amostragem do algoritmo BRIEF. Fonte: Adaptado de (CALONDER et al., 2010). . . . .	41
Figura 9 – Detecção de bordas estabelecida pelo algoritmo FAST. Fonte: (LOWE et al., 1999). . . . .	42
Figura 10 – Padrão de amostragem do algoritmo ORB. Fonte: (KRIG, 2014). . . . .	43
Figura 11 – Pirâmide escala-espço da imagem estabelecida pelo algoritmo BRISK. Fonte: Adaptado de (LEUTENEGGER; CHLI; SIEGWART, 2011). . . . .	44
Figura 12 – Padrão de amostragem do algoritmo BRISK. Fonte: (LEUTENEGGER; CHLI; SIEGWART, 2011). . . . .	44
Figura 13 – Padrão de amostragem do algoritmo FREAK. Fonte: (ALAHY; ORTIZ; VANDERGHEYNST, 2012). . . . .	46
Figura 14 – Representação da abordagem Saco de Características Visuais. Fonte: Elaborado pela autora. . . . .	49
Figura 15 – Arquitetura básica do modelo <i>Perceptron</i> Multicamadas. Fonte: Adaptado de (BEALE; JACKSON, 1990). . . . .	50
Figura 16 – Abordagem Saco de Características Visuais com classificador <i>Perceptron</i> Multicamadas para etapa de treinamento. Fonte: Elaborado pela autora. . . . .	51
Figura 17 – Abordagem Saco de Características Visuais com classificador <i>Perceptron</i> Multicamadas para etapa de teste. Fonte: Elaborado pela autora. . . . .	51
Figura 18 – Arquitetura básica do modelo de Rede Neural Convolutacional. Fonte: Adaptado de (PEEMEN; MESMAN; CORPORAL, 2011). . . . .	53

Figura 19 – Representação de um <i>loop</i> em uma arquitetura de Rede Neural Recorrente. Fonte: Adaptado de (OLAH, 2015). . . . .	54
Figura 20 – Representação de uma arquitetura de Rede Neural Recorrente desenrolada. Fonte: Adaptado de (OLAH, 2015). . . . .	54
Figura 21 – Representação de eficiência com dependências de curto prazo de uma arquitetura de Rede Neural Recorrente. Fonte: Adaptado de (OLAH, 2015). . . . .	55
Figura 22 – Representação de ineficiência com dependências de longo prazo de uma arquitetura de Rede Neural Recorrente. Fonte: Adaptado de (OLAH, 2015). . . . .	55
Figura 23 – Bloco do modelo de Rede de Memória Longa de Curto Prazo tradicional. Fonte: Adaptado de (GREFF et al., 2016). . . . .	56
Figura 24 – Arquitetura básica do modelo híbrido de arquitetura de Rede Neural Artificial. Fonte: Adaptado de (DONAHUE et al., 2015). . . . .	57
Figura 25 – Representação da detecção de pontos-chave utilizando a biblioteca <i>OpenCV</i> . Fonte: Elaborado pela autora. . . . .	69
Figura 26 – Representação de um ponto-chave utilizando a biblioteca <i>OpenCV</i> . Fonte: Elaborado pela autora. . . . .	69
Figura 27 – Fluxograma da camada de Detecção de Características Locais. Fonte: Elaborado pela autora. . . . .	70
Figura 28 – Fluxograma da camada de Convolução de Descritor Local. Fonte: Elaborado pela autora. . . . .	71
Figura 29 – Comportamento da convolução através dos 2 pesos de descritor, gerados através da reformulação do descritor BRISK. . . . .	72
Figura 30 – Arquitetura básica do modelo de Rede Neural Convolutacional de Descritores. Fonte: Elaborado pela autora. . . . .	73
Figura 31 – Visão esquemática do microcomputador NVIDIA Jetson Nano. Fonte: (NVIDIA, 2019) . . . . .	75
Figura 32 – Estrutura das camadas escondidas do classificador <i>Perceptron</i> Multicamadas. Fonte: Elaborado pela autora. . . . .	78
Figura 33 – Amostras presente no conjunto de dados visuais MNIST. Fonte: Adaptado de (LECUN et al., 1998) . . . . .	79
Figura 34 – Distribuição de amostras do conjunto de dados visuais MNIST. Fonte: Elaborado pela autora. . . . .	79
Figura 35 – Amostras presente no conjunto de dados visuais JAFFE. Fonte: Adaptado de (LYONS et al., 1998) . . . . .	80
Figura 36 – Distribuição de amostras do conjunto de dados visuais JAFFE. Fonte: Elaborado pela autora. . . . .	80

Figura 37 – Amostras presente no conjunto de dados visuais Extended CK+. Fonte: Adaptado de (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010) . . . . .	81
Figura 38 – Distribuição de amostras do conjunto de dados visuais Extended CK+. Fonte: Elaborado pela autora. . . . .	81
Figura 39 – Amostras presente no conjunto de dados visuais FEI. Fonte: Adaptado de (THOMAZ; GIRALDI, 2010) . . . . .	82
Figura 40 – Distribuição de amostras do conjunto de dados visuais FEI. Fonte: Elaborado pela autora. . . . .	82
Figura 41 – Amostras presente no conjunto de dados visuais CIFAR-10. Fonte: Adaptado de (KRIZHEVSKY; HINTON et al., 2009) . . . . .	83
Figura 42 – Distribuição de amostras do conjunto de dados visuais CIFAR-10. Fonte: Elaborado pela autora. . . . .	83
Figura 43 – Amostras presente no conjunto de dados visuais FER-2013. Fonte: Adaptado de (GOODFELLOW et al., 2013) . . . . .	84
Figura 44 – Distribuição de amostras do conjunto de dados visuais FER-2013. Fonte: Elaborado pela autora. . . . .	84
Figura 45 – Conjuntos de dados visuais reorganizarmos por tamanho (quantidade de amostras $\times$ resolução). Fonte: Elaborado pela autora. . . . .	87
Figura 46 – Matriz de Confusão sobre conjunto de dados visuais FEI com o descritor BRISK com o modelo ML5. Fonte: Elaborado pela autora. . . . .	92
Figura 47 – Taxa de perda de treinamento sobre o conjunto de dados visuais FEI com o descritor BRISK com o modelo ML5. Fonte: Elaborado pela autora. . . . .	93
Figura 48 – Teste de Variação da Taxa de Aprendizado. Fonte: Elaborado pela autora. . . . .	97
Figura 49 – Curvas de aprendizado no conjunto de dados visuais MNIST. Fonte: Elaborado pela autora. . . . .	98
Figura 50 – Matriz de Confusão sobre conjunto de dados visuais MNIST. Fonte: Elaborado pela autora. . . . .	100
Figura 51 – Curvas de aprendizado no conjunto de dados visuais CIFAR-10. Fonte: Elaborado pela autora. . . . .	101
Figura 52 – Matriz de Confusão sobre conjunto de dados visuais CIFAR-10. Fonte: Elaborado pela autora. . . . .	103
Figura 53 – Sistema proposto de Detecção de Fechamento de <i>Loop</i> baseado no modelo híbrido de arquitetura de Rede Neural Artificial proposta. Fonte: Elaborado pela autora. . . . .	107
Figura 54 – Interface do <i>software SD Memory Card Formatter</i> . Fonte: Elaborado pela autora. . . . .	121
Figura 55 – Interface do <i>software Etcher</i> . Fonte: Elaborado pela autora. . . . .	122
Figura 56 – Interface Etcher — parâmetros selecionados. Fonte: Elaborado pela autora. . . . .	123
Figura 57 – Mensagem de alerta do <i>Windows</i> . Fonte: Elaborado pela autora. . . . .	123

## LISTA DE TABELAS

Tabela 1 – Recursos de Inteligência Artificial incorporados em processos de negócios. Fonte: Adaptado de (HAI, 2021). . . . .	27
Tabela 2 – Invariância dos Descritores. Fonte: Elaborado pela autora. . . . .	59
Tabela 3 – Comparação entre métodos utilizados pelos principais trabalhos relacionados. Fonte: Elaborado pela autora. . . . .	65
Tabela 4 – Pares de Descritores e Detectores. Fonte: Elaborado pela autora. . . . .	77
Tabela 5 – Tempo de processamento (min.) na etapa de Representação de Características da abordagem Saco de Características Visuais da etapa de treinamento. Fonte: Elaborado pela autora. . . . .	88
Tabela 6 – Tempo de processamento (min.) na etapa de Geração de Vocabulário Visual da abordagem Saco de Características Visuais da etapa de treinamento. Fonte: Elaborado pela autora. . . . .	89
Tabela 7 – Tempo de processamento (min.) na etapa de Representação de Imagem da abordagem Saco de Características Visuais da etapa de treinamento. Fonte: Elaborado pela autora. . . . .	89
Tabela 8 – Tempo de processamento (min.) nas etapas de Representação de Características e Representação de Imagem da abordagem Saco de Características Visuais da etapa de teste. Fonte: Elaborado pela autora. . . . .	90
Tabela 9 – Taxa de acurácia (%) nas etapas de teste do classificador <i>Perceptron</i> Multicamadas no conjunto de dados visuais FEI. Fonte: Elaborado pela autora. . . . .	90
Tabela 10 – Taxa de acurácia (%) na etapa do teste classificador <i>Perceptron</i> Multicamadas em cada conjunto de dados visuais. Fonte: Elaborado pela autora. . . . .	91
Tabela 11 – Média (%) dos valores de Precisão, <i>Recall</i> , <i>F1-Score</i> e Suporte sobre o conjunto de dados visuais FEI com o descritor BRISK com o modelo ML5. Fonte: Elaborado pela autora. . . . .	93
Tabela 12 – Média (%) das métricas de precisão, <i>Recall</i> , <i>F1-Score</i> e suporte sobre o conjunto de dados visuais MNIST. Fonte: Elaborado pela autora. . . . .	100
Tabela 13 – Média (%) das métricas de precisão, <i>Recall</i> , <i>F1-Score</i> e suporte sobre o conjunto de dados visuais CIFAR-10. Fonte: Elaborado pela autora. . . . .	103
Tabela 14 – Comparação entre DescNet e linha de Base. Fonte: Elaborado pela autora. . . . .	104

## LISTA DE ABREVIATURAS E SIGLAS

AGV	<i>Autonomous Guided Vehicles</i>
AI	<i>Artificial Intelligence</i>
AKAZE	<i>Accelerated KAZE</i>
ANN	<i>Artificial Neural Networks</i>
AOS	<i>Additive Operator Splitting</i>
AUV	<i>Autonomous Underwater Vehicles</i>
BEBLID	<i>Boosted Efficient Binary Local Image Descriptor</i>
BNN	<i>Binarized Neural Networks</i>
BRIEF	<i>Binary Robust Independent Elementary Features</i>
BRISK	<i>Binary Robust Invariant Scalable Keypoints</i>
BoVF	<i>Bag of Visual Features</i>
BoVW	<i>Bag of Visual Words</i>
BoW	<i>Bag of Words</i>
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CBIR	<i>Content-Based Image Retrieval</i>
CIFAR-100	<i>Canadian Institute For Advanced Research 100</i>
CIFAR-10	<i>Canadian Institute For Advanced Research 10</i>
CK+	<i>Cohn-Kanade</i>
CNN	<i>Convolutional Neural Networks</i>
CPU	<i>Central Process Unit</i>
CVAE	<i>Conditional Variational Autoencoder</i>
CV	<i>Computer Vision</i>
DESC-NET	<i>Descriptor Convolutional Neural Network</i>
DL	<i>Deep Learning</i>

DS	Demanda Social
DoG	<i>Difference of Gaussian</i>
EBIA	Estratégia Brasileira de Inteligência Artificial
EKF	<i>Extended Kalman Filter</i>
FAST	<i>Features from Accelerated Segment Test</i>
FEI	<i>FEI Face Database</i>
FER-2010	<i>Facial Expression Recognition 2010</i>
FLDA	<i>Fisher's Linear Discriminant Analysis</i>
FREAK	<i>Fast Retina Keypoint</i>
GPAI	<i>Global Partnership on Artificial Intelligence</i>
GPU	<i>Graphics Processing Unit</i>
GVC	<i>Generic Visual Categorization</i>
ICML	<i>Challenges in Representation Learning</i>
IFR	<i>International Federation of Robotics</i>
IG	Informação Geométrica
ILSVRC-2017	<i>ImageNet Large Scale Visual Recognition Challenge 2017</i>
IME	Instituto Militar de Engenharia
IR	<i>Information Retrieval</i>
ISOMAP	<i>Isometric Mapping</i>
JAFFE	<i>Japanese Female Facial Expression</i>
K-NN	<i>K-Nearest Neighbors</i>
LATCH	<i>Learned Arrangements of Three Patch Codes</i>
LBCNN	<i>Local Binary Convolutional Neural Networks</i>
LBC	<i>Local Binary Convolution</i>
LBD	<i>Local Binary Descriptors</i>
LBP	<i>Local Binary Pattern</i>

LDB	<i>Local Difference Binary</i>
LDC	<i>Local Descriptor Convolution</i>
LFDA	<i>Local Fisher's Discriminant Analysis</i>
LFD	<i>Local Feature Detection</i>
LLE	<i>Locally Linear Embedding</i>
LRCN	<i>Long-term Recurrent Convolutional Networks</i>
LiDAR	<i>Light Detection And Ranging</i>
MCTI	Ministério da Ciência, Tecnologia e Inovações
MLP	<i>Multilayer Perceptron</i>
ML	<i>Machine Learning</i>
MNIST	<i>Modified National Institute of Standards and Technology database</i>
NDF	<i>Nonlinear Diffusion Filtering</i>
NLP	<i>Natural Language Processing</i>
ORB	<i>Oriented FAST and Rotated BRIEF</i>
PCA	<i>Principal component analysis</i>
PPgED	Programa de Pós-graduação em Engenharia de Defesa
PPgSC	Programa de Pós-graduação em Sistemas e Computação
PR	<i>Pattern Recognition</i>
RGB	<i>Red, Green, Blue</i>
RNN	<i>Recurrent Neural Networks</i>
ReLU	<i>Rectified Linear Units</i>
RoboLAB	Laboratório de Robótica e Inteligência Computacional
SGD	<i>Stochastic Gradient Descent</i>
SIFT	<i>Scale Invariant Feature Transform</i>
SLAM	<i>Simultaneous Localization and Mapping</i>
SSE	<i>Sum of Squared Errors</i>

SURF	<i>Speeded-up Robust Features</i>
SVM	<i>Support Vector Machine</i>
Stanford HAI	<i>Stanford Institute for Human-Centered Artificial Intelligence</i>
TM	<i>Template Matching</i>
UAV	<i>Unmanned Aerial Vehicles</i>
UGV	<i>Unmanned Ground Vehicles</i>
VO	<i>Visual Odometry</i>
VSLAM	<i>Visual SLAM</i>

# SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b>	<b>20</b>
1.1	MOTIVAÇÃO	25
1.2	JUSTIFICATIVA	25
1.3	OBJETIVOS	29
1.3.1	OBJETIVO GERAL	29
1.3.2	OBJETIVOS ESPECÍFICOS	29
1.4	CONTRIBUIÇÕES	30
1.5	ESTRUTURA DA DISSERTAÇÃO	31
<b>2</b>	<b>TÉCNICAS DE DETECÇÃO E DESCRIÇÃO DE CARACTERÍSTICAS EM IMAGENS</b>	<b>33</b>
2.1	DETECÇÃO E DESCRIÇÃO DE CARACTERÍSTICAS	33
2.1.1	<i>SCALE INVARIANT FEATURE TRANSFORM (SIFT)</i>	35
2.1.2	<i>SPEEDED-UP ROBUST FEATURES (SURF)</i>	37
2.1.3	<i>KAZE (KAZE)</i>	39
2.1.4	<i>BINARY ROBUST INDEPENDENT ELEMENTARY FEATURES (BRIEF)</i>	41
2.1.5	<i>ORIENTED FAST AND ROTATED BRIEF (ORB)</i>	42
2.1.6	<i>BINARY ROBUST INVARIANT SCALABLE KEYPOINTS (BRISK)</i>	43
2.1.7	<i>FAST RETINA KEYPOINT (FREAK)</i>	45
2.1.8	<i>ACCELERATED KAZE (AKAZE)</i>	46
2.2	SACO DE CARACTERÍSTICAS VISUAIS	46
2.2.1	REPRESENTAÇÃO DE CARACTERÍSTICAS	47
2.2.2	GERAÇÃO DE VOCABULÁRIO VISUAL	47
2.2.3	REPRESENTAÇÃO DE IMAGEM	48
2.2.4	<i>PERCEPTRON</i> MULTICAMADAS	49
2.3	REDES NEURAS CONVOLUCIONAIS	52
2.4	REDES NEURAS RECORRENTES	54
2.4.1	REDE DE MEMÓRIA LONGA DE CURTO PRAZO	56
2.5	MODELO HÍBRIDO DE ARQUITETURA DE REDE NEURAL ARTIFICIAL	57
<b>3</b>	<b>REVISÃO DE LITERATURA</b>	<b>59</b>
3.1	COMPARATIVO ENTRE TRABALHOS RELACIONADOS E ABORDAGEM PROPOSTA	64
3.2	COMENTÁRIOS GERAIS	66
<b>4</b>	<b>FORMULAÇÃO DO PROBLEMA E SOLUÇÃO PROPOSTA</b>	<b>67</b>

4.1	PROBLEMA PROPOSTO . . . . .	67
4.1.1	RESULTADOS ESPERADOS . . . . .	67
4.2	SOLUÇÃO PROPOSTA . . . . .	68
4.2.1	REDE NEURAL CONVOLUCIONAL DE DESCRITORES . . . . .	68
4.2.1.1	DETECÇÃO DE CARACTERÍSTICAS LOCAIS . . . . .	69
4.2.1.2	CONVOLUÇÃO DE DESCRITOR LOCAL . . . . .	70
<b>5</b>	<b>MATERIAIS E MÉTODOS . . . . .</b>	<b>74</b>
5.1	KIT DE DESENVOLVIMENTO NVIDIA JETSON NANO . . . . .	74
5.2	CONFIGURAÇÃO EXPERIMENTAL . . . . .	76
5.3	CONJUNTOS DE DADOS VISUAIS . . . . .	78
<b>6</b>	<b>EXPERIMENTOS E RESULTADOS . . . . .</b>	<b>86</b>
6.1	AVALIAÇÃO COMPARATIVA ENTRE DESCRITORES DE CARACTERÍSTICAS POR MEIO DA ABORDAGEM DE SACO DE CARACTERÍSTICAS VISUAIS COM <i>PERCEPTRON</i> MULTICAMADAS . . . . .	86
6.1.1	ABORDAGEM PROPOSTA . . . . .	86
6.1.2	RESULTADOS . . . . .	86
6.2	REDE NEURAL CONVOLUCIONAL DE DESCRITORES . . . . .	94
6.2.1	ABORDAGEM PROPOSTA . . . . .	94
6.2.2	DETALHES DA IMPLEMENTAÇÃO . . . . .	95
6.2.3	RESULTADOS . . . . .	97
6.2.3.1	RESULTADOS NO CONJUNTO DE DADOS VISUAIS MNIST . . . . .	97
6.2.3.2	RESULTADOS NO CONJUNTO DE DADOS VISUAIS CIFAR-10 . . . . .	101
6.2.3.3	COMPARAÇÃO COM LINHA DE BASE . . . . .	104
6.3	COMENTÁRIOS GERAIS . . . . .	104
<b>7</b>	<b>CONSIDERAÇÕES FINAIS . . . . .</b>	<b>106</b>
7.1	CONCLUSÃO . . . . .	106
7.2	TRABALHOS FUTUROS . . . . .	106
	<b>REFERÊNCIAS . . . . .</b>	<b>109</b>
	<b>A – JETSON NANO: ETAPAS DE INSTALAÇÃO . . . . .</b>	<b>120</b>
A.1	GRAVAÇÃO DA IMAGEM NO CARTÃO MICROSD . . . . .	120
A.1.1	FORMATAÇÃO DO CARTÃO MICROSD . . . . .	120
A.1.2	GRAVAÇÃO DA IMAGEM NO CARTÃO MICROSD . . . . .	121
	<b>B – JETSON NANO: ETAPAS DE CONFIGURAÇÃO . . . . .</b>	<b>124</b>
B.1	EXPANSÃO DA MEMÓRIA <i>SWAP</i> . . . . .	124

B.2	INSTALAÇÃO DA BIBLIOTECA <i>OPENCV</i> . . . . .	125
B.2.1	ATUALIZAÇÃO DE PACOTES . . . . .	125
B.2.2	PACOTES NECESSÁRIOS PARA FORMATOS DE IMAGEM E VÍDEO . . . . .	125
B.2.3	DOWNLOAD DOS MÓDULOS DO <i>OPENCV</i> E CONTRIBS . . . . .	125
B.2.4	DESCOMPACTAÇÃO DOS PACOTES . . . . .	125
B.2.5	RENOMEANDO OS DIRETÓRIOS . . . . .	125
B.2.6	CRIAÇÃO DE UM NOVO DIRETÓRIO . . . . .	125
B.2.7	CRIANDO O <i>OPENCV</i> ATRAVÉS DO CMAKE . . . . .	126
B.2.8	COMPILAÇÃO DO <i>OPENCV</i> COM OS MÓDULOS CONTRIBS . . . . .	126
B.3	INSTALAÇÃO DE OUTRAS DEPENDÊNCIAS . . . . .	126
B.3.1	PIP3 . . . . .	127
B.3.2	CYTHON . . . . .	127
B.3.3	NUMPY . . . . .	127
B.3.4	MATPLOTLIB . . . . .	127
B.3.5	PANDAS . . . . .	127
B.3.6	SCIPY . . . . .	127
B.3.7	SCIKIT-LEARN . . . . .	127
B.3.8	KERAS . . . . .	127
B.3.9	TENSORFLOW . . . . .	128
	<b>C – REPOSITÓRIOS . . . . .</b>	<b>129</b>
	<b>A – CONTRIBUIÇÕES CIENTÍFICAS . . . . .</b>	<b>130</b>

# 1 INTRODUÇÃO

A Robótica Móvel Autônoma Inteligente é um campo de pesquisa da Robótica Moderna bastante explorada nas últimas décadas (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011; KIM et al., 2018). Um problema fundamental abordado pela Robótica Móvel em tarefas de navegação autônoma de plataformas robóticas, é o problema de Localização e Mapeamento Simultâneos (em inglês, *Simultaneous Localization and Mapping* — SLAM) (DURRANT-WHYTE; BAILEY, 2006; BAILEY; DURRANT-WHYTE, 2006; CSORBA, 1997; GUIVANT; NEBOT; BAIKER, 2000a; WILLIAMS et al., 2000) que constitui-se na resolução de tarefas complexas, tais quais: (a) mapeamento, que consiste na composição (construção ou reconhecimento incremental) do ambiente onde a plataforma robótica móvel está inserida através de um mapa, concomitantemente à tarefa de (b) localização, que consiste no reconhecimento de sua posição ao longo do caminho neste ambiente pela plataforma robótica móvel utilizando como referência o mapa gerado. Entretanto, estas duas tarefas não podem ser realizadas de formas independentes, isto é, elas atuam de forma complementar. É importante que a plataforma robótica móvel monitore o ambiente em que está inserida, para a realização da captura de dados e informações sobre a pose (posição e orientação) da mesma, e a captura de dados e informações sobre possíveis obstáculos (estáticos ou dinâmicos) neste ambiente. Desta forma, a plataforma robótica móvel torna-se capaz de traçar sua trajetória, determinando passos para a execução de uma tarefa estabelecida de forma eficiente e segura, sem nenhum tipo de auxílio externo. A influência da incerteza na localização e mapeamento pode ser observada na Figura 1.

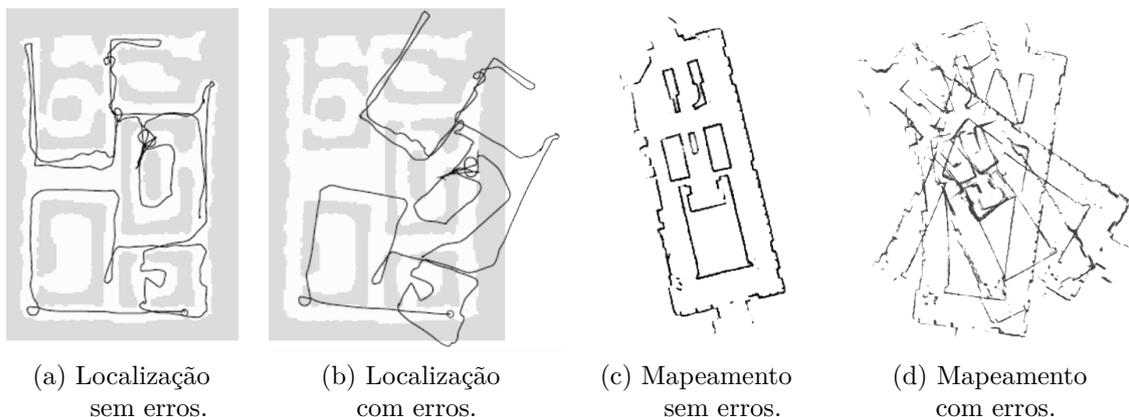


Figura 1 – Influência da incerteza na localização e mapeamento. Fonte: (GUIZILINI, 2008) apud (THRUN; BURGARD; FOX, 2005).

Em ambos os cenários vistos na Figura 1, podemos observar um conjunto de dados coletados em um ambiente interno por uma plataforma robótica móvel. Da esquerda para a direita, na Figura 1a, é possível observar a localização apropriada, isto é, correta, no

ambiente interno; enquanto, na Figura 1b, observamos o acúmulo do erro no decorrer da trajetória da plataforma robótica móvel neste mesmo ambiente, gerando, desta forma, uma localização incorreta. Já na Figura 1c, é possível observar o mapa gerado sem a apresentação de erros; enquanto, na Figura 1d, observamos a geração errônea do mapa, desta forma, produzindo um mapa que não condiz com o mapa real do ambiente interno.

O problema de SLAM abriu uma gama de possibilidades para aplicações de Veículos Terrestres não Tripulados (em inglês, *Unmanned Ground Vehicles* — UGV) no âmbito civil e militar (GUIVANT; NEBOT; BAIKER, 2000b; WANG; THORPE; THRUN, 2003; LEVINSON; MONTEMERLO; THRUN, 2007; LEE; FRAUNDORFER; POLLEFEYS, 2013; BRESSON et al., 2016; HÄNE et al., 2017). Veículos de Combate não Tripulados e carros autônomos são exemplos de Veículos Terrestres não Tripulados, que nada mais são do que veículos que não exigem a presença de um ser humano a bordo para conduzi-los; desta forma, os Veículos Terrestres não Tripulados podem exercer uma navegação autônoma, através de um computador embarcado, por exemplo. Defesa, operações militares, busca, resgate e salvamento, e serviços domésticos são apenas algumas das diversas áreas de aplicação do uso de Veículos Terrestres não Tripulados.

Uma das principais tarefas de Veículos Terrestres não Tripulados consiste na tarefa de localização. A tarefa de localização, como já mencionada anteriormente, permite a uma plataforma robótica móvel ser capaz de navegar em um ambiente de interação, rastreando seus passos, detectando obstáculos e evitando-os apropriadamente para que a execução de uma determinada tarefa seja finalizada com sucesso (i.e., através da coleta de dados e informações neste ambiente). Um método robusto utilizado para a resolução da tarefa de localização de Veículos não Tripulados é a Odometria fundamentada na visão — a Odometria Visual (em inglês, *Visual Odometry* — VO) (NISTÉR; NARODITSKY; BERGEN, 2004; NISTÉR; NARODITSKY; BERGEN, 2006; FLOROS; ZANDER; LEIBE, 2013; NAIXIN et al., 2019). A Odometria Visual é aplicada em diversos campos de pesquisa e possui uma vasta lista de aplicações em distintos sistemas robóticos móveis, tais como, Veículos Subaquáticos Autônomos (em inglês, *Autonomous Underwater Vehicles* — AUV), Veículos Terrestres não Tripulados e Veículos de Exploração Espacial não Tripulados, como é o caso dos veículos *Spirit* e *Opportunity* (CHENG; MAIMONE; MATTHIES, 2005; MAIMONE; CHENG; MATTHIES, 2007), e *Perseverance* (FARLEY et al., 2020; MAKI et al., 2020).

Desta forma, nos últimos anos, surge um crescente interesse na utilização de câmeras, como uma alternativa barata a sensores e *lasers* para tarefas de construção de mapas e localização de SLAM, provando que é possível aliar técnicas de Visão Computacional (em inglês, *Computer Vision* — CV) com o problema de SLAM, o que é conhecido na literatura como SLAM baseado em visão — VSLAM (em inglês, *Visual SLAM*). Um tópico em destaque é a Detecção de Fechamento de *Loop* para a correção da Odometria

Visual, tornando-se uma tarefa desafiadora para um sistema VSLAM, dado que o principal objetivo da Detecção de Fechamento de *Loop* está em corrigir a deriva, ou seja, corrigir a incerteza nos cálculos estimados acumulada no decorrer da trajetória da plataforma robótica móvel. Com isto, a plataforma robótica móvel é capaz de reconhecer quando regressou a um local previamente mapeado em um ambiente de interação (i.e., fechamento de *loop* em um mapa), e neste momento terá informações sobre sua pose e localização. Uma visão esquemática do problema de Detecção de Fechamento de *Loop* para uma plataforma robótica móvel, onde a mesma deve percorrer uma trajetória circular em relação a um sistema fixo de coordenadas  $\Sigma_{fixo}$  pode ser observada na Figura 2.

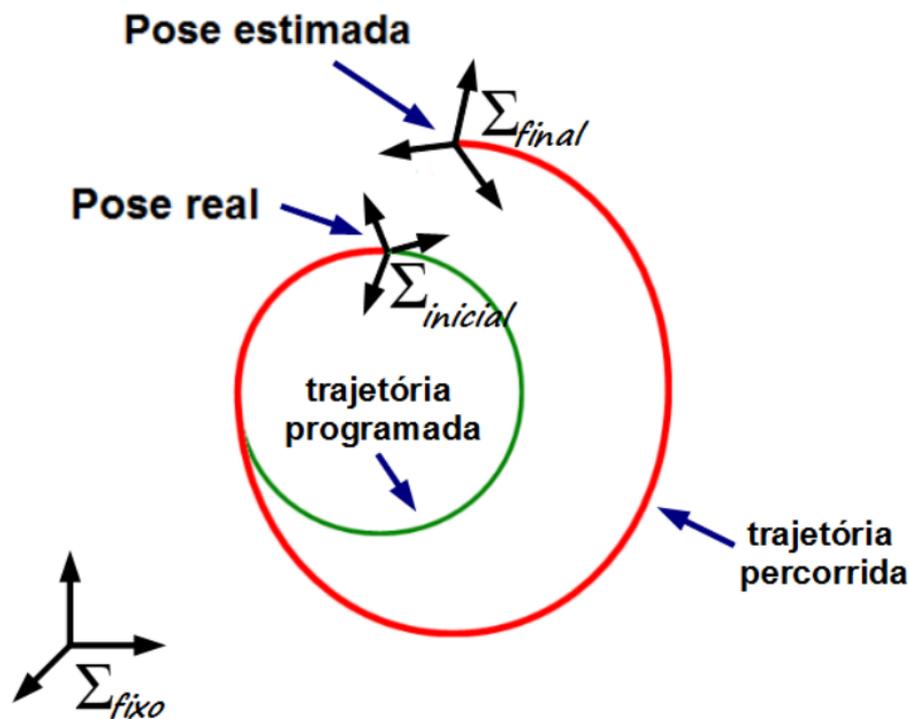


Figura 2 – Visão esquemática do problema de Detecção de Fechamento de *Loop*. Fonte: (MOREIRA, 2017).

Na Figura 2, em verde, é representada uma trajetória programada, onde sua origem é determinada pelo ponto  $(x_{inicial}, y_{inicial}, \theta_{inicial})$ . Desde modo, a pose inicial da plataforma robótica móvel está associada ao sistema de coordenadas  $\Sigma_{inicial}$ , enquanto, em vermelho, é representada a trajetória realizada, onde seu término é determinado no ponto  $(x_{final}, y_{final}, \theta_{final})$ . Desde modo, a pose final da plataforma robótica móvel está associada ao sistema de coordenadas  $\Sigma_{final}$ . A diferença entre os sistemas de coordenadas  $\Sigma_{final}$  e  $\Sigma_{inicial}$  pode ser caracterizada como o acúmulo de erros da Odometria Visual. Deste modo, tais informações coletadas são aplicadas para corrigir a deriva nos cálculos estimados da Odometria Visual para um sistema VSLAM. A importância da Detecção de Fechamento de *Loop* para a correção da Odometria Visual para um sistema VSLAM pode ser observada na Figura 3.

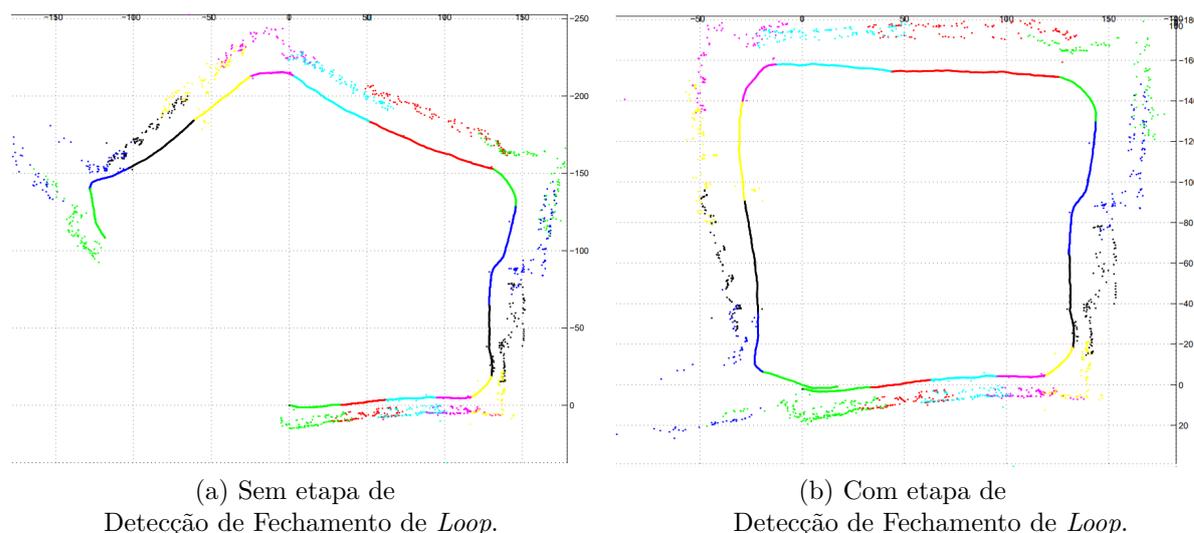


Figura 3 – Importância da Detecção de Fechamento de *Loop*. Fonte: Adaptado de (CLEMENTE et al., 2007).

Em ambos os cenários vistos na Figura 3, podemos observar a trajetória percorrida por uma plataforma robótica móvel onde é avaliada a odometria gerada. Da esquerda para a direita, na Figura 3a, é possível observar que a pose final difere da pose inicial da plataforma robótica móvel; enquanto, na Figura 3b, observamos que a etapa de Detecção de Fechamento de *Loop* corrigiu a trajetória estimada pela Odometria Visual, visto que, a pose final e a pose inicial da plataforma robótica móvel são as mesmas. Portanto, a Detecção de Fechamento de *Loop* é um tópico de relevância na área do VSLAM.

Em vista disto, nas duas últimas décadas observa-se o crescente estudo de outras técnicas para solucionar o problema de VSLAM, como é o caso do Reconhecimento de Padrões (em inglês, *Pattern Recognition* — PR) e Aprendizado de Máquina (em inglês, *Machine Learning* — ML) baseado em Aprendizado Profundo (em inglês, *Deep Learning* — DL). Técnicas como Redes Neurais Convolucionais (em inglês, *Convolutional Neural Networks* — CNN) (ZHANG et al., 1998) e Redes Neurais Recorrentes (em inglês, *Recurrent Neural Networks* — RNN) (RUMELHART; HINTON; WILLIAMS, 1985) proporcionam bons resultados no processo de extração de características e padrões de classificação em imagens e em tarefas de VSLAM (WANG et al., 2017; KUMAR; BHANDARKAR; PRASAD, 2018; VALENTE; JOLY; FORTELLE, 2019). Deste modo, a realização de tarefas de reconhecimento de objetos (em geral, em imagens) torna-se uma tarefa fácil e corriqueira nas relações interpessoais dos seres humanos, porém, computacionalmente, tal tarefa apresenta um alto grau de complexidade. Com isto, as arquiteturas de Redes Neurais Convolucionais, originalmente proposta por (LECUN et al., 1998), vêm conquistando espaço em desafios de reconhecimento e classificação de imagens a partir do ano de 2012 no desafio *Imagenet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). A partir de estudos, pesquisas e aprimoramentos, hoje existem diferentes modelos de arquiteturas de Redes

Neurais Convolucionais, e seus resultados as tornam estado da arte para a resolução de problemas na área de Visão Computacional. Tais arquiteturas em destaque são: LeNet (LECUN et al., 1998), AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), VGG (SIMONYAN; ZISSERMAN, 2014), GoogLeNet (SZEGEDY et al., 2015), ResNet (HE et al., 2016), Xception (CHOLLET, 2017), entre outras. Além da variedade de arquiteturas disponíveis, as Redes Neurais Convolucionais avançaram também em outros aspectos, tais como: função de ativação, *design* de camadas e otimização. Entretanto, um problema ainda enfrentado ao treinar tais arquiteturas, está relacionado ao poder computacional necessário, um recurso caro ou até mesmo indisponível.

O presente trabalho apresenta uma nova abordagem para explorar filtros convolucionais esparsos e binários em Redes Neurais Convolucionais tradicionais. Assim, propomos a reformulação das camadas convolucionais por meio de Descritores Binários Locais (em inglês, *Local Binary Descriptors* — LBD). Desta forma, apresentamos duas novas camadas para arquiteturas profundas — Detecção de Características Locais (em inglês, *Local Feature Detection* — LFD), como uma primeira camada, e Convolução de Descritor Local (em inglês, *Local Descriptor Convolution* — LDC), como uma alternativa viável e eficiente às camadas convolucionais. Os Descritores Binários Locais inspiram os fundamentos do *design* das camadas de Detecção de Características Locais e Convolução de Descritor Local. A camada de Detecção de Características Locais é responsável por produzir o processo de Detecção e Descrição de características por meio do Descritor Binário Local escolhido. Já a camada de Convolução de Descritor Local consiste em um conjunto de filtros convolucionais esparsos e binários reformulados por meio do Descritor Binário Local, seguidos de uma função de ativação não linear. Um estudo teórico e experimentos apoiam a abordagem proposta, que une Redes Neurais Convolucionais com camadas de Detecção de Características Locais e camadas de Convolução de Descritor Local, denominada neste trabalho de Rede Neural Convolucional de Descritores (em inglês, *Descriptor Convolutional Neural Network* — DescNet). Através dos experimentos e simulações realizados, alcançamos resultados significativos (vide Seção 6.2.3) sobre a camada convolucional tradicional em conjuntos de dados visuais competitivos, com MNIST (LECUN et al., 1998) e CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009), enquanto permite economia no número de parâmetros aprendíveis do modelo e, conseqüentemente, economia computacional significativa, tornando a DescNet um modelo aplicável em ambientes reais, onde há recursos escassos e limitados.

Desta forma, propomos ainda, uma nova abordagem para a solução do problema de Detecção de Fechamento de *Loop* baseado em um modelo híbrido de arquitetura de Rede Neural Artificial (em inglês, *Artificial Neural Networks* — ANN) — a Rede Convolucional Recorrente de Longo Prazo (em inglês, *Long-term Recurrent Convolutional Networks* — LRCN) com blocos DescNet de forma a reduzir o custo computacional executado na etapa de Detecção de Fechamento de *Loop*, fornecendo uma alternativa robusta e barata para

sistemas VSLAM, a ser melhor explorada nas próximas etapas deste trabalho.

## 1.1 Motivação

Com os esforços voltados para os avanços tecnológicos direcionados aos *hardwares*, os recursos da Inteligência Artificial (em inglês, *Artificial Intelligence* — AI) ganham força e, com isso, novas técnicas e métodos estão sendo aplicadas cada vez mais em nosso dia-a-dia. Assim sendo, algoritmos baseados em Aprendizado Profundo, como as Redes Neurais Convolucionais, vêm sendo empregadas em sistemas especializados de detecção e reconhecimento de objetos (em geral, em imagens). Em face disto, a motivação fundamental desta Dissertação consiste na busca de alternativas baratas, robustas e com desempenho computacional eficiente, para explorar filtros convolucionais esparsos e binários em Redes Neurais Convolucionais tradicionais. Assim, propomos a reformulação das camadas convolucionais por meio de Descritores Binários Locais.

Sob esta ótica, somos estimulados nesta proposta, a explorar técnicas esparsas, binárias e ternárias aplicadas a Redes Neurais Convolucionais, o que motiva a integração com uma plataforma robótica autônoma em um ambiente real para a solução do problema de Detecção de Fechamento de *Loop* de um sistema VSLAM, em trabalhos futuros.

## 1.2 Justificativa

Em um estudo publicado pela Federação Internacional de Robótica (em inglês, *International Federation of Robotics* — IFR) em seu relatório de resumo executivo (IFR, 2019a), no que diz respeito ao ano de 2018, é apontado que o maior mercado de robôs industriais no mundo por regiões é a Ásia (incluindo Austrália e Nova Zelândia), enquanto o segundo maior mercado de robôs industriais no mundo é ocupado pela Europa. Ainda segundo este relatório, é descrito que por países: China, Japão, Estados Unidos, República da Coreia e Alemanha detêm juntos 74% das instalações globais de robôs industriais.

Em recente relatório (IFR, 2021a) publicado pela Federação Internacional de Robótica, é apontado que, por conta das constantes inovações técnicas e tendência constante em direção à automação em robôs industriais, desde o ano de 2010, existe um aumento considerável pela demanda de robôs industriais. Entretanto, ainda segundo este relatório, é apresentado que as instalações globais de robôs industriais entre os anos de 2019 e 2020 caíram 12% (373.240 unidades) reflexo direto causado pelas incertezas da economia global e conflitos comerciais. Ainda assim, as instalações globais de robôs industriais obteve uma movimentação financeira de 13,8 bilhões de dólares.

É possível inferir que mesmo com as incertezas da economia global e conflitos comerciais, a pesquisa relacionada ao desenvolvimento científico e tecnológico de robôs

industriais está crescendo nos países citados, ao passo que sua comercialização possui um alto nível de investimento financeiro. O que não nos surpreende, dado que os países citados são países desenvolvidos, que investem em uma produção nacional, e conseqüentemente potencializam o desenvolvimento científico e tecnológico, pontos cruciais para que um país atinja sua soberania. Neste relatório divulgado pela Federação Internacional de Robótica, não há informações referente à posição do Brasil neste cenário; entretanto, em um relatório publicado no ano de 2017 (IFR, 2017), para o ano de 2016, o Brasil atingiu apenas o número total de 1.207 unidades vendidas de robôs industriais.

Ainda segundo um relatório publicado no ano de 2019 (IFR, 2019b) pela Federação Internacional de Robótica, no que diz respeito a robôs de serviços vendidos no mundo no ano de 2018, houve um aumento de 61% (mais de 271.000 unidades) nas vendas de robôs de serviços em comparação ao ano de 2017. O setor de Veículos Guiados Autônomos (em inglês, *Autonomous Guided Vehicles* — AGV), utilizados em ambientes de produção, armazenagem e distribuição possui um total de 41% de todas as unidades vendidas, seguido pelo setor de robôs de inspeção e manutenção que detêm 39% de todas as unidades vendidas, e pelo setor de robôs com aplicações de defesa que conta com 5% de todas as unidades vendidas, sendo os Veículos Aéreos não Tripulados (em inglês, *Unmanned Aerial Vehicles* — UAV) — os famosos drones com maior participação.

No relatório (IFR, 2021b) publicado pela Federação Internacional de Robótica no ano de 2021, o mercado de robôs de serviços cresceu entre 2019 e 2020, 32%, onde se obteve uma movimentação financeira de 11,2 bilhões de dólares. Isso se dá pelo fato de que este mercado não foi afetado pela crise pandêmica global do novo coronavírus; pelo contrário, a crise pandêmica global estimulou o mercado de robôs de serviços, criando uma demanda adicional para aplicações de robôs de serviços. Ainda este estudo, entre 2019 e 2020, as aplicações em defesa representam 15% das vendas totais de robôs de serviço.

A partir dos dados apresentados acima, é possível observar que independente das aplicações, sejam elas, defesa, operações militares, busca, resgate e salvamento, ou serviços domésticos, existe um mercado forte, em crescente estado de desenvolvimento, que empregam o uso de Veículos Terrestres não Tripulados e plataformas robóticas móveis, aplicações estas, que se baseiam em tarefas de navegação autônoma. Como citado acima, em 2020, vimos este mercado explodir ainda mais, em meio a crise pandêmica global do novo coronavírus, visto que o emprego destes robôs de serviço para uso pessoal e doméstico, e robôs médicos, propiciam um maior distanciamento social entre os seres humanos, realizando diversas tarefas cotidianas enquanto transitam em áreas de quarentena sem risco para a proteção individual do ser humano. Isto porque o robô não se infecta com coronavírus e não ficam assintomáticos, portanto não transmitem COVID-19 para outros seres humanos, mantendo os indivíduos e trabalhadores seguros.

Em março de 2021, o Instituto Stanford de Inteligência Artificial Centrada no Hu-

mano (em inglês, *Stanford Institute for Human-Centered Artificial Intelligence* — Stanford HAI) em parceria com organizações da indústria, academia e governo lançou a quarta edição do relatório *AI Index* (HAI, 2021). Nesta edição, é apontado que a Inteligência Artificial e suas sub-áreas tornaram-se uma importante disciplina de pesquisa com um amplo arsenal de aplicações comerciais. Muitos destes robôs de serviço comentados acima são movidos e integrados por sensores, câmeras e técnicas de Inteligência Artificial. Portanto, o estudo e aprimoramento entre técnicas e métodos de Reconhecimento de Padrões e Aprendizado de Máquina baseado em Aprendizado Profundo se fazem cada vez mais necessárias para a integração com este tipo de plataformas robóticas móveis de modo a serem exploradas em cenários da vida real. Entretanto, no relatório é ainda apontado que poucas técnicas e métodos de Inteligência Artificial são implantadas em plataformas robóticas móveis. Com isto, o relatório destaca que para o ano de 2020, o tipo de recurso de Inteligência Artificial adotado, pôde variar dependendo do setor, como pode ser observado na Tabela 1.

Tabela 1 – Recursos de Inteligência Artificial incorporados em processos de negócios.  
Fonte: Adaptado de (HAI, 2021).

Recursos de Inteligência Artificial	Indústria						
	Todas as Indústrias	Automotiva e Montagem	Serviços Comerciais, Jurídicos e Profissionais	Bens de Consumo e Varejo	Serviços Financeiros	Saúde e Farma	Tecnologia e Telecom
Veículos Autônomos	7%	20%	7%	13%	6%	1%	9%
Visão Computacional	18%	33%	13%	10%	18%	15%	34%
Interfaces de Conversação	15%	16%	17%	9%	24%	10%	32%
Aprendizado Profundo	16%	19%	19%	6%	19%	14%	30%
Geração de Linguagem Natural	11%	12%	14%	6%	18%	12%	18%
Compreensão de Fala em Linguagem Natural	12%	14%	15%	6%	19%	11%	25%
Compreensão de Texto em Linguagem Natural	13%	19%	18%	9%	26%	15%	33%
Outras técnicas de Aprendizado de Máquina	23%	27%	25%	12%	32%	19%	37%
Robótica Física	13%	31%	11%	23%	8%	10%	14%
Automação de Processos Robóticos	22%	33%	13%	14%	37%	18%	34%

Como observado na Tabela 1, e também apontado no relatório, é adotado pelas indústrias os recursos de Inteligência Artificial que melhor se adéquam e atendem às suas respectivas funções primárias. Como, por exemplo, a indústria de bens de consumo e varejo que adota 23% dos recursos de Inteligência Artificial relacionados a Robótica Física, por

se tratar de um setor onde a manufatura e a distribuição exercem um papel importante. Outro caso notável, é da indústria automotiva e montagem, que adota igualmente 33% dos recursos de Inteligência Artificial relacionados a Visão Computacional e Automação de Processos Robóticos, por se tratar de um setor onde a manufatura e linhas de montagens baseadas em visão desempenham um papel significativo neste setor.

Ainda no relatório é apontado que, para as próximas décadas, técnicas de Inteligência Artificial estão definidas para moldar a competitividade ao nível global. Desde modo, é apresentado no relatório uma gama de países e regiões ao redor do mundo que, com intuito de promover o desenvolvimento de técnicas de Inteligência Artificial, estabeleceram iniciativas e estratégias para impulsar a implementação de políticas (ao nível governamental e intergovernamental) direcionadas à Inteligência Artificial, abordando diversas vertentes, tais quais, governança, implicações sociais e éticas ao passo que adotam estas tecnologias em meio a sociedade. É relatado ainda, que até dezembro de 2020, mais de 30 outros países e regiões publicaram documentos semelhantes ao nível “*Estratégia de Inteligência Artificial*”, como é apontado no relatório. Este fenômeno pode ser caracterizado pela pressão causada pelo Canadá, que no ano de 2017 publicou o que é considerado como sendo o primeiro documento de estratégia nacional de Inteligência Artificial.

Em 2021, o Ministério da Ciência, Tecnologia e Inovações — MCTI do Brasil estabeleceu pela Portaria MCTI n.º 4.617, de 6 de abril de 2021, alterada pela Portaria MCTI n.º 4.979, de 13 de julho de 2021, a Estratégia Brasileira de Inteligência Artificial — EBIA, tornando-se um pontapé inicial para fomentar e regularizar o uso de Inteligência Artificial em amplo âmbito nacional, já que apresenta apenas diretrizes gerais acerca da temática. Além disso, o Brasil tornou-se membro da Parceria Global em Inteligência Artificial (em inglês, *Global Partnership on Artificial Intelligence* — GPAI ou *Gee-pay*) em dezembro de 2020. A Parceria Global em Inteligência Artificial é uma iniciativa internacional, onde o compromisso está em facilitar colaborações internacionais, difundir o conhecimento teórico e prático, além do apoio a pesquisas de ponta relacionadas com o tema, com o objetivo de tornar-se uma referência global acerca do tema Inteligência Artificial. Ainda segundo o relatório *AI Index*, entre os anos de 2016 a 2020, a contratação de mão de obra especializada em Inteligência Artificial no Brasil cresceu gradativamente, ao lado de outros países como Índia, Canadá, Cingapura e África do Sul.

Como mencionado, para as próximas décadas, técnicas de Inteligência Artificial estão definidas para moldar a competitividade ao nível global. Em vista disto, para que o Estado Brasileiro alcance cada vez mais a sua autoafirmação nacional, independência, autonomia e soberania frente a outros Estados em um contexto internacional, é necessário que haja maiores investimentos e parcerias entre as iniciativas públicas e privadas, além de investimentos em ciência, tecnologia e educação. O investimento em uma produção nacional, ou seja, nestes setores do país, nos proporcionam competências e capacidades estratégicas

que podem ser empregadas no âmbito civil e militar, fortalecendo os diversos segmentos da indústria, principalmente a indústria bélica do país, ao mesmo tempo em que se cria uma maior concorrência e mitiga a dependência de importações, fortalecendo o protagonismo do Estado Brasileiro no cenário internacional, concomitante com o fortalecimento de sua soberania.

Desta forma, esta Dissertação justifica-se pela diversa gama de aplicações, tanto no âmbito civil quanto militar, onde é possível empregar o uso de um sistema de baixo custo, com desempenho computacional eficiente para a solução de detecção e reconhecimento de objetos (em geral, em imagens). Pretendemos ainda, nas próximas etapas, integrar este mesmo sistema para a solução do problema de Detecção de Fechamento de *Loop*, onde o sistema proposto seja capaz de corrigir a deriva nos cálculos estimados da Odometria Visual, que, aplicado a um sistema VSLAM, possa, de alguma forma, trazer valor acadêmico e científico para a comunidade acadêmica de Robótica, e igualmente, de alguma forma, possa atender as demandas dos diversos setores de robôs de serviços, como (já mencionado), defesa, operações militares, busca, resgate e salvamento, e serviços domésticos.

## 1.3 Objetivos

Nesta Seção, serão apresentados o objetivo geral e os objetivos específicos desta Dissertação.

### 1.3.1 Objetivo Geral

O objetivo geral desta Dissertação constitui-se na busca de alternativas baratas, robustas e com desempenho computacional eficiente para explorar filtros convolucionais esparsos e binários em Redes Neurais Convolucionais tradicionais. Sob esta ótica, somos estimulados nesta Dissertação, a explorar técnicas esparsas, binárias e ternárias aplicadas a Redes Neurais Convolucionais. Assim, propomos a reformulação de filtros convolucionais através de Descritores Binários Locais para produzir um modelo esparso e binário como uma alternativa viável a Redes Neurais Convolucionais tradicionais, proporcionando bons resultados no processo de extração de características e padrões de classificação em imagens.

### 1.3.2 Objetivos Específicos

Para que o objetivo geral desta Dissertação seja alcançado, foi estabelecido, o desenvolvimento de objetivos específicos atingidos no decorrer do desenvolvimento deste trabalho, a saber:

- **Comparação entre Descritores Binários Locais:** Este objetivo consiste na revisão bibliográfica de estudos comparativos entre algoritmos de Descritores Binários

Locais. Esta revisão serve de base para a seleção do melhor candidato para a tarefa de extração de características, sendo, posteriormente, reformulado através de filtros convolucionais da arquitetura de Rede Neural Convolucional do nosso modelo. O desafio deste objetivo é crítico, pois a escolha de um Descritor Binário Local que não seja adequado ao problema proposto acarretará, posteriormente, em um modelo ineficiente, com baixo nível de acurácia, o que, igualmente, produzirá resultados não satisfatórios;

- **Reformulação do Descritor Binário Local através de filtros convolucionais:** Este objetivo consiste no estudo de melhores práticas e implementação da reformulação do Descritor Binário Local através de filtros convolucionais, de forma a atingir o mesmo propósito;
- **Implementação da adaptação de Rede Neural Convolucional:** Este objetivo consiste na apresentação do novo *design* de camadas para a arquitetura proposta.
- **Realização de testes:** Este objetivo consiste na realização de experimentos e testes em conjuntos de dados visuais competitivos, usados para avaliar, posteriormente, a eficiência e acurácia da arquitetura proposta;
- **Validação e análise dos resultados:** Este objetivo consiste na validação e análise dos resultados obtidos (e.g., precisão de classificação, consumo computacional, entre outros) no processo de extração de características e padrões de classificação em imagens.

## 1.4 Contribuições

As principais contribuições desenvolvidas ao longo desta Dissertação consistem em:

- Avaliação da abordagem de Saco de Características Visuais (em inglês, *Bag of Visual Features* — BoVF) (inspirado pela técnica Saco de Palavras (em inglês, *Bag of Words* — BoW) (BLEI; NG; JORDAN, 2003), extraindo características através de Descritores de Características Locais (em inglês, *Local Feature Descriptors* — LFD) e Descritores Binários Locais no microcomputador Jetson Nano da NVIDIA para as tarefas de reconhecimento e classificação em seis conjuntos de dados visuais por meio do classificador *Perceptron* Multicamadas (em inglês, *Multilayer Perceptron* — MLP) (MINSKY; PAPER, 1990);
- Reformulação de filtros convolucionais de uma Rede Neural Convolucional tradicional através de Descritores Binários Locais, denominada Rede Neural Convolucional de Descritores — DescNet

- Proposta para a solução do problema de Detecção de Fechamento de *Loop* para sistemas VSLAM baseado em um modelo híbrido de arquitetura de Rede Neural Artificial com filtros convolucionais esparsos e binários.

## 1.5 Estrutura da Dissertação

Para além da **Introdução**, esta Dissertação está organizada da seguinte maneira:

- Capítulo 2 — **Técnicas de Detecção e Descrição de Características em Imagens**: Neste Capítulo é apresentada uma visão geral da fundamentação teórica necessária para compreender os conceitos envolvidos das técnicas utilizadas para o desenvolvimento deste trabalho;
- Capítulo 3 — **Revisão de Literatura**: Neste Capítulo é apresentada uma visão geral dos principais trabalhos analisados no decorrer do desenvolvimento desta Dissertação;
- Capítulo 4 — **Formulação do Problema e Solução Proposta**: Neste Capítulo é apresentado o problema proposto, bem como, a solução proposta;
- Capítulo 5 — **Materiais e Métodos**: Neste Capítulo é apresentado as metodologias adotadas para o desenvolvimento desta Dissertação;
- Capítulo 6 — **Experimentos e Resultados**: Neste Capítulo são apresentados os resultados dos experimentos e simulações realizados no decorrer do desenvolvimento desta Dissertação;
- Capítulo 7 — **Considerações Finais**: Neste Capítulo são apresentadas as considerações finais desta Dissertação, bem como, as contribuições científicas realizadas durante a evolução deste trabalho, a conclusão e sugestões para trabalhos futuros.

Ademais, existem os Apêndices A, B, C, compostos da seguinte maneira:

- Apêndice A — **Jetson Nano: Etapas de Instalação**: Neste Apêndice é apresentado as etapas de instalação do microcomputador Jetson Nano da NVIDIA;
- Apêndice B — **Jetson Nano: Etapas de Configuração**: Neste Apêndice é apresentado as etapas de configuração do microcomputador Jetson Nano da NVIDIA;
- Apêndice C — **Repositórios**: Neste Apêndice é apresentado os links de acesso aos repositórios *git* relacionados aos códigos-fontes desenvolvidos nesta Dissertação.

Existe ainda um Anexo A, a saber:

- Anexo A — **Contribuições Científicas:** Neste Anexo são apresentadas as publicações realizadas durante a evolução deste trabalho de Dissertação.

## 2 TÉCNICAS DE DETECÇÃO E DESCRIÇÃO DE CARACTERÍSTICAS EM IMAGENS

Neste Capítulo, serão apresentados os conceitos básicos e teóricos que dão sustentação para o desenvolvimento desta Dissertação, a saber:

- **Detecção e Descrição de Características:** São apresentados nesta Seção os Descritores de Características Locais e Descritores Binários Locais, técnicas utilizadas para extração de características presentes em imagens;
- **Saco de Características Visuais:** Nesta Seção é apresentada a abordagem de Saco de Características Visuais, técnica utilizada para detectar e calcular semelhanças em imagens;
- **Redes Neurais Convolucionais:** É apresentada nesta Seção a arquitetura básica de uma Rede Neural Convolucional, técnica de Aprendizado de Máquina baseada em Aprendizado Profundo utilizada no processo de extração de características e padrões de classificação em imagens.
- **Redes Neurais Recorrentes:** Nesta Seção é apresentada a arquitetura básica de uma Rede Neural Recorrente, modelo de sequência neural utilizado para trabalhar com dados sequenciais de entrada.
- **Modelo Híbrido de Arquitetura de Rede Neural Artificial:** É apresentada nesta Seção a arquitetura básica do modelo híbrido de arquitetura de Rede Neural Artificial, capaz de suportar previsões de sequência.

### 2.1 Detecção e Descrição de Características

Encontrar “*características interessantes*” em uma imagem é a função de um Descritor de Características Locais (KRIG, 2016). Tal tarefa torna-se importante em aplicações de tópicos relacionados a Visão Computacional e Processamento de Imagens, como é o caso de estrutura em movimento, detecção de objetos e recuperação de imagens.

Na literatura, podemos definir característica como sendo uma informação imprescindível para que a tarefa computacional exercida seja solucionada; desta forma, as características podem ser configuradas como pontos, arestas, cantos, objetos, ou padrões que são factíveis de serem descobertos frequentemente com probabilidades elevadas em uma imagem. Isto posto, podemos classificar um “*característica interessante*” como vários termos que podem ser usados alternadamente, tais quais: ponto de interesse (em inglês,

*interest point*) (KE; SUKTHANKAR et al., 2004; ABDEL-HAKIM; FARAG, 2006; SCOVANNER; ALI; SHAH, 2007), ponto de referência (em inglês, *landmark*), ponto-chave (em inglês, *keypoint*) ou ponto de ancoragem (em inglês, *anchor point*). Neste trabalho utilizaremos o termo ponto-chave para discriminar uma “*característica interessante*”.

Através da descrição de característica é possível obter informações sobre cada ponto-chave, onde, a partir deste momento, é possível realizar a tarefa de correspondência de características, isto porque, de maneira ideal, espera-se que tais informações obtidas sejam invariantes em relação à transformação de alguma forma (e.g., rotação e escala) da imagem, desta forma, sendo possível encontrar novamente a característica. A correspondência de características está presente em muitas aplicações do VSLAM, como reconhecimento de objetos, calibração da câmera e registro de imagens.

Existem duas classes distintas de algoritmos de detecção e descrição de características, a saber: (a) Descritores de Características Locais, e; (b) Descritores Binários Locais. Os Descritores de Características Locais consagrados na literatura são conhecidos como Transformação de Característica Invariável de Escala (em inglês, *Scale Invariant Feature Transform* — SIFT) (LOWE et al., 1999; LOWE, 2004), Características Robustas Aceleradas (em inglês, *Speeded-up Robust Features* — SURF) (BAY; TUYTELAARS; GOOL, 2006), e KAZE, proposto originalmente por ALCANTARILLA; BARTOLI; DAVISON. Já os Descritores Binários Locais são alternativas não patenteadas, leves e rápidas em comparação aos Descritores de Características Locais SIFT e SURF. A vantagem em se utilizar um Descritor Binário Local como alternativa às técnicas SIFT e SURF, é que os Descritores Binários Locais são, em geral, mais rápidos. Em contrapartida, os Descritores Binários Locais tendem a ser menos robustos em comparação ao SIFT e SURF; desta forma, potencializaremos a robustez do Descritor Binário Local ao acoplá-lo em nossa arquitetura de Rede Neural Convolucional proposta.

Os Descritores Binários Locais consagrados na literatura e notáveis em aplicações de SLAM são conhecidos como Características Elementares Independentes Robustos Binários (em inglês, *Binary Robust Independent Elementary Features* — BRIEF) (CALONDER et al., 2010; CALONDER et al., 2011), Descritor Binário Local FAST Orientado e BRIEF Rotacionado (em inglês, *Oriented FAST and Rotated BRIEF* — ORB), proposto originalmente por RUBLEE et al., Pontos-chave Escaláveis Invariáveis e Robustos Binários (em inglês, *Binary Robust Invariant Scalable Keypoints* — BRISK), originalmente proposto por (LEUTENEGGER; CHLI; SIEGWART, 2011), KAZE Acelerado (em inglês, *Accelerated KAZE* — AKAZE) proposto por ALCANTARILLA; SOLUTIONS, e Descritor Binário Local Ponto-chave da Retina Rápida (em inglês, *Fast Retina Keypoint* — FREAK), proposto originalmente por ALAHI; ORTIZ; VANDERGHEYNST. Os três Descritores de Características Locais em conjunto com os cinco Descritores Binários Locais apresentados, são descritos em seguida:

### 2.1.1 Scale Invariant Feature Transform (SIFT)

O Descritor de Características Locais SIFT (LOWE et al., 1999) é um algoritmo de descrição de características locais e contém ainda seu próprio método de detecção de características locais. Este algoritmo extrai diversas características invariantes presentes em imagens, ou seja, fornece invariância à iluminação, escala, rotação, ruído, ponto de vista, sendo considerado um algoritmo com forte robustez; entretanto, não rápido o suficiente.

Para a realização da primeira etapa de detecção de extremos no espaço de escala das imagens é produzida a partir da convolução de *kernels* gaussianos com a imagem de entrada. Deste modo, pode ser observado na Figura 4 a pirâmide escala-espaco da imagem estabelecida pelo algoritmo SIFT.

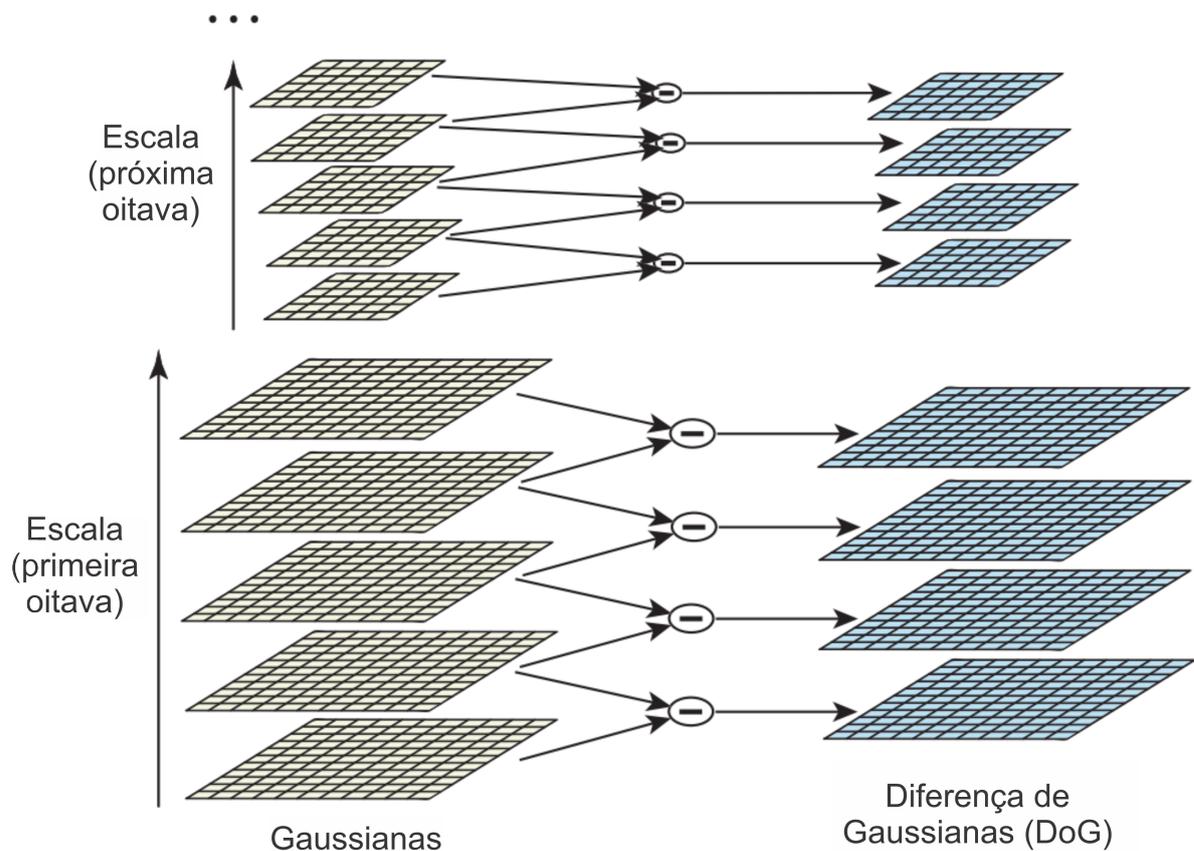


Figura 4 – Pirâmide escala-espaco da imagem estabelecida pelo algoritmo SIFT. Fonte: Adaptado de (LOWE et al., 1999).

Na Figura 4, ao lado esquerdo é possível observar os *kernels* gaussianos, enquanto ao lado direito é possível observar a Diferenças de Gaussianas (em inglês, Difference of Gaussian — DoG).

O espaço de escala de uma imagem  $L(x, y, \sigma)$  é definido pela Equação 2.1:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (2.1)$$

onde,  $I$  corresponde a uma imagem,  $(x, y)$  representam as coordenadas de localização e  $\sigma$  representa o desvio-padrão da função gaussiana  $G(x, y, \sigma)$ , tal qual:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}. \quad (2.2)$$

A partir do espaço de escala de uma imagem é gerado outro conjunto de imagens, através da função de Diferenças de Gaussianas, onde imagens DoG descobrem pontos-chave estáveis em diferentes escalas presentes na imagem. Portanto, a função de Diferenças de Gaussianas pode ser definida pela seguinte Equação 2.3:

$$D(x, y, \sigma) = L(x, y, \eta\sigma) - L(x, y, \sigma) \quad (2.3)$$

onde  $\eta$  representa uma constante de escala.

A partir deste momento, a Diferença de Gaussianas é usada para calcular as aproximações laplacianas de gaussianas. Portanto, um píxel em uma imagem é comparado com seus oito vizinhos, bem como nove pixels vizinhos na escala superior e nove pixels vizinhos na escala inferior. Para que um píxel torne-se candidato a ser um ponto-chave, é necessário que o mesmo seja o maior ou o menor da sua vizinhança como pode ser observado na Figura 5.

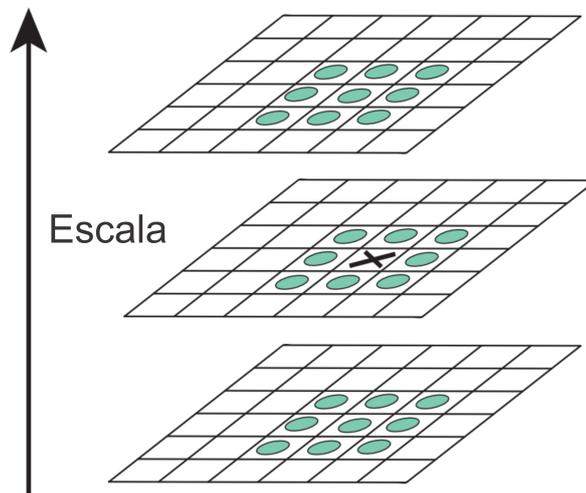


Figura 5 – Detecção de extremos no espaço de escala gaussiano estabelecida pelo algoritmo SIFT. Fonte: Adaptado de (LOWE et al., 1999).

Como observado na Figura 5,  $x$  representa a detecção de extremos no espaço de escala gaussiano com seus 26 vizinhos (representados por círculos verdes) em uma região  $3 \times 3$  da imagem atual e imagens adjacentes. Neste momento, é executada a etapa de localização exata dos pontos-chave. Portanto, nesta etapa são eliminados os pontos-chave

que possuem baixo contraste ou que estejam ao longo de uma borda. Esta etapa é executada com o auxílio da Expansão Quadrática de Taylor da DoG, expressa pela Equação 2.4:

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2} \mathbf{x} \quad (2.4)$$

onde  $\mathbf{x} = (x, y, \sigma)$ ,  $x$  representa a localização precisa do ponto-chave, calculada a partir da derivada de  $D(x)$  em relação a  $x$ , igualada a zero, conforme Equação 2.5:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2} \frac{\partial D}{\partial \mathbf{x}}. \quad (2.5)$$

Após estabelecidos os pontos-chave, é necessário definir as suas respectivas orientações e descritores, dada por meio da Equação 2.6:

$$\theta(x, y) = \tan^{-1} \left( \frac{L(x, y + 1) - L(x, y - 1)}{L(x + 1, y) - L(x - 1, y)} \right). \quad (2.6)$$

Com isto, é criado um histograma com os dados das orientações dos pixels da vizinhança de cada ponto-chave, contendo 36 intervalos de mesmo tamanho. Os componentes que compõem o histograma são ponderados pela magnitude  $m(x, y)$  dos gradientes, vide Equação 2.7:

$$m(x, y) = \sqrt{(L(x + 1, y) - L(x - 1, y))^2 + (L(x, y + 1) - L(x, y - 1))^2}. \quad (2.7)$$

Por fim, as orientações e magnitude de uma região de vizinhança de um ponto-chave formarão o descritor SIFT para aquele determinado ponto.

### 2.1.2 Speeded-up Robust Features (SURF)

O Descritor de Características Locais SURF (BAY; TUYTELAARS; GOOL, 2006) é um algoritmo de descrição de características locais inspirado no SIFT e inclui, além disto, seu próprio método de detecção de características locais. Assim como o SIFT, este algoritmo extrai diversas características invariantes presentes em imagens. Entretanto, apesar de ser inspirado e baseado nas mesmas etapas e princípios do SIFT, possui variações em sua formulação em cada etapa, além disto, é considerado um algoritmo com forte robustez e mais rápido que o SIFT.

Para realizar a etapa de detecção de pontos-chave, os autores propõem o detector *Fast-Hessian*, onde é demonstrado bons resultados como poder computacional e acurácia.

Desta forma, SURF se baseia no determinante da matriz Hessiana. Portanto, o determinante da matriz Hessiana de um dado píxel, conforme demonstrada como na Equação 2.8:

$$H(f(x, y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}. \quad (2.8)$$

Para que o determinante da matriz Hessiana seja adaptativo à escala, é necessário filtrar uma imagem  $I$  por um filtro gaussiano, desta forma, dado um ponto  $X = (x, y)$ , em uma imagem  $I$ , temos:

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \quad (2.9)$$

onde  $H$  é a matriz Hessiana em  $x$  na escala  $\sigma$ ,  $L_{xx}(\mathbf{x}, \sigma)$  é a convolução da derivada de segunda ordem gaussiana  $\frac{\partial^2}{\partial x^2}g(\sigma)$  com a imagem  $I$  no ponto  $x$ , enquanto  $L_{xy}(\mathbf{x}, \sigma)$  e  $L_{yy}(\mathbf{x}, \sigma)$  são semelhantes a  $L_{xy}(\mathbf{x}, \sigma)$  como pode ser observado na Figura 6.

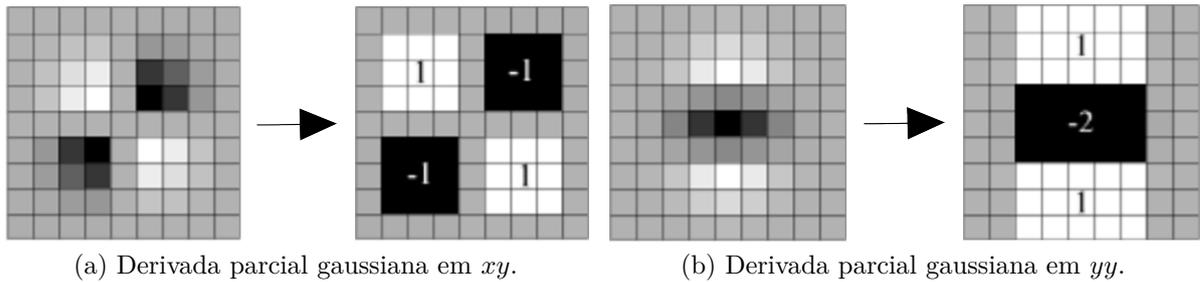


Figura 6 – Aproximações de núcleo Gaussiano estabelecida pelo algoritmo SURF. Fonte: Adaptado de (BAY; TUYTELAARS; GOOL, 2006).

Como observado na Figura 6, os filtros de tamanho  $9 \times 9$  representam aproximações para derivadas gaussianas de segunda ordem, enquanto os píxels cinza correspondem ao valor zero. A função gaussiana eleva o custo computacional do processo. Desta forma, a função gaussiana é aproximada por um núcleo laplaciano. Os núcleos gaussianos aproximados por laplacianos adquirem um custo computacional muito baixo, desta forma, podemos expressar o determinante da Hessiana (aproximada) como:

$$\det(\mathcal{H}_{\text{approx}}) = D_{xx}D_{yy} - (wD_{xy})^2 \quad (2.10)$$

onde a aproximação da função gaussiana por um núcleo laplaciano é representado por  $D$ . Desta forma, um píxel será considerado como sendo um ponto-chave se o determinante da Hessiana na sua vizinhança for maior que um determinado limiar.

A partir deste momento, para cada ponto-chave detectado é atribuído um vetor de características (descriptor). A definição dos descritores se dá pela atribuição de orientação

e descrição dos componentes. A atribuição de orientação torna o SURF invariante a rotação. Deste modo, para determinar a orientação, é necessário calcular *wavelets* de Haar (STANKOVIĆ; FALKOWSKI, 2003) na direção  $x$  e  $y$ , em uma vizinhança circular de raio  $6\sigma$  ao redor do ponto-chave. Neste caso, uma gaussiana  $\sigma = 2.5$  centrada no ponto-chave é usada para calcular e ponderar as *wavelets*.

Em seguida, é definida um conjunto de seis janelas deslizantes de tamanho  $\pi/3$  para calcular a soma das respostas da *wavelets* vertical e horizontal em uma área de varredura, de forma a encontrar a orientação definida pela janela deslizante que retorna o maior valor de soma de *wavelets*. Esta orientação torna-se a orientação principal do descritor de características.

Por fim, para definir o vetor de características, é determinado uma região quadrada de tamanho  $20\sigma$  centrada em torno do ponto-chave, ao mesmo tempo, em que é orientada por toda a extensão da orientação já obtida na etapa descrita acima. Em seguida, esta região é dividida em 16 sub-regiões quadradas de tamanho  $4 \times 4$ . Para cada sub-região, vinte e cinco pontos de amostra regularmente distribuídos, portanto, de tamanho  $5 \times 5$  são utilizados para calcular 16 *wavelets* de Haar, desta forma,  $d_x$  e  $d_y$  correspondem as respostas das *wavelets* de Haar nas direções horizontal e vertical respectivamente, onde o tamanho do filtro deve ser  $2\sigma$ .

Desta forma as respostas das *wavelets* de Haar  $d_x$  e  $d_y$  são então somadas em cada sub-região, além dos valores absolutos das respostas  $|d_x|$  e  $|d_y|$ . Portanto, cada sub-região possui um vetor  $v$  de 4 dimensões como pode ser visto abaixo:

$$\mathbf{v} = \left[ \sum d_x, \sum d_y, \left| \sum d_x \right|, \left| \sum d_y \right| \right]. \quad (2.11)$$

A partir disto, concatena-se os 16 vetores  $v_i$  de cada sub-região obtém-se um vetor  $d$  de 64 componentes, conforma a Equação 2.12:

$$\mathbf{d} = [v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}, v_{16}]. \quad (2.12)$$

Os Descritores de Características Locais SIFT e SURF são protegidos por patente; desta forma, não podem ser utilizados em aplicações de uso comercial gratuitamente.

### 2.1.3 KAZE (KAZE)

O descritor KAZE (ALCANTARILLA; BARTOLI; DAVISON, 2012) (“Vento” em japonês) é um algoritmo de descrição de características locais. Os autores propõem a detecção de extremos no espaço de escala de difusão não linear das imagens (diferente dos algoritmos SIFT e SURF, que realizam a detecção de extremos no espaço de escala

gaussiana) através da Filtragem de Difusão não Linear (em inglês, *Nonlinear Diffusion Filtering* — NDF).

A escala de difusão não linear apresenta bom desempenho em relação à escala gaussiana, de modo a preservar as principais características presentes em uma imagem, como cantos e arestas, aumentando significativamente a precisão da localização de características em comparação aos algoritmos SIFT e SURF. A Figura 7 apresenta a comparação entre as escalas gaussiana e difusão não linear.



Figura 7 – Comparação entre Escala Gaussiana e Escala de Difusão Não Linear. Fonte: Adaptado de (ALCANTARILLA; BARTOLI; DAVISON, 2012).

Como pôde ser observado na Figura 7, na primeira linha é apresentado o espaço de escala gaussiana (difusão linear). Enquanto na segunda linha é apresentado o espaço de escala de difusão não linear.

Desta forma, para produzir o espaço de escala de difusão não linear é utilizado a técnica de Divisão de Operador Aditivo (em inglês, *Additive Operator Splitting* — AOS) (WEICKERT; ROMENY; VIERGEVER, 1998), como pode ser visto na Equação 2.13:

$$L^{i+1} = \left( I - \tau \sum_{l=1}^m A_l(L^i) \right)^{-1} L^i. \quad (2.13)$$

Desta forma, é calculado a resposta do determinante normalizado em escala do Hessiano através do espaço de escala não linear para a detecção de pontos-chave, vide Equação 2.14:

$$L_{\text{Hessian}} = \sigma^2 \left( L_{xx}L_{yy} - L_{xy}^2 \right) \quad (2.14)$$

onde  $L_{xx}$  e  $L_{yy}$  são as derivadas horizontais e verticais de segunda ordem, respectivamente, e  $L_{xy}$  é o derivativo cruzado de segunda ordem. Desta forma, O conjunto de derivadas de primeira e segunda ordem são aproximados por meio de filtros *Scharr*  $3 \times 3$  de diferentes tamanhos de etapas de derivadas  $i$ . E, por fim, a posição do ponto-chave é calculada.

A partir deste momento, é possível realizar a descrição dos descritores. E para isto, os autores recorrem ao SURF modificado, conhecido como M-SURF, onde os autores adaptam para que opere de forma exitosa com o espaço de escala não linear do KAZE, onde, por fim, é gerado um vetor descritor de comprimento 64, normalizado-o em um vetor unitário tornando-se invariante a contraste.

#### 2.1.4 Binary Robust Independent Elementary Features (BRIEF)

O Descritor Binário Local denominado BRIEF (CALONDER et al., 2010; CALONDER et al., 2011) consiste em criar vetor de *bits* a partir do teste binário ( $\tau$ ) em uma vizinhança de pixels centrada em cada ponto-chave, onde  $\tau(p; x, y)$  é definido pela Equação 2.15:

$$\tau(p; x, y) = \begin{cases} 1 & : p(x) < p(y) \\ 0 & : p(x) \geq p(y) \end{cases} \quad (2.15)$$

onde  $p(x)$  é a intensidade de  $p$  em um ponto  $x$ . A escolha de um conjunto de pares de  $n(x, y)$  define, conseqüentemente, um conjunto de testes binários. Onde podemos definir  $n$  como sendo o comprimento do vetor de características binária (comumente, definido por 128, 256 ou 512 pares de pixels).

Considerando 256 pares de pixels ( $pk1, pk2$ ), com  $1 \leq k \leq 256$ , são escolhidos em uma vizinhança local de tamanho  $48 \times 48$  em torno de um ponto-chave. O  $k$ -ésimo *bit* do descritor terá valor 1 se  $pk1 < pk2$ , e valor 0 em caso contrário. Por fim, BRIEF pode ser expresso pela Equação 2.16:

$$f_{n_d}(\mathbf{p}) := \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(\mathbf{p}; \mathbf{x}_i, \mathbf{y}_i). \quad (2.16)$$

Portanto, BRIEF não possui um padrão de amostragem, desta forma, utiliza pares de amostragem aleatórios. A Figura 8 apresenta o padrão de 128 pares de amostragem do BRIEF.

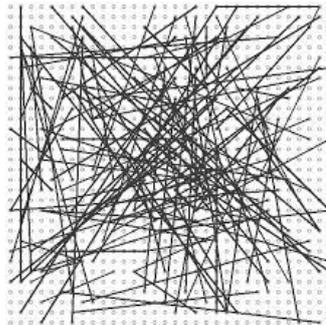


Figura 8 – Padrão de 128 pares de amostragem do algoritmo BRIEF. Fonte: Adaptado de (CALONDER et al., 2010).

### 2.1.5 Oriented FAST and Rotated BRIEF (ORB)

O Descritor Binário Local denominado ORB (RUBLEE et al., 2011) baseia-se na técnica Características do Teste Acelerado de Segmentos (em inglês, *Features from Accelerated Segment Test* — FAST) (ROSTEN; DRUMMOND, 2006) (não abordado neste trabalho) e no algoritmo BRIEF (apresentado anteriormente).

Portanto, através do algoritmo FAST são detectados e selecionados pontos-chave que contenham informações sobre a localização de bordas consideradas determinantes em uma imagem de entrada  $I$ , enquanto suas orientações são calculadas através da intensidade de centróide que assume que, para cada borda, sua intensidade é deslocada de seu centro como pode ser observado na Figura 9.

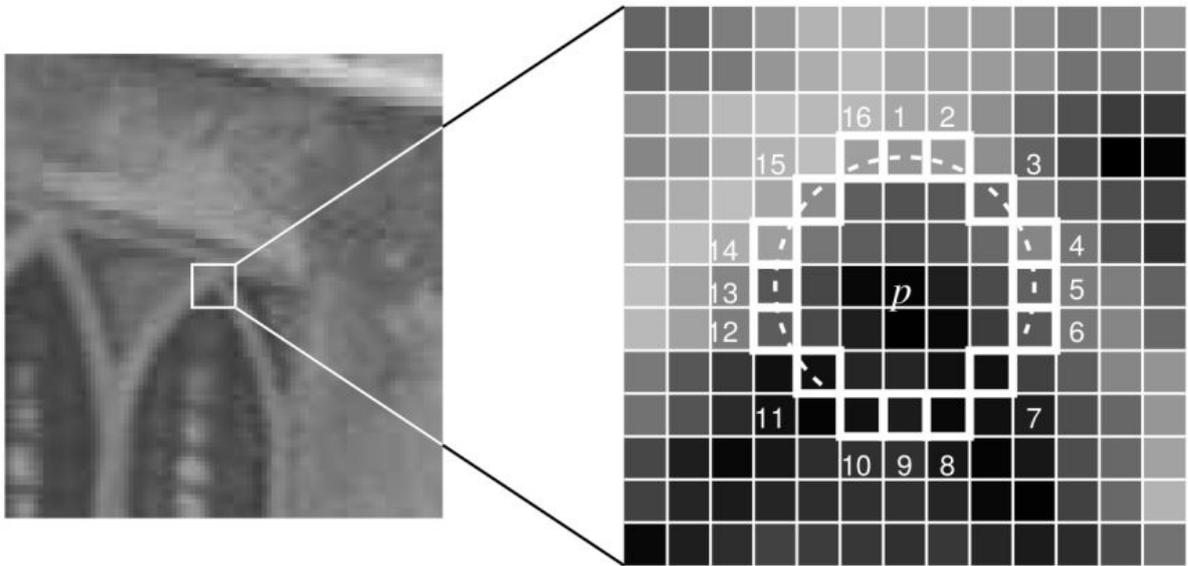


Figura 9 – Detecção de bordas estabelecida pelo algoritmo FAST. Fonte: (LOWE et al., 1999).

Portanto, este vetor pode ser tomado para imputar uma orientação, obtidas a partir do cálculo de momentos expressos pela Equação 2.17:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \quad (2.17)$$

onde os deslocamentos em relação à posição do ponto-chave nos eixos horizontal e vertical da imagem  $I$  são representados respectivamente por  $p$  e  $q$ , enquanto todas as coordenadas dos pontos dentro de uma certa vizinhança do ponto-chave são representados respectivamente por  $x$  e  $y$ . A partir deste momento, as coordenadas do centróide podem ser calculadas, vide Equação 2.18:

$$C = \left( \frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right). \quad (2.18)$$

Desta maneira, para um canto de centro  $O$ , é possível construir o vetor  $\overrightarrow{OC}$ , onde sua orientação é dada por:

$$\theta = \text{atan2}(m_{01}, m_{10}). \quad (2.19)$$

E por fim, desta maneira, é determinada a orientação do ponto-chave em questão pelo algoritmo ORB. Portanto, o padrão de amostragem do ORB usa pares aprendidos. A Figura 10 apresenta o padrão de amostragem do ORB.

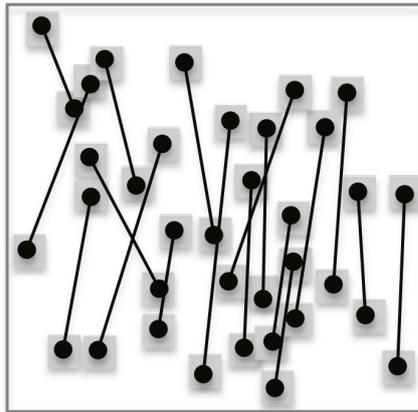


Figura 10 – Padrão de amostragem do algoritmo ORB. Fonte: (KRIG, 2014).

### 2.1.6 Binary Robust Invariant Scalable Keypoints (BRISK)

O Algoritmo denominado BRISK (LEUTENEGGER; CHLI; SIEGWART, 2011) utiliza como detector de pontos-chave o algoritmo chamado Teste de Segmento Acelerado Adaptativo e Genérico (em inglês, *Adaptive and Generic Accelerated Segment Test* — AGAST) (MAIR et al., 2010), sendo este uma versão com desempenho acelerado do algoritmo FAST.

Para a etapa de detecção de pontos-chave de modo a manter a invariância à escala, são estabelecidas camadas de pirâmide escala-espço da imagem. Deste modo, a pirâmide escala-espço consistem em 4 oitavas  $c_i$  e 4 intra-oitavas  $d_i$  para  $i = \{0, 1, \dots, n - 1\}$ . Desta forma, cada camada intra-oitava  $d_i$  encontra-se entre camadas  $c_i$  e  $c_i + 1$  de oitavas. Desta forma, a identificação de um ponto-chave na oitava  $c_i$  se dá pela análise dos 8 pontos vizinhos das oitavas inferiores e superiores. A partir das camadas de oitavas  $c_i$  um ponto-chave local é refinado, usado para ajustar o valor máximo da medida  $s$  ao longo do eixo da escala, com a finalidade de determinar na posição interpolada, a escala real do ponto-chave detectado. A medida  $s$  é determinada por meio do algoritmo FAST. Desta forma, é definida como sendo o limiar máximo onde um ponto-chave é ou não, considerando como sendo um canto, como pode ser observado na Figura 11.

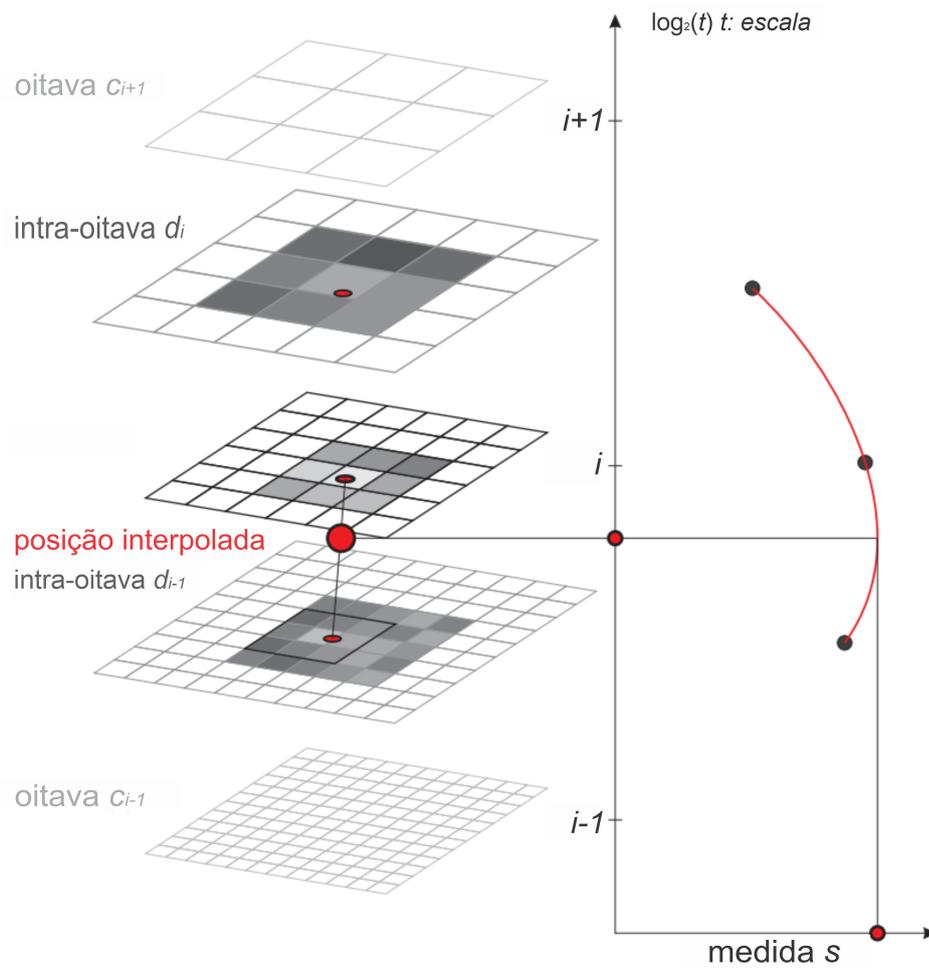


Figura 11 – Pirâmide escala-espaco da imagem estabelecida pelo algoritmo BRISK. Fonte: Adaptado de (LEUTENEGGER; CHLI; SIEGWART, 2011).

Já para a etapa de descrição de características, o descritor pode ser definido como um vetor de características binária resultante de comparações de intensidades entre pares de pontos. Deste modo, o descritor BRISK utiliza um padrão simétrico que possui pontos de amostragem posicionados em círculos concêntricos de tamanhos diferentes em torno do ponto-chave, como pode ser observado na Figura 12.

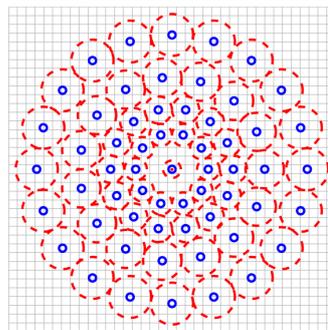


Figura 12 – Padrão de amostragem do algoritmo BRISK. Fonte: (LEUTENEGGER; CHLI; SIEGWART, 2011).

Além disto, o descritor BRISK estima a orientação do ponto-chave e a rotação do padrão de amostragem por meio da soma de todos os gradientes locais entre os pares, com isso, torna-o variante à rotação e escala.

### 2.1.7 Fast Retina Keypoint (FREAK)

O Descritor Binário Local FREAK (ALAHÍ; ORTIZ; VANDERGHEYNST, 2012), para a realização da primeira etapa de detecção de pontos-chave, sugere o uso de padrão de amostragem da retina, ou seja, circular; entretanto, com maior densidade de pontos próximos ao centro, a densidade de pontos cai exponencialmente. Desta forma, o descritor binário  $F$  é construído a partir de uma sequência de Diferença de Gaussianas de um *bit*, como pode ser visto na Equação 2.20:

$$F = \sum_{0 \leq a < N} 2^a T(P_a). \quad (2.20)$$

No qual  $P_a$  representa um par de campos receptivos e  $N$  representa o tamanho do descritor. Desta forma, podemos definir  $T(P_a)$  como:

$$T(P_a) = \begin{cases} 1 & \text{se } (I(P_a^{r_1}) - I(P_a^{r_2})) > 0 \\ 0 & \text{caso contrário,} \end{cases} \quad (2.21)$$

onde  $(P_a^{r_1})$  representa a intensidade do primeiro campo receptivo referente ao par  $P_a$ . A partir destes campos receptivos, é possível encontrar diversos pares, gerando um grande descritor. Um problema encontrado a partir deste ponto é que muitos destes pares encontrados podem ou não serem úteis de modo a descrever as principais características presentes em uma imagem de forma eficiente. Desta forma, é aplicado um algoritmo semelhante ao ORB com a finalidade de aprender os melhores pares a partir dos dados de treinamento.

Deste modo, é criada uma matriz  $D$  de pontos-chave extraídos, portanto, cada linha corresponde a um ponto-chave, depois, é calculado a média de cada coluna, seguida pela ordenação destas mesmas colunas conforme a variância mais alta, desta forma (para ter uma característica discriminante), permanecendo a melhor coluna (média de 0,5). É ainda adicionada de forma iterativa as colunas restantes que possuem baixa correlação com as colunas selecionadas. Com isto, é realizada a soma dos gradientes locais selecionando dos pares, com o objetivo de estimar a orientação do ponto-chave conforme Equação 2.7:

$$O = \frac{1}{M} \sum_{P_a \in G} (I(P_o^{r_1}) - I(P_o^{r_2})) \frac{P_o^{r_1} - P_o^{r_2}}{|0P_o^{r_1} - P_o^{r_2}|0} \quad (2.22)$$

onde,  $G$  constitui o conjunto de pares utilizados para calcular os gradientes locais,  $M$  representa o número de pares em  $G$ , enquanto  $P_o^{r_1}$  representa o vetor de coordenadas

espacial do centro do campo receptivo.

O padrão de amostragem do algoritmo FREAK é inspirado na retina do olho humano, desta forma, o descritor FREAK utiliza um padrão assimétrico que possui pontos de amostragem sobrepostos de tamanhos variados, aumentando a densidade de amostragem a medida que se aproxima em torno do ponto-chave como pode ser observado na Figura 13.

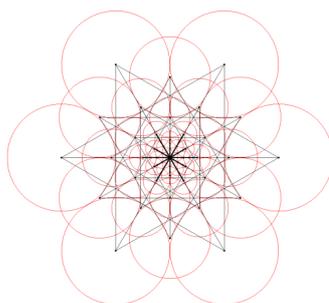


Figura 13 – Padrão de amostragem do algoritmo FREAK. Fonte: (ALAHÍ; ORTIZ; VANDERGHEYNST, 2012).

### 2.1.8 Accelerated KAZE (AKAZE)

O Descritor Binário Local denominado AKAZE é a versão acelerada do algoritmo KAZE apresentado por (ALCANTARILLA; SOLUTIONS, 2011). AKAZE também faz uso de um espaço de escala de difusão não linear assim como o KAZE, entretanto, utiliza como descritor, um descritor binário modificado e baseado no método de Diferença Binária Local (em inglês, *Local Difference Binary* — LDB) (YANG; CHENG, 2012) sendo um método mais rápido e eficiente ao ser comparado com o método de Divisão de Operador Aditivo utilizado pelo algoritmo KAZE.

O descritor denominado Diferença Binária Local Modificado (em inglês, *Modified-Local Difference Binary* — M-LDB), para a etapa de descrição de características sub-divide em regiões de  $2 \times 2$ ,  $3 \times 3$  e  $4 \times 4$  pixels a vizinhança local de cada ponto-chave, e a partir deste ponto, é realizado a comparação de valores médios de intensidade  $I$  dos pixels e derivadas  $dx$  e  $dy$  em cada sub-região, desta forma, definindo o descritor. O descritor AKAZE possui uma limitação onde só é possível utilizá-lo ao lado do seu próprio detector, ou ao lado do detector KAZE.

## 2.2 Saco de Características Visuais

A abordagem de Saco de Características Visuais, proposta originalmente por (CSURKA et al., 2004), é inspirada pela abordagem de Saco de Palavras (BLEI; NG; JORDAN, 2003) utilizada nas áreas de Processamento de Linguagem Natural (em inglês,

*Natural Language Processing* — NLP) e Recuperação de Informações (em inglês, *Information Retrieval* — IR) onde é aplicada, por exemplo, em categorização de textos, onde a classificação de um documento se dá pela frequência de palavras. Já a abordagem de Saco de Características Visuais, é caracterizada pela categorização de imagens, onde a classificação de uma imagem se dá pela frequência de características visuais, tornando-se uma abordagem, simples e de baixo custo computacional.

Para melhor compreender a abordagem Saco de Características Visuais com o classificador *Perceptron* Multicamadas, apresentaremos em detalhes as técnicas utilizadas em cada etapa da abordagem de Saco de Características Visuais, seguidas do comportamento do modelo *Perceptron* Multicamadas utilizado neste experimento e simulação preliminar. Portanto, a abordagem Saco de Características Visuais pode ser dividida em três etapas: (a) Etapa de representação de características; (b) Etapa de geração de vocabulário visual, e; (c) Etapa de representação de imagem. Podemos ver essas etapas em mais detalhes a seguir:

### 2.2.1 Representação de Características

Através dos descritores já apresentados (vide Seção 2.1), os pontos-chave serão detectados a partir de uma imagem de entrada (apresentada tanto nas amostras de treinamento quanto de teste em cada conjunto de dados visuais), e então as características de interesse (descritores) serão computados. Ao final desta operação, é obtido um vetor de características com dimensões número de pontos-chave  $\times$  número de descritores.

### 2.2.2 Geração de Vocabulário Visual

Agora, é necessário gerar o dicionário de vocabulário visual possível sobre a coleção de vetores de características. Para aprender o vocabulário visual, geralmente, é realizado um algoritmo de agrupamento. Para este experimento, decidimos usar o algoritmo de agrupamento K-Means (MACQUEEN et al., 1967). O agrupamento K-Means é um algoritmo de aprendizado não-supervisionado padrão usado para particionar amostras em  $k$  *clusters* distintos, agrupando amostras com características relevantes. Portanto, considerando um conjunto de dados visuais  $T$ , onde cada  $n$  pontos-chave é representado por um vetor real  $d$ -dimensional:

$$T = \{x_i\}_{i=1}^n. \quad (2.23)$$

O algoritmo de agrupamento K-Means visa particionar os  $n$  pontos-chave em  $k(\leq n)$  *clusters* distintos  $C = \{c_1, c_2, \dots, c_k\}$ . Portanto, para cada  $C_i$ , um centróide  $\mu_i$  está

associado a ele. Desta forma, podemos estabelecer  $\mu_i$  como a média da posição de todos os elementos do *cluster*, onde  $n_i$  é o número de elementos em cada *cluster*  $C_i$ , ou seja:

$$\mu_i = \frac{1}{n_i} \sum_{x_j \in C_i} x_j. \quad (2.24)$$

Como mencionado acima, o algoritmo de agrupamento K-Means visa particionar os  $n$  pontos-chave em  $k (\leq n)$  *clusters* distintos, onde a distribuição resultante é aquela em que cada ponto-chave está próximo dos outros dentro do mesmo *cluster*. Assim, assume-se que quanto mais próximos, mais semelhantes eles são. Em outras palavras, o algoritmo de agrupamento K-Means visa minimizar uma função objetivo, neste caso, a Soma dos Erros Quadrados (em inglês, *Sum Square Error* — SSE) da distância de cada ponto-chave em relação à sua distância aos centróides:

$$J = \sum_{j=1}^k \sum_{i=1}^n |0x_i^{(j)} - \mu_i|^2. \quad (2.25)$$

Na Equação 2.25,  $J$  representa a função objetivo,  $k$  o número de *clusters*,  $n$  o número de casos e  $|0x_i^{(j)} - \mu_i|^2$  a distância euclidiana entre um ponto-chave  $x_i$  e um centróide  $\mu_i$ . Com isso, o algoritmo de agrupamento K-Means retorna o centro (centróide) de cada grupo (*cluster*), e cada centro do *cluster* será tratado como o vocabulário do dicionário visual (atuando como uma característica visual).

### 2.2.3 Representação de Imagem

Finalmente, as frequências de cada *cluster* são calculadas (computadas tanto para o conjunto de treinamento quanto para o conjunto de teste), resultando em um Histograma de Características Visuais, nada menos que o Saco de Características Visuais. Finalmente, o histograma de características visuais pode ser usado posteriormente com um classificador (por exemplo, K-vizinhos mais próximos (em inglês, *K-Nearest Neighbors* — K-NN) (ZHANG; ZHOU, 2007; ZHANG; ZHOU, 2005), Máquina de Vetores de Suporte (em inglês, *Support Vector Machine* — SVM) (CORTES; VAPNIK, 1995; GROOT, 2012), Bayes Ingênuo (em inglês, *Naive Bayes*) (FRIEDMAN; GEIGER; GOLDSZMIDT, 1997; SAHAMI, 1996; LANGLEY et al., 1992) ou *Perceptron* Multicamadas). Neste trabalho, foi escolhido o *Perceptron* Multicamadas para ser utilizado como classificador. Na Figura 14, é possível observar o que acontece na abordagem Saco de Características Visuais de forma ilustrativa.

Onde, (a) representa as amostras de imagens apresentadas em um conjunto de dados visuais; (b) representa o Vocabulário Visual gerado pelo algoritmo de agrupamento K-Means; (c) representa as frequências de cada *cluster* calculadas, resultando em um Histograma de Características Visuais.

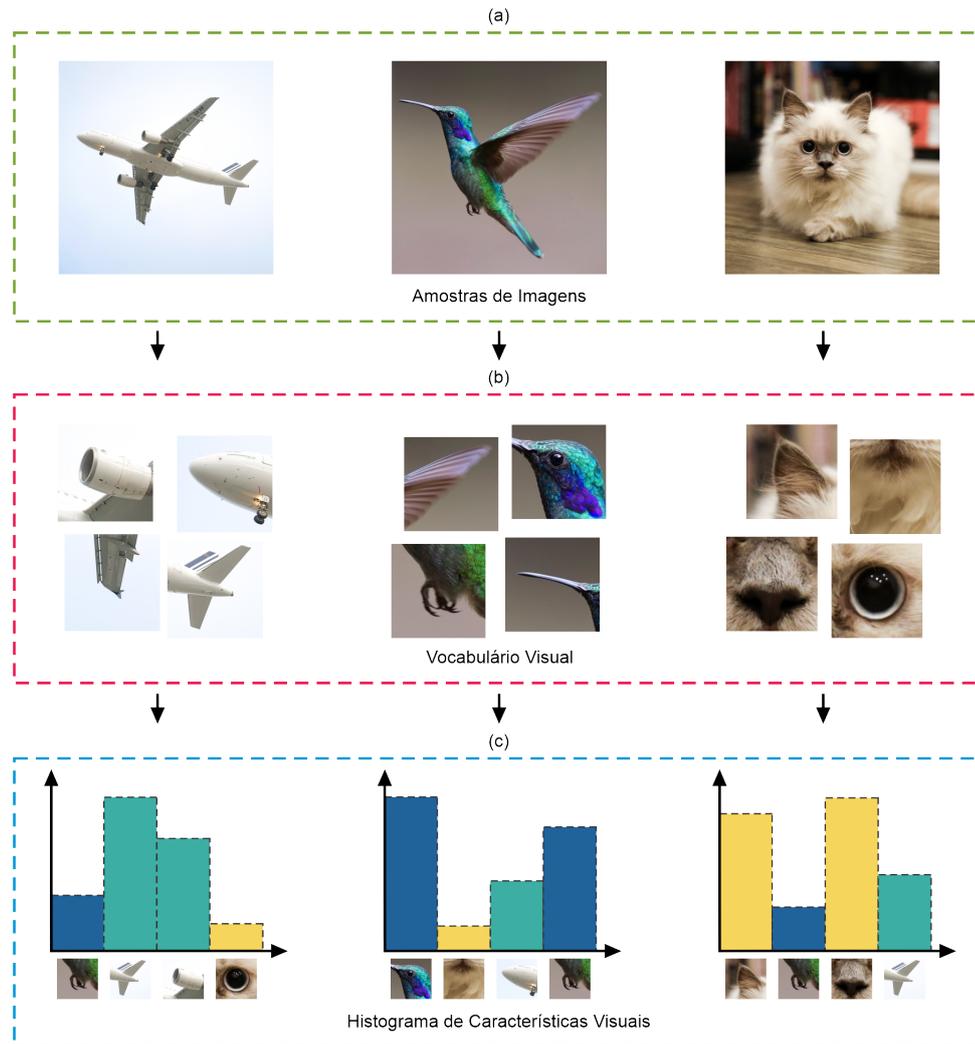


Figura 14 – Representação da abordagem Saco de Características Visuais. Fonte: Elaborado pela autora.

## 2.2.4 Perceptron Multicamadas

Com o modelo *Perceptron* Multicamadas usado neste trabalho, cada nó de entrada  $x_i$  ( $x_1, x_2, \dots, x_n$ ) está associado a um peso sináptico  $w_i$  ( $w_1, w_2, \dots, w_n$ ), onde o valor dos pesos sinápticos são inicialmente inicializados de forma aleatória. A entrada possui também um limiar de ativação  $b$  (*bias*) que é a representação de um valor fixo diferente de 0. Neste sentido, ocorre o processo de combinação linear, produzindo o potencial de ativação  $s$ , onde os nós de entrada são ponderados por seus respectivos pesos sinápticos associados ( $x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n$ ), e pelo limiar de ativação  $b$ , onde  $n$  representa o tamanho do vetor de entrada como pode ser observado na Equação 2.26:

$$s = \sum_{i=1}^n w_i x_i + b. \quad (2.26)$$

Após este processo, a soma das entradas ponderadas por seus pesos sinápticos associados mais o limiar de ativação, ou seja, o potencial de ativação  $s$  é submetido a

uma função de ativação  $f$  (a função de ativação mais utilizada em modelos *Perceptron* Multicamadas é a função sigmóide), como pode ser visto na Equação 2.27:

$$f(s) = \frac{1}{(1 + e^{-s})} \quad (2.27)$$

onde  $s$  representa o potencial de ativação. Logo, se o potencial de ativação  $s$  for maior que o limiar de ativação será considerado que o modelo *Perceptron* Multicamadas estará ativado (a saída  $y$  será igual à 1), e será considerado que o modelo *Perceptron* Multicamadas estará desativado, caso contrário. A arquitetura básica do modelo *Perceptron* Multicamadas com uma única camada escondida pode ser vista na Figura 15, onde uma arquitetura básica de *Perceptron* Multicamadas pode ser dividida nas camadas listadas abaixo:

- **Camada de Entrada** (ou em inglês, *Input Layer*): Onde o conjunto de dados é inserido na rede;
- **Camada Escondida** (ou em inglês, *Hidden Layer*): É nesta camada onde ocorre a maioria dos processamentos da rede;
- **Camada de Saída** (ou em inglês, *Output Layer*): Onde o resultado obtido através do processamento realizado na camada escondida é exibido.

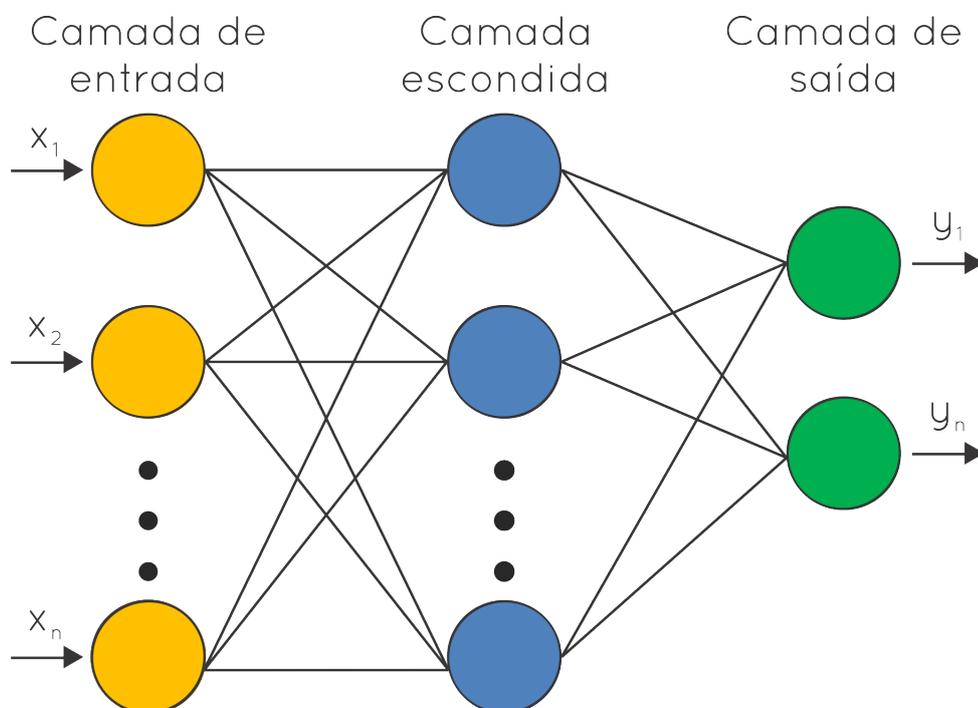


Figura 15 – Arquitetura básica do modelo *Perceptron* Multicamadas. Fonte: Adaptado de (BEALE; JACKSON, 1990).

Vale ressaltar que arquiteturas profundas de uma rede *Perceptron* Multicamadas, possuem  $n$  camadas escondidas.

Para um melhor entendimento, a Figura 16 mostra um fluxograma da abordagem Saco de Características Visuais com classificador *Perceptron* Multicamadas para a etapa de treinamento.

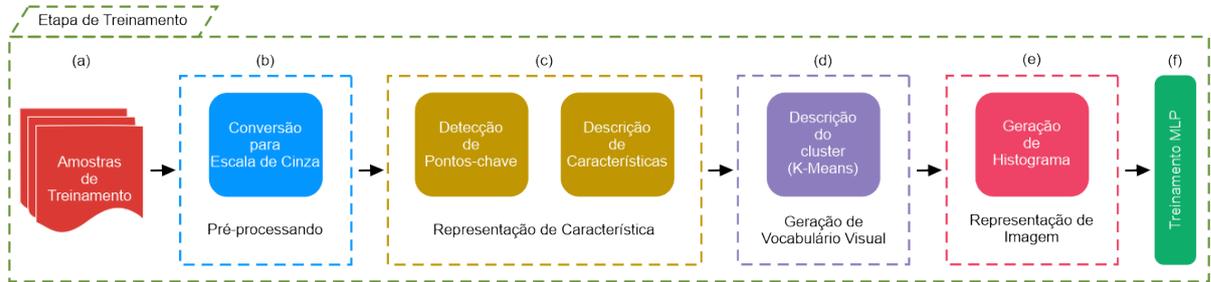


Figura 16 – Abordagem Saco de Características Visuais com classificador *Perceptron* Multicamadas para etapa de treinamento. Fonte: Elaborado pela autora.

Onde (a) representa as amostras de treinamento apresentadas em um conjunto de dados visuais; (b) representa a etapa de pré-processamento; (c) representa a etapa de Representação de Característica, onde os pontos-chave são detectados e descritos; (d) representa a etapa de Geração de Vocabulário Visual, onde as descrições dos pontos-chave computados na etapa anterior são agrupadas para a geração de Vocabulário Visual; (e) representa a etapa de Representação de Imagem, ou seja, é gerado um histograma de frequência de características visuais; e (f) representa a etapa de treinamento de *Perceptron* Multicamadas, onde o histograma de frequência gerado pela etapa anterior, é utilizado como um vetor de características para treinamento.

Em comparação, a Figura 17 mostra um fluxograma da abordagem Saco de Características Visuais com classificador *Perceptron* Multicamadas para a etapa de teste.

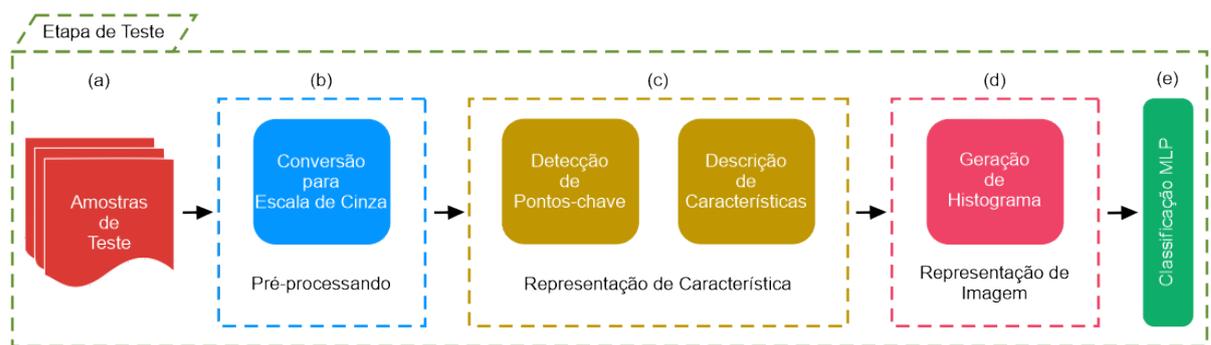


Figura 17 – Abordagem Saco de Características Visuais com classificador *Perceptron* Multicamadas para etapa de teste. Fonte: Elaborado pela autora.

Onde (a) representa as amostras de teste apresentadas em um conjunto de dados visuais; (b) representa a etapa de pré-processamento; (c) representa a etapa de Representação de Característica, onde os pontos-chave são detectados e descritos; (d) representa a etapa de Representação de Imagem, ou seja, é gerado um histograma de frequência de características visuais; e (e) representa a etapa de classificação de *Perceptron* Multicamadas,

onde o histograma de frequência gerado pela etapa anterior é utilizado como um vetor de características para classificação e/ou recuperação da imagem.

A principal diferença entre as Figuras 16 e 17, consiste na etapa (d) Geração de Vocabulário Visual presente na Figura 16. Isto porque, a etapa (d) Geração de Vocabulário Visual é executada apenas na etapa de treinamento, enquanto a etapa de teste, aplica ao conjunto de teste o Vocabulário Visual gerado previamente pela etapa de treinamento.

## 2.3 Redes Neurais Convolucionais

A arquitetura de Rede Neural Convolutacional proposta originalmente por (LECUN et al., 1998), a partir do ano de 2012 vêm conquistando espaço em desafios de reconhecimento e classificação de imagens como na competição *Imagenet* (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Hoje, temos a disposição, uma gama de diferentes arquiteturas de Redes Neurais Convolucionais, para citar algumas: LeNet (LECUN et al., 1998), AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), VGG (HE et al., 2016), GoogLeNet (SZEGEDY et al., 2015), ResNet (HE et al., 2016), Xception (CHOLLET, 2017), e seus resultados as tornam estado da arte para a resolução de tarefas na área de Visão Computacional, tarefas relacionadas a detecção de objetos (GIRSHICK et al., 2014), reconhecimento de faces (PARKHI et al., 2015), classificação de vídeo (KARPATHY et al., 2014), segmentação (LONG; SHELHAMER; DARRELL, 2015), estimativa de pose humana (TOSHEV; SZEGEDY, 2014), rastreamento de objetos (WANG; YEUNG, 2013), entre outras. Ao treinar tais arquiteturas, um problema ainda enfrentado está relacionado ao poder computacional necessário, onde, muitas vezes torna-se um recurso indisponível e/ou até mesmo caro. De modo a minimizar o problema de poder computacional, temos como objetivo reformular a camada convolutacional de nosso modelo através de um Descritor Binário Local.

Apesar da semelhança com a arquitetura *Perceptron* Multicamadas, a principal diferença está na operação que ocorre nas imagens inseridas na rede. Portanto, a Figura 18 mostra a arquitetura básica do modelo de Rede Neural Convolutacional, que pode ser dividida em:

- **Camada de Entrada** (ou em inglês, *Input Layer*): onde o conjunto de dados é inserido na rede (i.e., base de imagens). Após, o conjunto de dados visuais é passado para a;
- **Camada de Convolução** (ou em inglês, *Convolutional Layer*): o objetivo desta camada consiste em extrair características de imagens de entrada, e é neste momento que ocorre a principal operação nas imagens inseridas na rede — a convolução. O

resultado da convolução é comumente conhecido como um mapa de características (em inglês, *feature maps*) que é posteriormente passado para a;

- **Camada ReLU** (ou em inglês, *ReLU Layer*): a camada ReLU (em inglês, *Rectified Linear Units*) é utilizada para que a rede possa convergir mais rápido e aplica para as camadas escondidas da rede uma função de ativação não linear à saída  $x$  da camada anterior. Logo, a função de ativação ReLU contribui em uma etapa de treinamento da rede mais rápida para arquiteturas profundas. O conjunto é então passado para a;
- **Camada de Subamostragem** (ou em inglês, *Pooling Layer*): reduz a dimensionalidade dos *feature maps* que serão utilizados pelas camadas seguintes, entretanto, sem perder as informações mais relevantes. Após, é passado para a;
- **Camada de *Flatten*** (ou em inglês, *Flatten Layer*): a camada *Flatten* possui o objetivo de “achatar” todo o volume de dados 3D em um único vetor 1D de características para posteriormente servir de entrada para a camada totalmente conectada. E finalmente a;
- **Camada totalmente conectada** (ou em inglês, *Fully Connected Layer*): essa camada age como um classificador, se localiza ao final da rede. Geralmente é do tipo *soft-max* usado para classificar a imagem de entrada. Realiza tal tarefa ao reconhecer padrões que foram gerados pelas camadas anteriores. Portanto, estima a probabilidade de quais das  $n$  classes a imagem de entrada pertence.

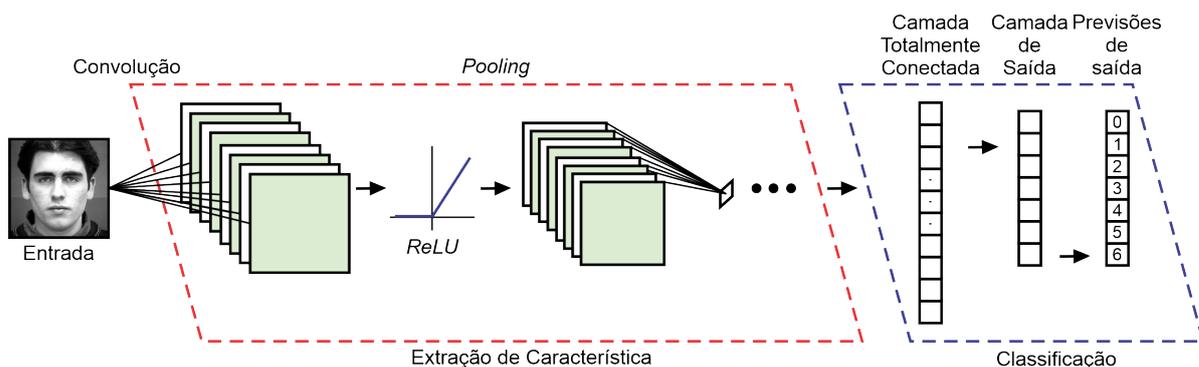


Figura 18 – Arquitetura básica do modelo de Rede Neural Convolutacional. Fonte: Adaptado de (PEEMEN; MESMAN; CORPORAAAL, 2011).

Como visto na Figura 18, para este exemplo, as  $n$  classes são representadas por seis expressões faciais básicas (raiva, nojo, medo, alegria, tristeza, surpresa), e neutro, formando sete emoções (sete classes). Desta forma, às sete emoções (sete classes) são representadas pelo modelo por valores numéricos inicializados a partir do valor zero, portanto, a emoção neutra, pode ser expressa pelo valor numérico seis como pode ser observado na Figura 18.

## 2.4 Redes Neurais Recorrentes

Baseada no trabalho de (RUMELHART; HINTON; WILLIAMS, 1985), uma Rede Neural Recorrente é um modelo de sequência neural, isto é, um tipo de Rede Neural Artificial projetada para trabalhar com dados sequenciais de entrada. Tais dados, podem ser sequências de texto, caligrafia, linguagem falada, séries temporais e genomas, dentre outras. Portanto, este tipo de modelo possui uma dimensão temporal, considerando tempo e sequência. Podemos trabalhar com dados sequenciais de entrada e reconhecer os padrões presentes nestes dados por meio das Redes Neurais Recorrentes, pois este tipo de modelo possui *loops* em sua arquitetura, os quais permitem que as informações que transitam pela rede persistam, ou seja, os *loops* possibilitam que as informações sejam passadas de etapa a etapa para um sucessor na rede. Na Figura 19 é possível observar a representação de um *loop* em uma arquitetura de Rede Neural Recorrente.

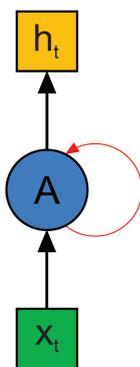


Figura 19 – Representação de um *loop* em uma arquitetura de Rede Neural Recorrente. Fonte: Adaptado de (OLAH, 2015).

Podemos considerar uma arquitetura de Rede Neural Recorrente como sendo várias cópias de si mesma, em outras palavras, várias cópias de seu *loop*, passando, portanto, uma informação para o seu sucessor. Podemos visualizar tal comportamento se “desenrolarmos” o *loop* de uma arquitetura, ilustrado pela Figura 20.

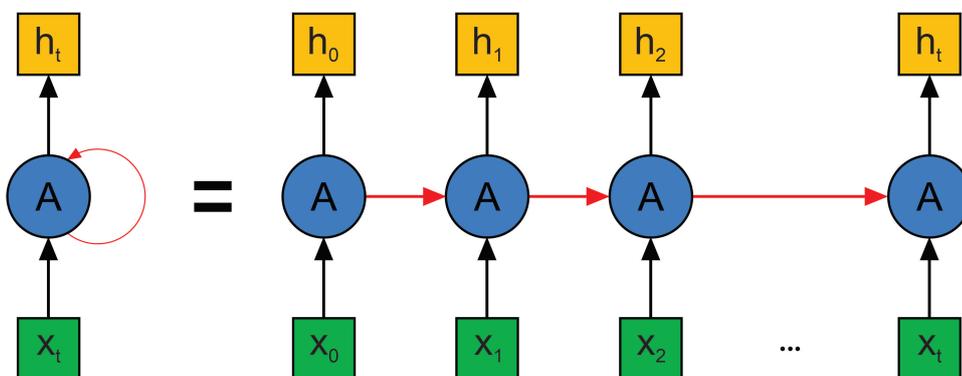


Figura 20 – Representação de uma arquitetura de Rede Neural Recorrente desenrolada. Fonte: Adaptado de (OLAH, 2015).

Visualizando esta representação, podemos inferir que este tipo de Rede Neural Artificial é o modelo certo para se trabalhar com dados sequenciais de entrada, portanto, dados de entrada que estão intimamente relacionados a listas ou sequências. Com pequenas diferenças entre as informações relevantes e os pontos necessários (tarefa atual), as Redes Neurais Recorrentes operam bem, e são capazes de aprenderem a usar as informações passadas, como pode ser visto na Figura 21.

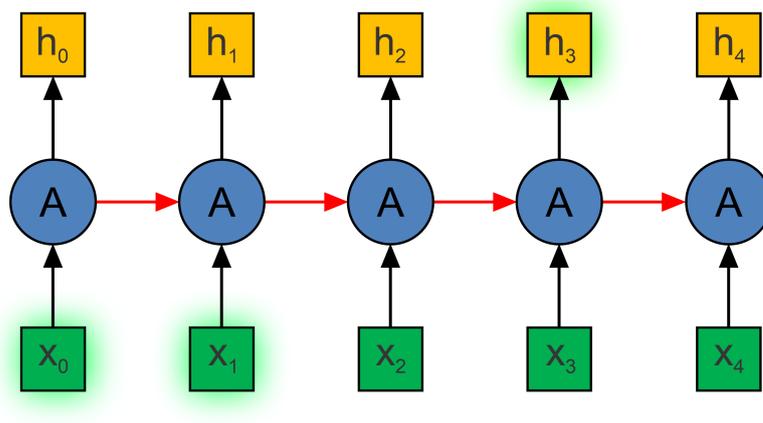


Figura 21 – Representação de eficiência com dependências de curto prazo de uma arquitetura de Rede Neural Recorrente. Fonte: Adaptado de (OLAH, 2015).

No entanto, em alguns casos, é necessário um histórico maior ou, dependendo do cenário, é necessário mais contexto. À medida que a lacuna entre informações relevantes e o ponto em que se faz necessário o seu uso aumenta, este modelo torna-se ineficiente, pois vai perdendo a sua capacidade de aprender a conectar informações, como ilustra a Figura 22.

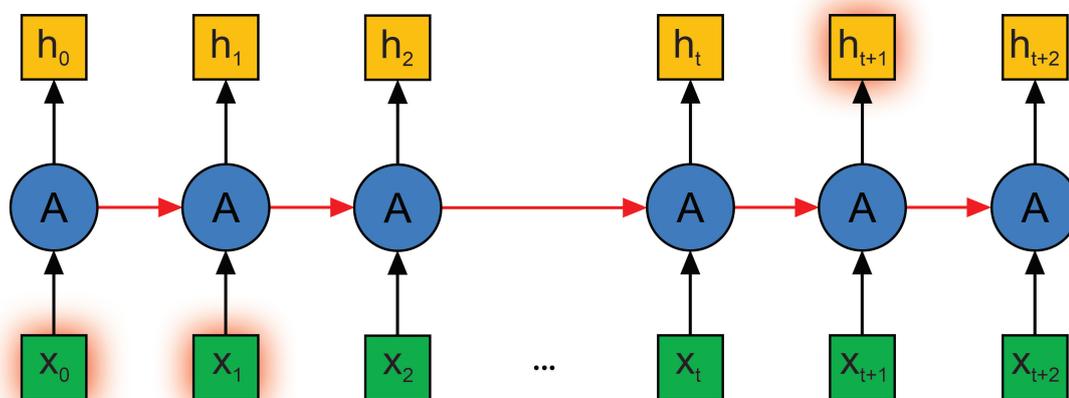


Figura 22 – Representação de ineficiência com dependências de longo prazo de uma arquitetura de Rede Neural Recorrente. Fonte: Adaptado de (OLAH, 2015).

Apesar de seus excelentes resultados nas duas últimas décadas, no início da década de 1990 alguns pesquisadores (HOCHREITER, 1991; BENGIO; SIMARD; FRASCONI, 1994) já apontavam que as Redes Neurais Recorrentes não eram tão eficientes para lidar com dependências de longo prazo.

### 2.4.1 Rede de Memória Longa de Curto Prazo

A arquitetura de maior sucesso em relação às Redes Neurais Recorrentes é a Rede de Memória Longa de Curto Prazo (em inglês, *Long Short-Term Memory* — LSTM) proposta originalmente por (HOCHREITER; SCHMIDHUBER, 1997). Esta arquitetura de Rede Neural Recorrente é capaz de lidar com dependências de longo prazo, pois foi desenvolvida com a finalidade de evitar tal problema, possuindo a capacidade de lembrar informações relevantes por um extenso intervalo de tempo. A Figura 23 mostra um bloco do modelo de Rede de Memória Longa de Curto Prazo tradicional.

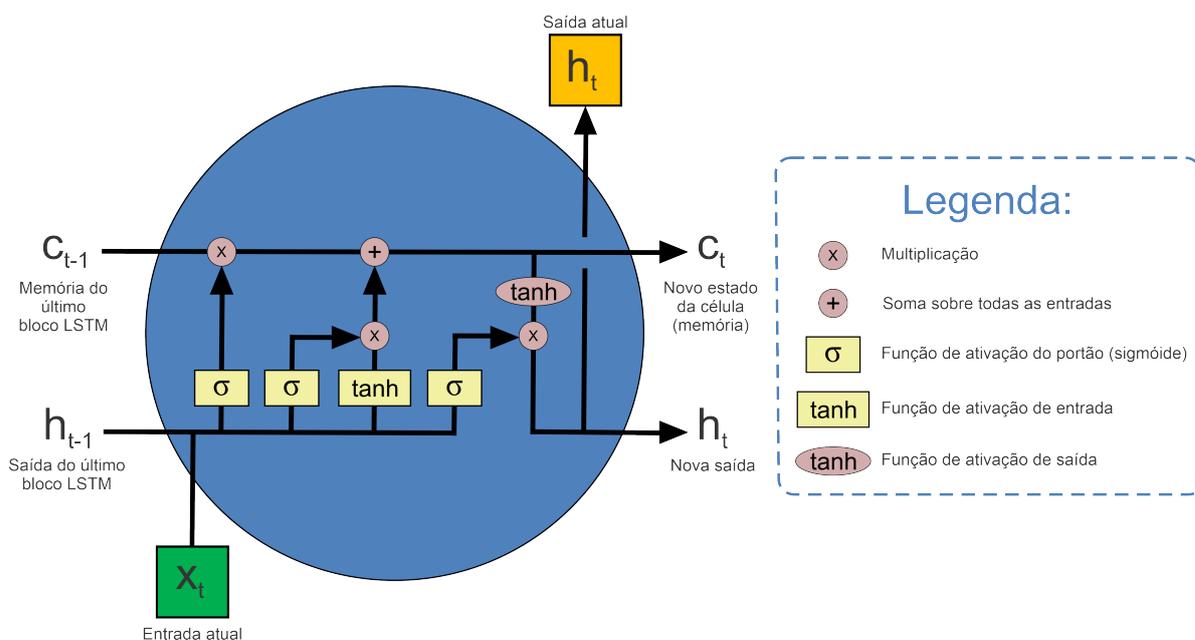


Figura 23 – Bloco do modelo de Rede de Memória Longa de Curto Prazo tradicional.  
Fonte: Adaptado de (GREFF et al., 2016).

Como já mencionado, na Figura 23 é possível observar um bloco Rede de Memória Longa de Curto Prazo tradicional e de que ele é composto. A saber: é composto por uma **Célula** (em inglês, *Cell*): a célula é considerada como sendo a unidade de memória do bloco Rede de Memória Longa de Curto Prazo, portanto, responsável por “lembrar” dos elementos presentes na sequência do conjunto de dados inserido na rede. O bloco Rede de Memória Longa de Curto Prazo tradicional é composto também, por três portões (em inglês, *Gates*) responsáveis pela manipulação e regularização de memória, ou seja, manipulação e regularização do fluxo de informações, tanto para dentro, quanto para fora da célula, onde pode ser dividida em:

- **Portão de Entrada** (em inglês, *Input Gate*): é responsável por controlar a inserção de informações relevantes para o estado da célula;
- **Portão de saída** (ou em inglês, *Output Gate*): é responsável por controlar a extração de informações relevantes presentes no estado atual da célula para ser posteriormente, apresentada como saída e, apresentada também como entrada da próxima célula;
- **Portão de Esquecer** (ou em inglês, *Forget Gate*): é responsável por controlar o descarte de informações que não são mais relevantes, ou seja, que se tornaram inúteis no estado da célula.

## 2.5 Modelo Híbrido de Arquitetura de Rede Neural Artificial

Ao trabalhar com Redes Neurais Convolucionais, supõe-se primariamente que os dados de entrada sejam imagens, isto porque as Redes Neurais Convolucionais foram desenvolvidas especificamente para mapear os dados de entrada (imagens) para uma variável de saída. Já as Redes Neurais Recorrentes foram desenvolvidas para trabalhar com dados de entrada de sequência (mas sucesso com textos) desta forma, este tipo de Rede Neural Artificial não é apropriada para que receba como dados de entrada, imagens. Portanto, a arquitetura básica do modelo híbrido de arquitetura de Rede Neural Artificial, pode ser observado na Figura 24.

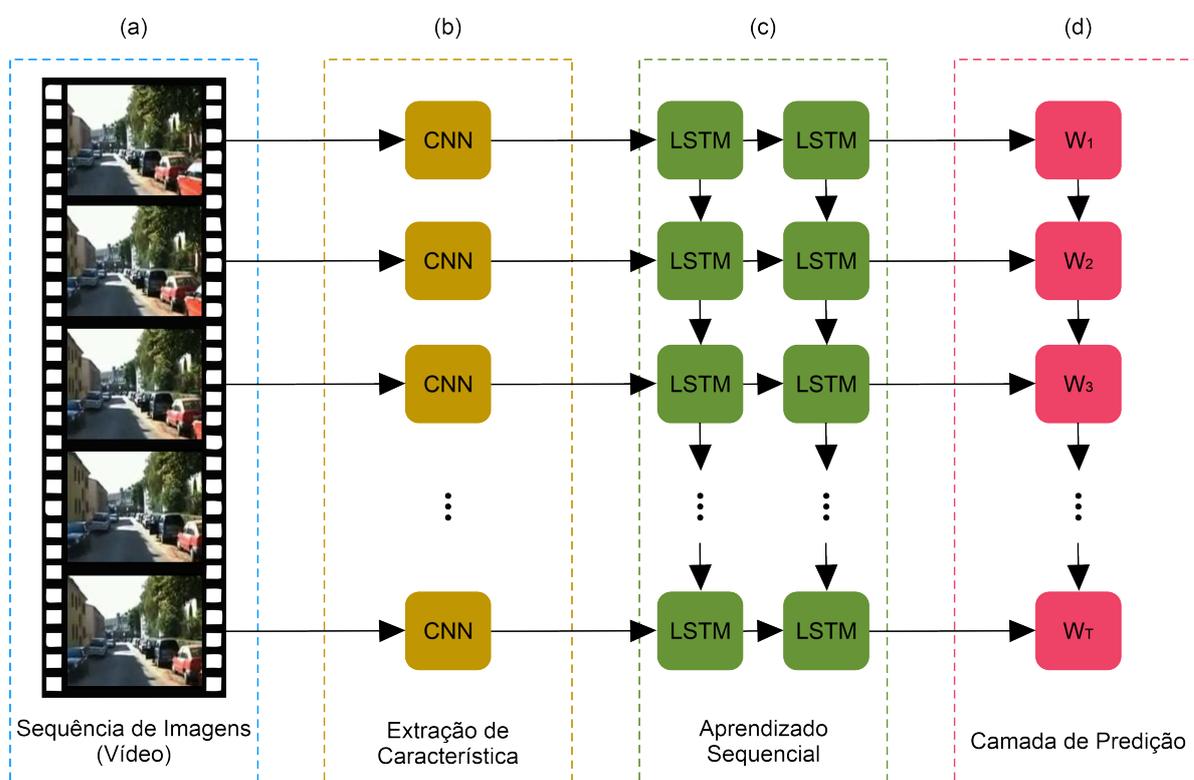


Figura 24 – Arquitetura básica do modelo híbrido de arquitetura de Rede Neural Artificial. Fonte: Adaptado de (DONAHUE et al., 2015).

Um modelo híbrido de arquitetura de Rede Neural Artificial, consiste em camadas de Rede Neural Convolutiva que realizam a extração de características nos dados de entrada (imagens), e que, combinados com Rede de Memória Longa de Curto Prazo, são capazes de suportar previsões de sequência (e.g., imagens sequenciais (vídeo)). Este processo descrito acima pôde ser observado na Figura 24. Originalmente, proposta por (DONAHUE et al., 2015), este modelo foi nomeado de Rede Convolutiva Recorrente de Longo Prazo, entretanto, como já mencionado na Seção 1.1, utilizaremos seu nome mais genérico e difundido na literatura — “*CNN LSTM*”.

Por fim, neste Capítulo, foi apresentado os conceitos básicos e teóricos relacionados a técnicas de detecção e descrição de características em imagens que dão sustentação para o desenvolvimento desta Dissertação. A seguir, será apresentada uma visão geral dos principais trabalhos analisados que se relacionam com o desenvolvimento desta Dissertação.

### 3 REVISÃO DE LITERATURA

Uma das etapas significativas da metodologia proposta nesta Dissertação, consiste no reconhecimento de pontos-chave em imagens. A eficiência entre estes algoritmos pode ser vista em diversos trabalhos realizado pela comunidade científica, onde foram aplicadas diversas análises e avaliações (PATEL et al., 2014; CHATOUX; LECCELLIER; FERNANDEZ-MALOIGNE, 2016; BAYRAKTAR; BOYRAZ, 2017; TAREEN; SALEEM, 2018).

Em destaque, em (HEINLY; DUNN; FRAHM, 2012) é apresentado a avaliação de desempenho entre diferentes Descritores Binários Locais, além de diferentes pares de detectores e descritores. Desta forma, os autores avaliam a performance entre três descritores, a saber: BRIEF, ORB e BRISK, enquanto utiliza, como linha de base, os descritores SIFT e SURF. BEKELE; TEUTSCH; SCHUCHERT apresentam uma avaliação dos Descritores Binários Locais, a saber: BRIEF, ORB, BRISK e FREAK. Os resultados obtidos mostram que o BRISK é o descritor de ponto-chave que fornece o maior número de melhores correspondências, além da maior porcentagem de precisão em comparação a todos os Descritores Binários Locais. Após o estudo da revisão de literatura e análise comparativa entre pares de detectores e descritores, é possível observar na Tabela 2 um resumo em relação à invariância assumida por cada descritor:

Tabela 2 – Invariância dos Descritores. Fonte: Elaborado pela autora.

Algoritmo	Detector	Descritor	Rotação	Escala	Luminosidade	Ponto de Vista
SIFT	✓	✓	✓	✓	✓	✓
SURF	✓	✓	✓	✓	✓	✓
KAZE	✓	✓	✓	✓	x	x
BRIEF	x	✓	x	x	✓	x
ORB	✓	✓	✓	x	✓	x
BRISK	✓	✓	✓	✓	✓	✓
FREAK	x	✓	✓	✓	✓	x
AKAZE	✓	✓	✓	✓	x	x

Legenda

Sim	Não
✓	x

Outro ponto significativo é a busca por redução de dimensionalidade, aplicada em algoritmos de detectores e descritores. Em vista disto, existe uma gama de estudos realizados, em sua maioria para solucionar o problema de Recuperação de Imagens com Base em Conteúdo (em inglês, *Content-Based Image Retrieval* — CBIR) (ZHUO; CHENG; ZHANG, 2014; TARAWNEH et al., 2020) e ORB-PCA apresentado por VINAY et al. para a tarefa de Reconhecimento Facial. Este tipo de estudo nos motiva a realizar a redução de dimensionalidade aplicada ao Descritor Binário Local escolhido, de modo a gerar um novo espaço de características de menor dimensão produzido pelo Descritor Binário Local.

Há ainda pesquisas recentes que indicam que recursos baseados em arquitetura de Rede Neural Convolutiva superam os recursos artesanais para representação de imagens de forma significativa explorada em Robótica. Em (DAI et al., 2019), os autores fornecem um estudo comparativo de três classes distintas de Descritores de Características Locais, são eles: recursos artesanais, Redes Neurais Convolucionais treinadas e Redes Neurais Convolucionais pré-treinadas para avaliar sua eficiência para a tarefa de correspondência de pontos-chave em aplicações de Robótica, considerando a capacidade dos descritores em lidarem com alterações condicionais (e.g., iluminação e ponto de vista). Os resultados finais do estudo comparativo realizado constata que: (a) Os descritores artesanais em comparação a qualquer método de descritores baseados em Redes Neurais Convolucionais, não são competitivos; (b) Referente a mudanças de ponto de vista, os métodos de descritores baseados em Redes Neurais Convolucionais treinadas têm um desempenho superior em relação aos métodos de descritores baseados em Redes Neurais Convolucionais pré-treinadas; (c) Em contrapartida, em relação a mudanças de iluminação, os métodos de descritores baseados em Redes Neurais Convolucionais pré-treinadas têm um desempenho superior aos métodos de descritores baseados em Redes Neurais Convolucionais treinadas; entretanto, consomem muita memória.

Na metodologia proposta nesta Dissertação, outra etapa significativa é a extração de características. Pretendemos realizar a extração de características com um método de descritores baseados em Redes Neurais Convolucionais treinadas devido aos bons resultados observados em (DAI et al., 2019). Portanto, nesta Dissertação, propomos a formulação das camadas convolucionais de Redes Neurais Convolucionais tradicionais de nossa abordagem por meio de Descritores Binários Locais. A ideia de combinar essas duas técnicas não é nova, pois ANWER et al. apresenta o modelo TEX-Nets, composto por um algoritmo de Padrão Binário Local (em inglês, *Local Binary Pattern* — LBP) que codifica modelos de Redes Neurais Convolucionais para resolver o dilema em aprender a descrição de textura robusta em arquiteturas de Aprendizado Profundo para reconhecimento de textura e tarefas de sensoriamento remoto, assim como classificação de cena. As conclusões demonstram que o modelo TEX-Nets em comparação com as técnicas de última geração para classificação de cena de sensoriamento remoto direciona a resultados aprimorados.

ONO et al. introduziu uma nova arquitetura profunda, a LF-Net, uma técnica de correspondência esparsa com o objetivo de aprender características locais. No final dos experimentos, é demonstrado que o modelo LF-Net supera as técnicas de última geração em relação à correspondência de características esparsos. No entanto, BARROSO-LAGUNA et al. propõe o modelo Key.Net, uma abordagem inovadora para tarefas de detecção de características que vinculam recursos artesanais e filtros aprendidos baseados em Redes Neurais Convolucionais. Com isto, para detectar pontos-chave em uma faixa de escalas, uma função de perda é usada para otimizar as propriedades principais dos pontos-chave respectivamente detectados por meio de uma camada de proposta de índice multiescala. Os resultados mostram que a abordagem apresentada excede os detectores de última geração em muitos termos, como desempenho e complexidade. WEI et al. apresenta o modelo LBP Dual-Channel CNN, uma nova técnica de classificação que combina Redes Neurais Convolucionais de canal duplo e um algoritmo de Padrão Binário Local para executar um amplo uso da capacidade de extração de características de Redes Neurais Convolucionais, ao mesmo tempo, em que usa do poder de discriminação de características do algoritmo de Padrão Binário Local. O autor mostra que os experimentos comparativos entre o modelo LBP Dual-Channel CNN e as técnicas de classificação de imagens hiperespectrais espaciais de última geração aumentam efetivamente a precisão da classificação, de fato, entre pequenas amostras marcadas para a etapa de treinamento.

Em qualquer caso, de modo a gerar um modelo com desempenho computacional eficiente, e que consuma menos poder computacional em comparação a outras arquiteturas de Redes Neurais Convolucionais, optou-se em reformular os descritores computados em filtros convolucionais com natureza esparsa e binária de camadas convolucionais da arquitetura Rede Neural Convolucional proposta. Este tipo de reformulação é viável, pois JUEFEI-XU; BODDETI; SAVVIDES apresentam a Rede Neural Convolucional Binária Local (em inglês, *Local Binary Convolutional Neural Networks* — LBCNN) que se baseia nos princípios do algoritmo de Padrão Binário Local. Nesta abordagem, é desenvolvida uma camada de Convolução Binária Local (em inglês, *Local Binary Convolution* — LBC), uma poderosa alternativa às camadas convolucionais. Desta forma, além de reduzir cálculos, e significativamente a quantidade de parâmetros aprendíveis durante a etapa de treinamento devido a sua natureza binária e esparsa, a camada de Convolução Binária Local reduz a complexidade do modelo; conseqüentemente, acarretando economias computacionais e requisitos de memória, tornando-se um modelo aplicável a ambientes reais que possuem recursos escassos e limitados.

O treinamento com pesos binários tem sido o tópico principal em muitos trabalhos (SOUDRY; HUBARA; MEIR, 2014; HWANG; SUNG, 2014; KIM; HWANG; SUNG, 2014), e a ideia de trabalhar com Redes Neurais Binarizadas (em inglês, *Binarized Neural Networks* — BNN) não é uma abordagem totalmente nova. Em destaque, COURBARI-AUX; BENGIO; DAVID mostram que é possível treinar modelos profundos com pesos

binários e ativações em tempo de execução com o modelo *BinaryConnect* apresentado. Os resultados dos experimentos demonstram que o modelo *BinaryConnect* apresentado pode atingir resultados do estado da arte conjuntos de dados visuais competitivos, como MNIST (LECUN et al., 1998), CIFAR-10 (KRIZHEVSKY; HINTON et al., 2009) e SVHN (NETZER et al., 2011).

HUBARA et al. nos apresenta uma Rede Neural Artificial com pesos binários e ativações em tempo de execução que obteve bons resultados quanto ao baixo consumo de: (a) consumo de energia e; (b) velocidade de computação. Portanto, as Redes Neurais Binarizadas substituem a maioria das operações aritméticas por operações *bit a bit*. RASTEGARI et al. apresenta duas abordagens diferentes para uma Rede Neural Convolutiva: a primeira chamada *Binary-Weight-Networks*, que usa filtros binarizados, e a segunda chamada XNOR, onde, tanto os filtros quanto a entrada para as camadas convolucionais são de carácter binário. Essas duas abordagens foram desenvolvidas porque uma Rede Neural Convolutiva tradicional geralmente requer uma Unidade de Processamento Gráfico (em inglês, *Graphics Processing Unit* — GPU), muita memória e tempo, dificultando o uso em dispositivos menores e que normalmente possuem recursos computacionais escassos. Ao aproximar os pesos e entradas dos números binários, o processamento com eles usa apenas a operação de adição e subtração, removendo a multiplicação, bem como as operações de contagem de *bits*. Com isso, foi possível obter resultados que requerem 32 vezes menos memória em ambos, bem como 52 vezes menos tempo no XNOR, e nem mesmo usar GPU nesse caso — apenas uma Unidade Central de Processamento (em inglês, *Central Process Unit* — CPU).

No que diz respeito a filtros esparsos, LIU et al. mostrou uma forma de reduzir a grande complexidade e custo de computação envolvidos em seu treinamento e processamento, evitando uma indesejável perda de precisão. Ao usar uma decomposição esparsa para expressar as etapas de filtragem em uma Rede Neural Convolutiva, é possível reduzir bastante a redundância, por meio da eliminação de parâmetros que não influenciam tanto nos resultados. Foi possível atingir uma dispersão de mais de 90% dos parâmetros, enquanto perdia apenas cerca de 1% de precisão. Com isso, uma Rede Neural Convolutiva esparsa poderia funcionar efetivamente quase quatro vezes mais rápido do que uma Rede Neural Convolutiva tradicional.

Outra etapa importante em nossa metodologia consiste no problema de Detecção de Fechamento de *Loop*. Quando falamos no problema de Detecção de Fechamento de *Loop* — uma das etapas mais significativas de um sistema VSLAM, ZHANG; SU; ZHU aponta que as principais técnicas e métodos considerados estado da arte para a tarefa de fechamento de *loop*, utilizam recursos artesanais e Saco de Palavras Visuais (SIVIC; ZISSERMAN, 2003; FILLIAT, 2007; GÁLVEZ-LÓPEZ; TARDOS, 2012). Apesar da comunidade científica explorar no campo da Robótica Móvel e VSLAM o estudo envolvendo a utilização de

técnicas e métodos de Aprendizado de Máquina baseados em Aprendizado Profundo, bem como recursos baseados em arquiteturas de Redes Neurais Convolucionais (como foi apresentado acima), quando se tratando do problema de Detecção de Fechamento de *Loop* para sistemas VSLAM, estas técnicas e métodos não são totalmente exploradas. Entretanto, é possível destacar o trabalho de ZHANG; SU; ZHU, que aborda o problema de Detecção de Fechamento de *Loop* para sistemas SLAM visuais baseado em Aprendizado Profundo, neste trabalho, os autores apresentam uma nova abordagem para a solução do problema de Detecção de Fechamento de *Loop* para sistemas VSLAM baseado em Rede Neural Convolucional.

SERMANET et al. propõe um modelo de Rede Neural Convolucional pré-treinado nomeado de *OverFeat*, usado para gerar descritores de imagens inteiras (e.g., vetores de alta dimensão), desta forma, os vetores são então processados para construir representações dos locais; portanto, é calculado a distância entre os vetores de características da Rede Neural Convolucional de diferentes quadros sendo definido a pontuação de similaridade com base nos dados processados e deste modo, é calculado a Matriz de Similaridade para detectar *loops*. Os resultados indicam que o modelo *OverFeat* é viável para a Detecção de Fechamento de *Loop* e fornece uma maneira alternativa para sistemas VSLAM, visto que o método apresentado supera o *Fab-Map* a uma taxa de *recall* mais alta. Desta forma, propomos um método de Detecção de Fechamento de *Loop* baseado em uma adaptação de um modelo híbrido de arquitetura de Rede Neural Artificial, entre a Rede Convolucional Recorrente de Longo Prazo com blocos DescNet a ser explorado em trabalho futuro.

Outro passo vital presente na metodologia proposta nesta Dissertação, e que, como já comentado, será melhor explorado nas próximas etapas deste trabalho, consiste na integração do sistema proposto (vide Seção 7.2) com um sistema VSLAM de uma plataforma robótica móvel. Foi determinado, o uso como hardware, do microcomputador Jetson Nano da NVIDIA (conforme detalhado na Seção 5.1). Desde o seu lançamento em 2019, há um interesse crescente pela comunidade científica na avaliação e uso da Jetson Nano para sistemas VSLAM (PENG et al., 2019), além de outras tarefas interessantes e pertinentes para o desenvolvimento deste trabalho, como a abordagem de Correspondência de Modelo (em inglês, *Template Matching* — TM) (BASULTO-LANTSOVA et al., 2020) e reconhecimento e classificação com a arquitetura da Rede Neural Convolucional (CHEN et al., 2019).

Em ROSA et al. os autores comparam a performance do sistema VSLAM ORB-SLAM (MUR-ARTAL; MONTIEL; TARDOS, 2015) em diferentes microcomputadores; a saber: Raspberry Pi 3B+, Jetson Nano e Zybo Zynq-7000 com o objetivo de avaliar se o sistema VSLAM ORB-SLAM é capaz de ser executado em tempo real, com a interação de um ambiente desconhecido, através de sensores limitados e equipamentos portáteis. VIJITKUNSAWAT; CHANTNGARM provam que a CNN-LSTM foi executada tão bem

como um modelo de carro autônomo usando o microcomputador Jetson Nano da NVIDIA. Ao comparar três técnicas de Aprendizado de Máquina: Máquina de Vetores de Suporte, ANN-MLP e CNN-LSTM, os resultados demonstram que, apesar da eficiência de todas as técnicas caírem gradualmente ao adicionar mais obstáculos e níveis de velocidade, a CNN-LSTM supera as outras técnicas aplicadas. Dessa forma, estamos motivados a usar o microcomputador Jetson Nano da NVIDIA como hardware do sistema proposto.

### 3.1 Comparativo entre Trabalhos Relacionados e Abordagem Proposta

Para compilar as características dos principais trabalhos aqui relacionados, a seguir, é possível observar uma Tabela comparativa. Portanto, a Tabela 3 faz referências aos principais trabalhos identificados em relação aos métodos utilizados pela abordagem proposta nesta Dissertação.

A principal diferença entre os métodos elencados diz respeito a complexidade e diferenças entre os cenários experimentais dos trabalhos aqui relacionados apresentados, e a abordagem proposta nesta Dissertação. Por fim, após o estudo da revisão de literatura e análise comparativa, não foram encontrados trabalhos para a solução do problema de Detecção de Fechamento de *Loop* para sistemas VSLAM baseado em um modelo híbrido de arquitetura de Rede Neural Artificial que possuem filtros convolucionais esparsos e binários. Desta forma, acreditamos ser este um dos diferenciais desta Dissertação.

Tabela 3 – Comparação entre métodos utilizados pelos principais trabalhos relacionados.  
 Fonte: Elaborado pela autora.

Métodos	Trabalhos												
	ORB-PCA (VINAY et al., 2015)	TEX-Nets (ANWER et al., 2018)	LF-Net (ONO et al., 2018)	Key.Net (BARROSO-LAGUNA et al., 2019)	LBP Dual-Channel CNN (WEI et al., 2019)	LBCNN (JUEFEI-XU; BODDETI; SAVVIDES, 2017)	BinaryConnect (COURBARIAUX; BENGIO; DAVID, 2015)	BNN-Binary-Weight (HUBARA et al., 2016)	Binary-Weight e XNOR (RASTEGARI et al., 2016)	SCNN (LIU et al., 2015)	LCD OverFeat-based (ZHANG; SU; ZHU, 2017)	Abordagem Proposta	
Aprendizado Profundo	x	✓	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Pesos Binários	x	x	x	x	x	✓	✓	✓	✓	x	x	✓	
Pesos Esparsos	x	x	x	x	x	✓	x	x	x	✓	x	✓	
Extração de Características baseada em Descritores	✓	✓	✓	✓	✓	✓	x	x	x	x	x	✓	
Redução de Dimensionalidade	✓	x	x	x	x	x	x	x	x	x	x	✓	
Detecção de Fechamento de <i>Loop</i>	x	x	x	x	x	x	x	x	x	x	✓	✓	

Legenda

<b>Sim</b>	<b>Não</b>
✓	x

## 3.2 Comentários Gerais

Como já apontado, o estudo envolvendo a utilização de técnicas e métodos de Aprendizado de Máquina baseados em Aprendizado Profundo não são totalmente exploradas pela comunidade científica quando se tratando do problema de Detecção de Fechamento de *Loop* para sistemas VSLAM. Portanto, vemos uma oportunidade com o desenvolvimento desta Dissertação, de apresentar uma reformulação de filtros convolucionais de uma Rede Neural Convolucional tradicional através de Descritores Binários Locais para produzir um modelo esparsos e binário como uma alternativa viável a Redes Neurais Convolucionais tradicionais proporcionando bons resultados no processo de extração de características e padrões de classificação em imagens.

Além disto, pretendemos nas próximas etapas deste trabalho apresentar (conforme detalhado na Seção 7.2) uma nova abordagem para a solução do problema de Detecção de Fechamento de *Loop* para sistemas VSLAM baseado em um modelo híbrido de arquitetura de Rede Neural Artificial proposta. Acreditamos que, através da apresentação dos conceitos básicos e teóricos, e da apresentação desta revisão de literatura, conseguimos, sustentar e fundamentar esta Dissertação. A seguir, será apresentada a formulação do problema proposto, bem como a solução proposta desenvolvida nesta Dissertação.

## 4 FORMULAÇÃO DO PROBLEMA E SOLUÇÃO PROPOSTA

Neste Capítulo, é apresentado a formulação do problema e a solução proposta. Portanto, passaremos por alguns tópicos, a saber:

- **Problema Proposto:** É caracterizado nesta Seção o problema proposto, bem como, os resultados esperados;
- **Solução Proposta:** Nesta Seção é apresentada a Rede Neural Convolutiva de Descritores proposta.

### 4.1 Problema Proposto

Como descrito no Capítulo 1, as arquiteturas de Redes Neurais Convolucionais vêm conquistando espaço em desafios de reconhecimento e classificação de imagens, proporcionando bons resultados no processo de extração de características e padrões de classificação em imagens. Todavia, um problema ainda enfrentado ao treinar tais arquiteturas, está relacionado ao poder computacional necessário, que torna-se um recurso caro ou até mesmo indisponível. Em vista disto, o principal problema abordado nesta Dissertação consiste no reconhecimento e classificação de imagens através de um modelo de Rede Neural Convolutiva, composto por filtros convolucionais esparsos e binários, onde a principal característica do modelo consiste em reproduzir economias em relação ao número de parâmetros aprendíveis e, conseqüentemente, economia computacional significativa tornando-se um modelo aplicável em ambientes reais com recursos escassos e limitados. Para isto, apresentamos uma nova abordagem, onde é proposto a reformulação das camadas convolucionais por meio de Descritores Binários Locais.

Sob esta ótica, como também descrito nos Capítulos 1 e 3, para a Detecção de Fechamento de *Loop* a maioria das técnicas e métodos de VSLAM são baseados em Saco de Palavras Visuais. Recursos de Aprendizado Profundo, como, por exemplo, Redes Neurais Convolucionais, têm sido apenas recentemente explorados em sistemas VSLAM. Assim, apresentamos ainda, a possível integração do modelo proposto, com uma plataforma robótica autônoma em um ambiente real para a solução do problema de Detecção de Fechamento de *Loop* de um sistema VSLAM, nas próximas etapas deste trabalho.

#### 4.1.1 Resultados Esperados

- Reformulação de filtros convolucionais de uma Rede Neural Convolutiva tradicional através de Descritores Binários Locais para produzir um modelo esparsos e binário

como uma alternativa viável a Redes Neurais Convolucionais tradicionais;

- Obtenção de resultados satisfatórios (e.g., precisão de classificação, consumo computacional, entre outros) sobre a eficiência e acurácia do sistema proposto.

## 4.2 Solução Proposta

Foram investigadas técnicas e métodos que pudessem ser utilizadas para o desenvolvimento da reformulação de filtros convolucionais de uma arquitetura de Rede Neural Convolutional tradicional através de um Descritor de Características Locais ou Descritor Binário Local, e conseqüentemente, utilizadas também na implementação da adaptação de Rede Neural Convolutional, apresentando o novo *design* de camadas para a arquitetura proposta.

### 4.2.1 Rede Neural Convolutional de Descritores

Desta forma, apresentamos duas novas camadas para arquiteturas profundas, a camada de Detecção de Características Locais, tratada primariamente com primeira camada, responsável pela (a) Detecção dos pontos-chave no conjunto de dados de treinamento, e (b) seguida pela Descrição dos descritores, gerando um vetor de características. E como alternativa à camada convolucionacional de uma Rede Neural Convolutional tradicional, propomos a camada de Convolução de Descritor Local. Descritores Binários Locais inspiram os fundamentos do *design* de camadas das camadas de Detecção de Características Locais e Convolução de Descritor Local. Desta forma, a camada de Convolução de Descritor Local consiste em (a) redução da dimensionalidade por meio do algoritmo Aprendizado de Dicionário (em inglês, *Dictionary Learning* — DL) (TOSIC; FROSSARD, 2011) aplicado ao vetor de características gerado pela camada de Detecção de Características Locais anterior, reduzindo a dimensionalidade do vetor de características e tornando-o esparso; (b) seguida por um processo de binarização, neste caso, a função Sinal, transformando a natureza do vetor de características em esparso e binário; (c) seguida ainda, por uma transformação do vetor de características em uma matriz quadrada.

A partir deste processo, é então realizado a transformação do vetor de características computado em filtros convolucionais, chamados neste trabalho de pesos do descritor (em inglês *descriptor-weights*), utilizados como *kernel* na operação de convolução desta camada para que a reformulação proposta atinja os mesmos objetivos das Redes Neurais Convolutionais tradicionais. Experimentalmente, demonstramos a viabilidade de produzir filtros convolucionais através de Descritores Binários Locais similares a filtros convolucionais tradicionais. Portanto, empiricamente, podemos supor que em comparação com uma camada convolucionacional de uma Rede Neural Convolutional tradicional, Redes Neurais Convolutionais com camadas de Detecção de Características Locais e Convolução de

Descritor Local, denominada Rede Neural Convolutacional de Descritores — DescNet, possui potencial em reduzir a quantidade de parâmetros aprendíveis de uma Rede Neural Convolutacional, o que, conseqüentemente, promove economias computacionais significativas.

Para melhor compreensão da nova abordagem para explorar filtros convolutacionais esparsos e binários com Redes Neurais Convolutacionais com camadas de Detecção de Características Locais e Convolução de Descritor Local, apresentamos a seguir em detalhes as técnicas utilizadas em cada etapa da reformulação proposta, seguida pelo comportamento da camada convolutacional neste experimento.

#### 4.2.1.1 Detecção de Características Locais

A primeira camada da arquitetura proposta é a camada de Detecção de Características Locais, responsável por executar a (a) detecção de pontos-chave  $Nl$  no conjunto de dados de treinamento. Na Figura 25 é possível observar a representação da detecção de pontos-chave para uma imagem  $I$  utilizando a biblioteca *OpenCV*.

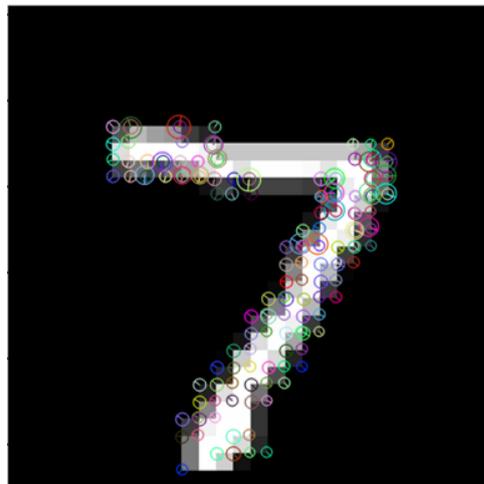


Figura 25 – Representação da detecção de pontos-chave utilizando a biblioteca *OpenCV*.  
Fonte: Elaborado pela autora.

Como observado na Figura 25, 150 pontos-chave foram detectados na imagem  $I$ . Já na Figura 26, é possível observar a representação de apenas um ponto-chave detectado utilizando a biblioteca *OpenCV*.

<KeyPoint 00000251CD92CEA0>

Figura 26 – Representação de um ponto-chave utilizando a biblioteca *OpenCV*. Fonte:  
Elaborado pela autora.

Em seguida, (b) os descritores são calculados, portanto, para cada imagem,  $N$  pontos-chave serão detectados e para cada ponto-chave,  $M$  descritores serão calculados, com isso; (c) é gerando um vetor de características  $F$ , e agora, podemos dizer que o vetor de características tem como dimensão o tamanho:  $F = (N, M)$  para cada imagem.

No que tange a imagem  $I$  apresentada na Figura 25, para cada 150 pontos-chave detectados, 64 descritores foram calculados, desta forma o vetor de características  $F$  é composto por 150 pontos-chave (dimensões), onde para cada ponto-chave existem 64 descritores (elementos) calculados. Por fim, a Figura 27 mostra o fluxograma da camada de Detecção de Características Locais do modelo de Rede Neural Convolutiva de Descritores.

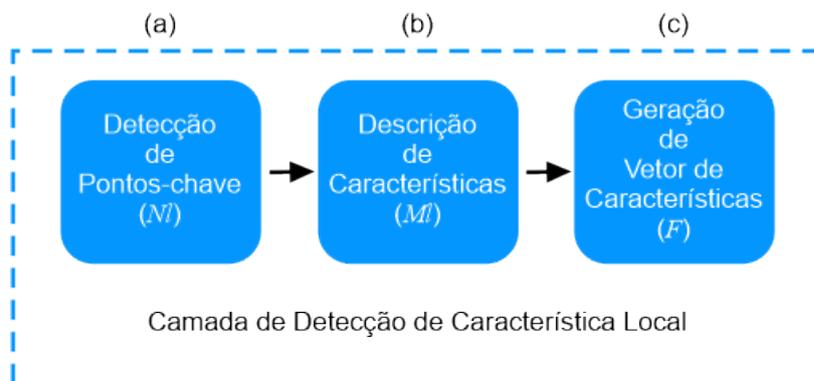


Figura 27 – Fluxograma da camada de Detecção de Características Locais. Fonte: Elaborado pela autora.

#### 4.2.1.2 Convolução de Descritor Local

E como alternativa à camada convolutiva de uma Rede Neural Convolutiva tradicional, propomos a camada de Convolução de Descritor Local. Portanto, quando se trata de descritores computados, procuramos sua representação relativamente diluída; isto é, por sua representação dimensional inferior e preservando suas características ou informações primárias ao mesmo tempo, sem perder qualidade. Portanto, transformaremos dados densos em dados esparsos para reduzir o número de dimensões a uma quantidade razoável, visto que o número de características é extremamente alto. Conseqüentemente, é necessário reduzir de alguma forma a dimensionalidade do vetor de características gerada pelos descritores. Por outro lado, reduzir o tamanho do vetor de características significa reduzir o poder distintivo do vetor de características.

Desde modo, optamos por usar uma técnica de aprendizado não-supervisionado conhecida como Redução de Dimensionalidade. A Redução de Dimensionalidade reduz o número total de características em um conjunto de características usando estratégias de Seleção ou Extração de Características. Existem muitos algoritmos de Redução de Dimensionalidade disponíveis, técnicas já estabelecidas na literatura, como Análise de Componentes Principais (em inglês, *Principal component analysis* — PCA) (HOTELLING, 1933), Análise Discriminante Linear de Fisher (em inglês, *Fisher's Linear Discriminant Analysis* — FLDA) (MCLACHLAN, 2004), Análise Discriminante Local de Fisher (em inglês, *Local Fisher's Discriminant Analysis* — LFDA) (SUGIYAMA, 2007), Mapeamento Isométrico (em inglês, *Isometric Mapping* — ISOMAP) (TENENBAUM; SILVA; LANGFORD, 2000) e Incorporação Linear Local (em inglês, *Locally Linear Embedding* — LLE)

(ROWEIS; SAUL, 2000). Após uma análise cuidadosa dos algoritmos de Redução de Dimensionalidade, optou-se por realizar a Redução de Dimensionalidade neste trabalho com base no Aprendizado de Dicionário. Aprendizado de Dicionário é uma técnica que permite a reconstrução de uma amostra de um dicionário. Portanto, o modelo possui um dicionário esparso (um conjunto de “átomos”) que pode ser melhor empregado para representar dados usando código esparso. E é a partir deste momento que o modelo passa a ter um dicionário esparso gerado pelo descritor de natureza reduzida.

Além de trabalhar com um vetor de características esparso, também determinamos que o vetor de características esparso precisa ser binarizado. Uma binarização simples é alcançada, produzindo um vetor de características de natureza esparsa e binária, onde esperamos que o modelo possa economizar memória e recursos computacionais.

Não queremos criar filtros de tamanho  $1 \times n$  em relação aos descritores computados. Queremos transformá-lo em uma matriz quadrada de alguma forma, já que o peso do filtro geralmente é uma matriz quadrada da ordem  $n$ . Portanto, uma simples transformação do vetor de características em uma matriz quadrada é feita. Com isso, os pesos do descritor do modelo são gerados e podem ser utilizados como *kernel* na operação de convolução desta camada.

Podemos resumir este processo da seguinte maneira: a camada de Convolução de Descritor Local consiste em (a) redução da dimensionalidade por meio do algoritmo Aprendizado de Dicionário aplicado ao vetor de características gerado pela camada de Detecção de Características Locais anterior, gerando um dicionário esparso  $D$  (conjunto de “átomos”), ou seja, gerando um vetor de características esparso a partir do descritor; (b) seguida por um processo de binarização, neste caso, a função Sinal, transformando a natureza do dicionário esparso  $D$ , em binário e esparso  $sgn(D)$ ; (c) seguida ainda, por uma transformação do dicionário esparso, que nada mais é do que um vetor de características em uma matriz quadrada. Finalmente, é definido para o centro de todas as matrizes quadradas um valor negativo (-1). Este processo é ilustrado no fluxograma da Figura 28.

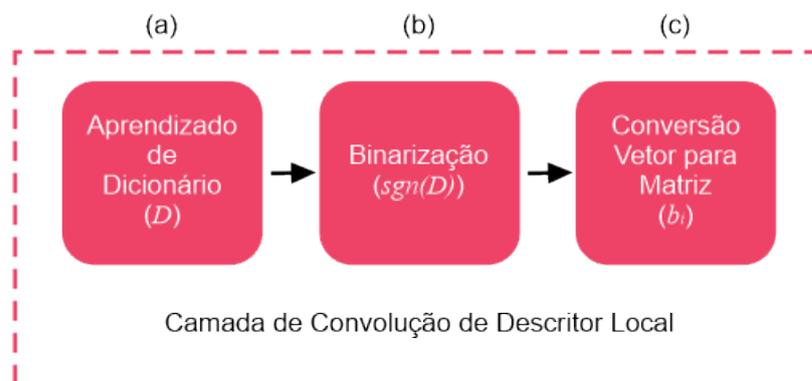


Figura 28 – Fluxograma da camada de Convolução de Descritor Local. Fonte: Elaborado pela autora.

Onde  $b_i$  representam os pesos de descritor, gerados a partir da matriz de transformação de  $sgn(D)$ .

Um *kernel* pode destacar uma determinada característica presente em uma imagem, como sombra, borda, entre outras, e o resultado da convolução é comumente conhecido como mapa de característica. Uma das etapas mais significativas dessa operação é a seleção dos filtros convolucionais a serem usados. Existem muitas maneiras de fazer isso, a mais comum, é determinada empiricamente. A capacidade de universalizar a seleção de filtros convolucionais é uma das motivações e inspirações por trás do *design* de reformulação proposto. A Figura 29 mostra o comportamento da convolução através dos 2 pesos de descritor (filtros convolucionais) de tamanho  $5 \times 5$ , gerados através da reformulação do descritor BRISK.

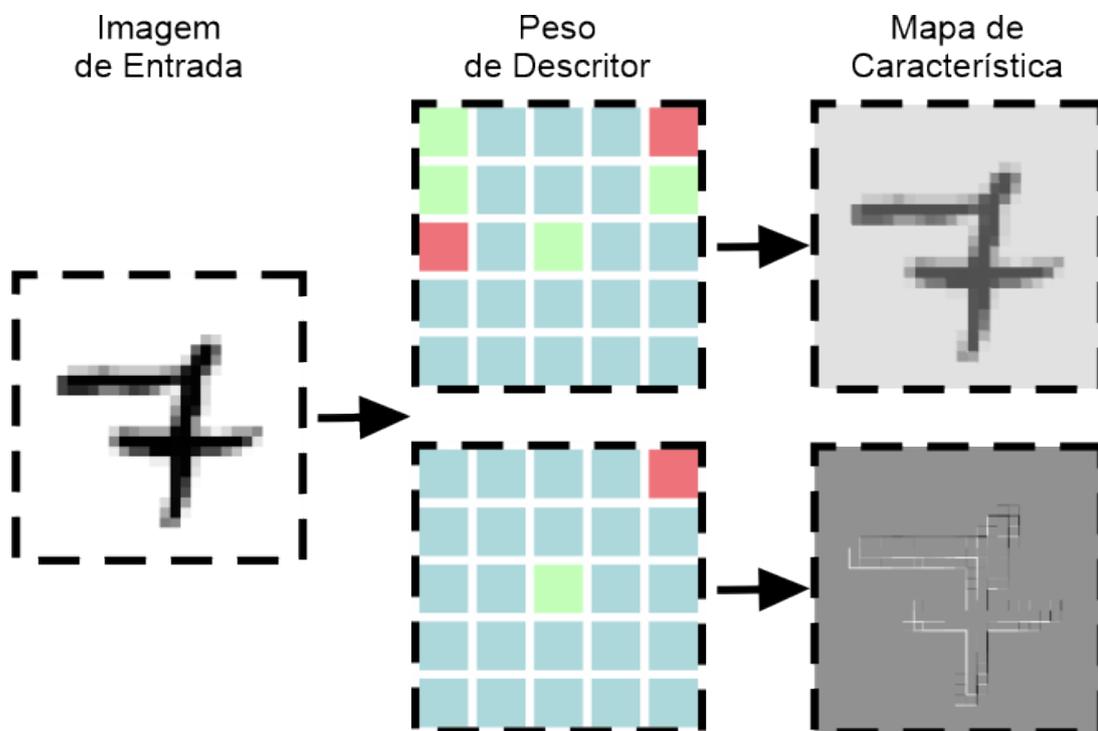


Figura 29 – Comportamento da convolução através dos 2 pesos de descritor, gerados através da reformulação do descritor BRISK.

Como visto na Figura 29, 2 pesos de descritor (filtros convolucionais)  $f$  de tamanho  $5 \times 5$  foram aplicados à imagem de entrada  $I$ , resultando em 2 mapas de características  $g$ . Os filtros vermelhos indicam que o filtro responderá positivamente aos pixels pretos nas imagens de entrada (1), enquanto os filtros verdes indicam que o filtro responderá negativamente aos pixels pretos nas imagens de entrada (-1). Os filtros azuis representam zero. Deste modo, a operação de convolução é a principal operação que ocorre nas imagens inseridas nas Redes Neurais Convolucionais, e seu principal objetivo está em extrair características de imagens de entrada. Portanto, uma imagem  $g$  é gerada pela convolução

de um filtro  $f$  com a imagem de entrada  $I$  definida pela Equação 4.1:

$$g(m, n) = \sum_{i=1}^q \sum_{j=1}^q f(i, j) I(m - i, n - j) \quad (4.1)$$

onde  $q$  é a dimensão da máscara de convolução. Deste modo, na Figura 30 é possível observar a arquitetura básica do modelo de Rede Neural Convolutacional de Descritores proposta nesta Dissertação.

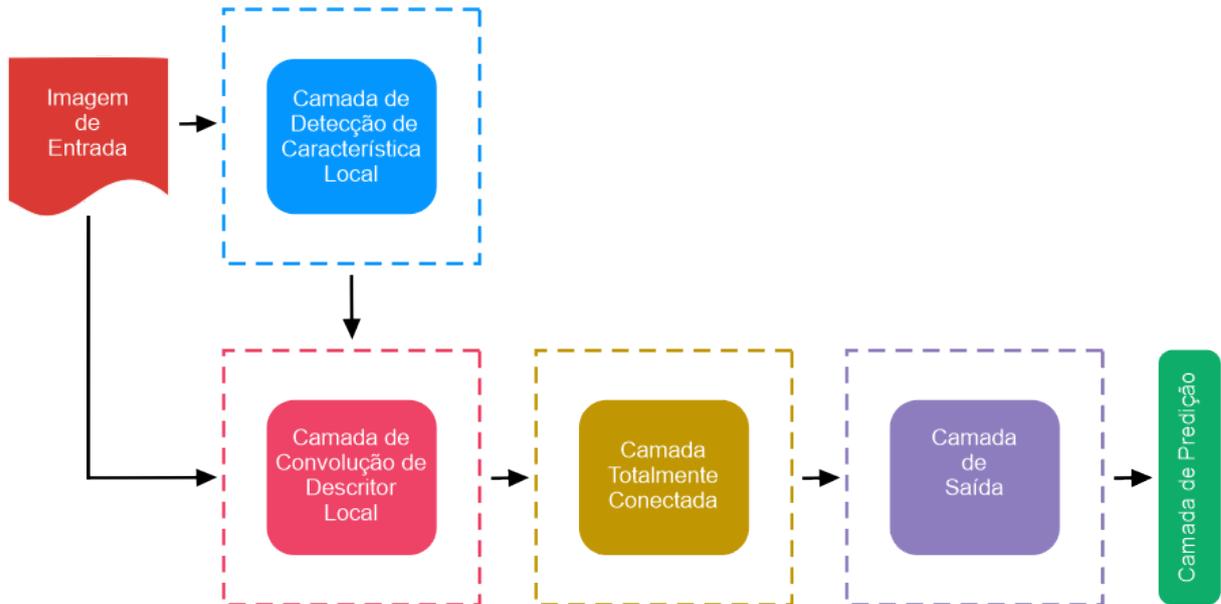


Figura 30 – Arquitetura básica do modelo de Rede Neural Convolutacional de Descritores. Fonte: Elaborado pela autora.

Por fim, neste Capítulo, foi apresentada a formulação do problema proposto, bem como, os resultados esperados. Foi apresentada ainda, a solução proposta desenvolvida nesta Dissertação. A seguir, serão apresentadas as metodologias adotadas para o desenvolvimento desta Dissertação.

## 5 MATERIAIS E MÉTODOS

Neste Capítulo, será apresentada uma breve descrição dos materiais e métodos empregados nos experimentos e simulações realizados no desenvolvimento desta Dissertação, a saber:

- **Kit de Desenvolvimento NVIDIA Jetson Nano:** É apresentado nesta Seção o microcomputador da família NVIDIA® Jetson™ — o Kit de Desenvolvimento NVIDIA Jetson Nano;
- **Configuração Experimental:** Nesta Seção é apresentada a configuração experimental adotada para a realização dos experimentos e simulados presentes nesta Dissertação;
- **Conjuntos de Dados Visuais:** É apresentado nesta Seção os conjuntos de dados visuais adotados para a realização dos experimentos e simulados presentes nesta Dissertação.

### 5.1 Kit de Desenvolvimento NVIDIA Jetson Nano

Utilizamos o microcomputador da família NVIDIA® Jetson™ — o Kit de Desenvolvimento NVIDIA Jetson Nano como hardware para os experimentos apresentados a Seção 6.1, que consistem na comparação da precisão, eficiência computacional e desempenho de cada um dos algoritmos dos Descritores de Características Locais e dos Descritores Binários Locais (apresentados na Seção 2.1) por meio da tarefa de reconhecimento e classificação pelo classificador *Perceptron* Multicamadas. Pretendemos, ainda, embarcar o microcomputador Jetson Nano a uma plataforma robótica móvel nas próximas etapas deste trabalho.

Optamos em utilizar o microcomputador Jetson Nano, por ser uma plataforma de baixo custo, fácil e intuitiva de se usar, e também, devido à eficiência e alto desempenho ao executar aplicações modernas de alta performance das áreas da Robótica, Inteligência Artificial, Aprendizado Profundo e Visão Computacional, como aplicações de detecção de objetos, segmentação, Veículos Terrestres e Aéreos não Tripulados (conforme observado no Capítulo 3), de forma rápida, eficiente, ao mesmo tempo, em que consome o mínimo de energia (apenas 5 a 10 watts) ao oferecer 472 GFLOPs, permitindo a execução de simultâneos processos de forma paralela. Na Figura 31 pode ser observado um modelo esquemático do microcomputador Jetson Nano com suas especificações técnicas.

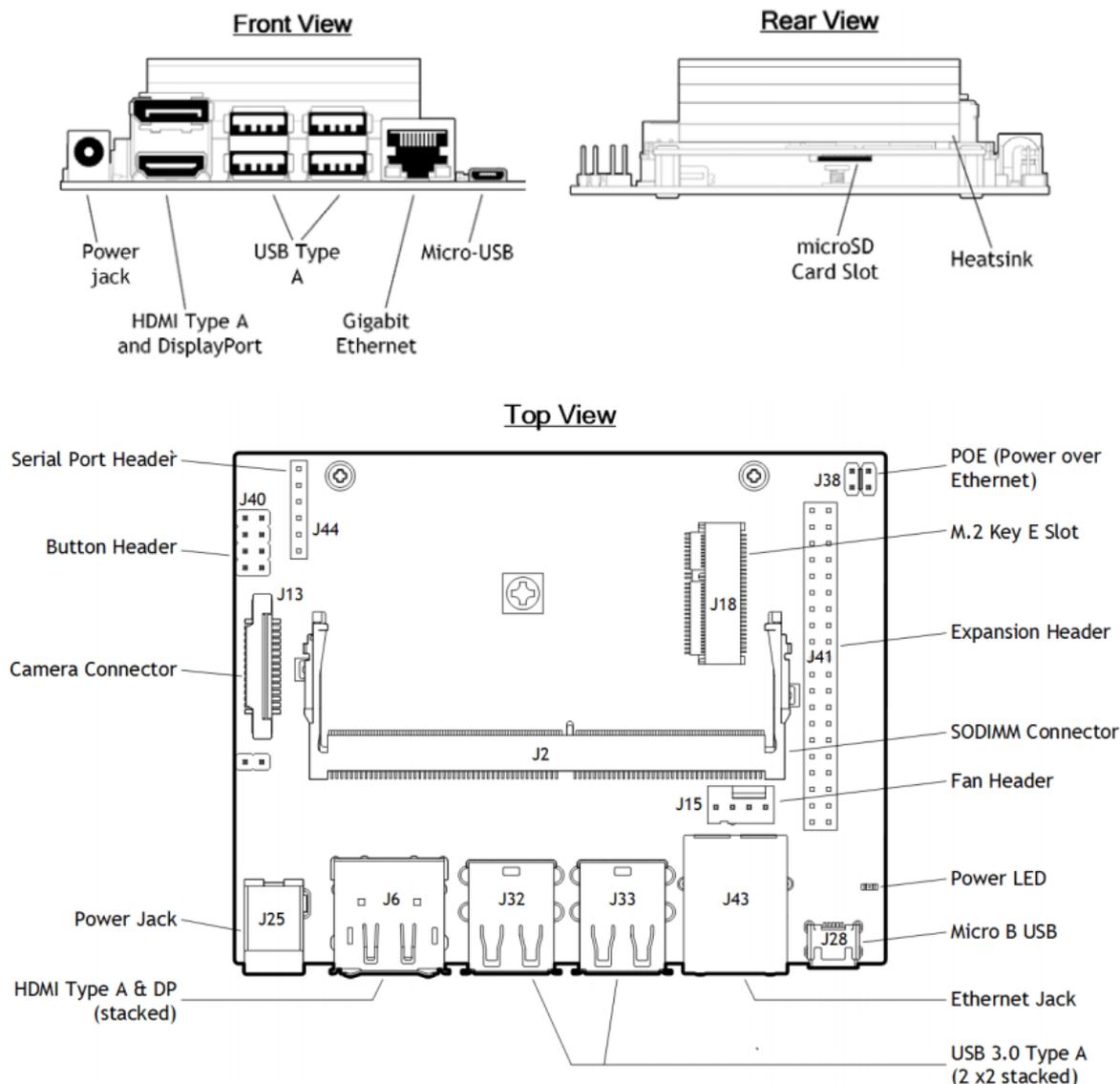


Figura 31 – Visão esquemática do microcomputador NVIDIA Jetson Nano. Fonte: (NVIDIA, 2019)

Outro ponto considerado, consiste em que o microcomputador Jetson Nano fornece um ambiente Linux baseado no Ubuntu 18.04 LTS, além da integração com bibliotecas para Aprendizado de Máquina baseada em Aprendizado Profundo como o TensorFlow e Keras, Visão Computacional como o OpenCV — bibliotecas estas, que manifestamos interesse em utiliza-las no desenvolvimento desta Dissertação. Deste modo, para a utilização do microcomputador Jetson Nano é necessário realizar algumas etapas de instalação e posteriormente, configuração para utilização de recursos como Python, OpenCV, TensorFlow, Keras, entre outros no microcomputador Jetson Nano. Neste contexto, foi descrito uma documentação apresentando as etapas de instalação e configuração para este dispositivo, disponível no Apêndice A e B.

## 5.2 Configuração Experimental

Os parâmetros de configuração de detectores e descritores apresentados na Seção 2.1 são descritos abaixo:

```
SIFT = cv.xfeatures2d.SIFT_create(  
    nfeatures = 0,  
    nOctaveLayers = 3,  
    contrastThreshold = 0.04,  
    edgeThreshold = 10,  
    sigma = 1.6)
```

```
SURF = cv.xfeatures2d.SURF_create(  
    hessianThreshold = 500,  
    nOctaves = 4,  
    nOctaveLayers = 3,  
    extended = True,  
    upright = False)
```

```
KAZE = cv.KAZE_create(  
    extended = False,  
    upright = False,  
    threshold = 0.001,  
    nOctaves = 4,  
    nOctaveLayers = 4,  
    diffusivity = cv.KAZE_DIFF_PM_G2)
```

```
BRIEF = cv.xfeatures2d.BriefdescriptorExtractor_create(  
    bytes = 16,  
    use_orientation = True)
```

```
ORB = cv.ORB_create(  
    nfeatures = 300,  
    scaleFactor = 1.2,  
    nlevels = 2,  
    edgeThreshold = 2,  
    firstLevel = 0,  
    WTA_K = 2,  
    scoreType = 0,  
    patchSize = 2,  
    fastThreshold = 20)
```

```

BRISK = cv.BRISK_create(
    thresh = 30,
    octaves = 3,
    patternScale = 1.0)

AKAZE = cv.AKAZE_create(
    descriptor_type = cv.AKAZE_descriptor_MLDB,
    descriptor_size = 0,
    descriptor_channels = 3,
    threshold = 0.001,
    nOctaves = 4,
    nOctaveLayers = 4,
    diffusivity = cv.KAZE_DIFF_PM_G2)

FREAK = cv.xfeatures2d.FREAK_create(
    orientationNormalized = True,
    scaleNormalized = True,
    patternScale = 22.0,
    nOctaves = 4)

```

Os descritores BRIEF e FREAK são apenas descritores. Para esses dois cenários, foram utilizados os detectores ORB e SURF, respectivamente, enquanto para os demais algoritmos, foram utilizados os mesmos descritores dos detectores, como pode ser visto na Tabela 4.

Tabela 4 – Pares de Descritores e Detectores. Fonte: Elaborado pela autora.

Detector	Descritor
SIFT	SIFT
SURF	SURF
KAZE	KAZE
ORB	BRIEF
ORB	ORB
BRISK	BRISK
AKAZE	AKAZE
SURF	FREAK

Em relação ao classificador, avaliamos a eficiência deste experimento com seis classificadores *Perceptron* Multicamadas. Entretanto, abaixo, pode ser observado a configuração do melhor desempenho dos classificadores *Perceptron* Multicamadas (os resultados serão detalhados mais a frente na Seção 6.1.2):

- **MLP6:** Stochastic Gradient Descent Solver, 0.9 Momentum, 0.0001 Alpha, 0.01 Adaptive Learning Rate, 2019 Random State, and 5.000 maximum iterations.

A função de ativação usada neste modelo é a função ReLU:

$$f(s) = \max(0, s) \quad (5.1)$$

portanto, se o potencial de ativação  $s$  for maior que o limite de ativação, o modelo *Perceptron* Multicamadas será considerado ativado (a saída  $y$  será igual a 1), e será considerado que o modelo *Perceptron* Multicamadas estará desativado, caso contrário. Na Figura 32, é possível ver a estrutura básica das camadas escondidas do modelo *Perceptron* Multicamadas usado para este experimento e simulação preliminar.

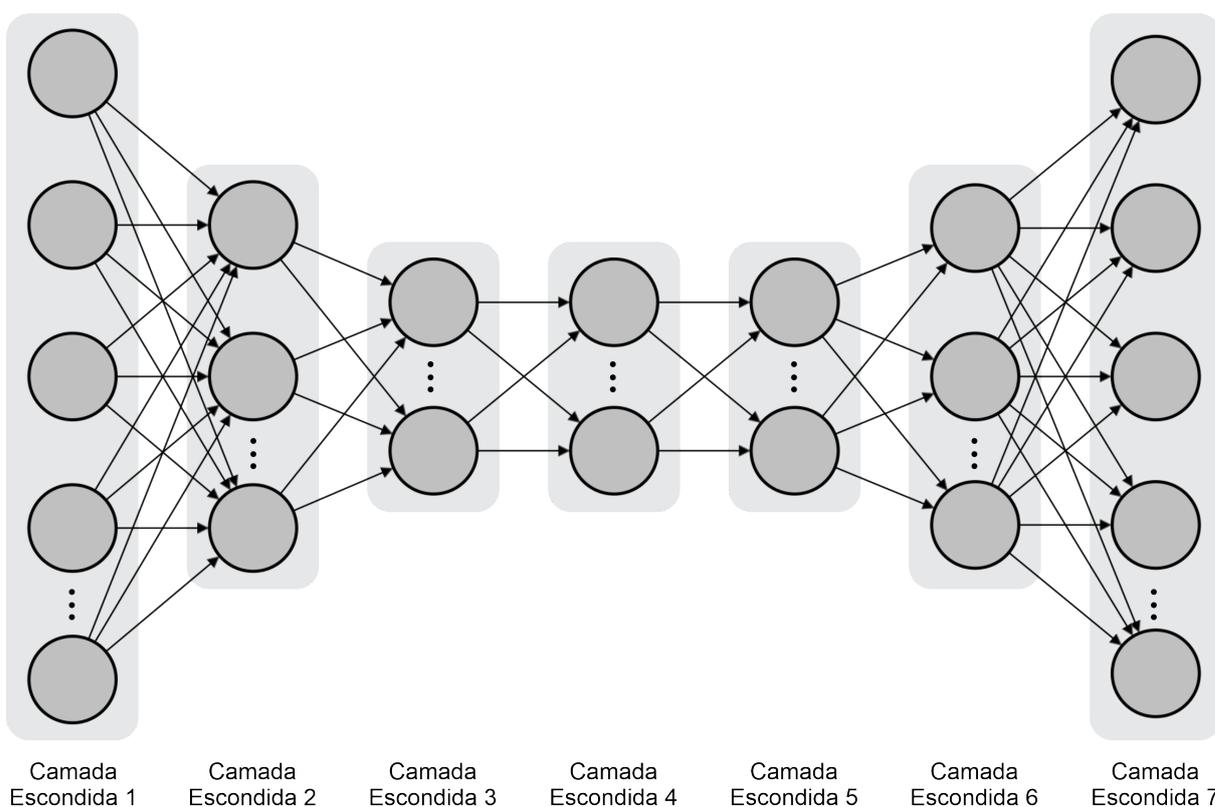


Figura 32 – Estrutura das camadas escondidas do classificador *Perceptron* Multicamadas. Fonte: Elaborado pela autora.

Onde, as camadas escondidas de índice 1 e 7 possuem 50 neurônios, as camadas escondidas de índice 2 e 6 possuem 30 neurônios, e por fim, as camadas escondidas de índice 3, 4 e 5 possuem 20 neurônios.

### 5.3 Conjuntos de Dados Visuais

Para avaliar as técnicas apresentadas no Capítulo 6 e validá-las com tarefas de reconhecimento e classificação, foram utilizados seis conjuntos de dados visuais de imagens públicas:

- **MNIST:** *Modified National Institute of Standards and Technology database* (MNIST) (LECUN et al., 1998), consiste em 70.000 imagens de dígitos manuscritas. Onde 60.000 imagens fazem parte do conjunto de treinamento e 10.000 imagens no conjunto de teste, o tamanho das imagens neste conjunto de dados visuais é de  $28 \times 28$  pixels. Na Figura 33 é possível observar algumas amostras no conjunto de dados visuais MNIST. Ainda, a distribuição de amostras do conjunto de dados visuais MNIST é apresentada na Figura 34.

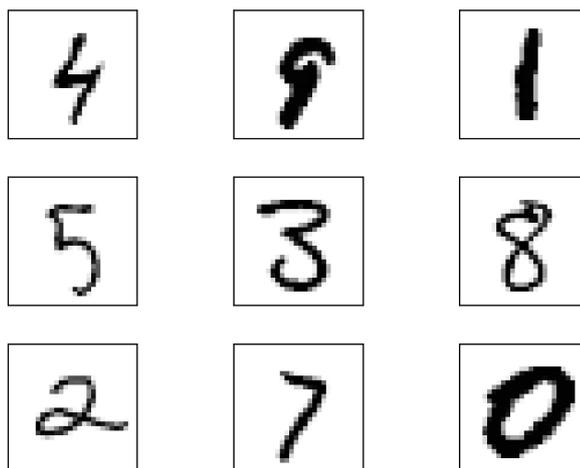


Figura 33 – Amostras presente no conjunto de dados visuais MNIST. Fonte: Adaptado de (LECUN et al., 1998)

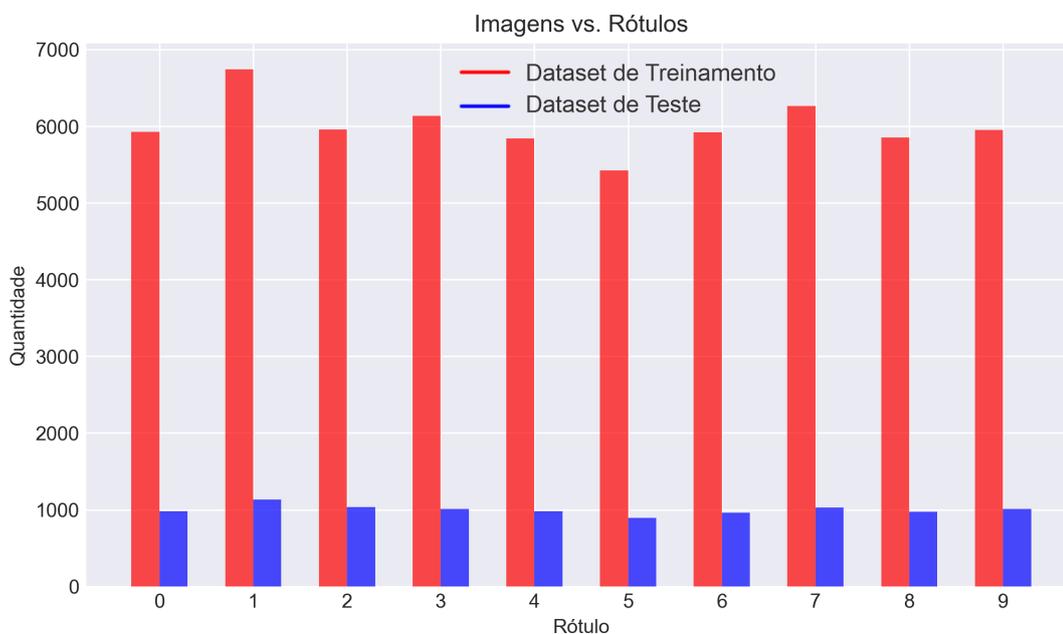


Figura 34 – Distribuição de amostras do conjunto de dados visuais MNIST. Fonte: Elaborado pela autora.

- **JAFFE:** *Japanese Female Facial Expression* (JAFFE) (LYONS et al., 1998), consiste em 213 imagens de 10 modelos japonesas. Selecionamos 150 imagens para o conjunto de treinamento, enquanto para o conjunto de teste, selecionamos 63 imagens. Para este experimento, definimos o tamanho das imagens neste conjunto de dados visuais como  $48 \times 48$  pixels. Na Figura 35 é possível observar algumas amostras no conjunto de dados visuais JAFFE. Ainda, a distribuição de amostras do conjunto de dados visuais JAFFE é apresentada na Figura 36.

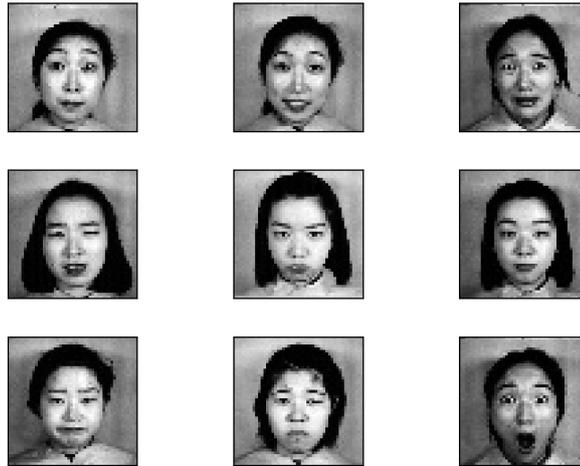


Figura 35 – Amostras presente no conjunto de dados visuais JAFFE. Fonte: Adaptado de (LYONS et al., 1998)

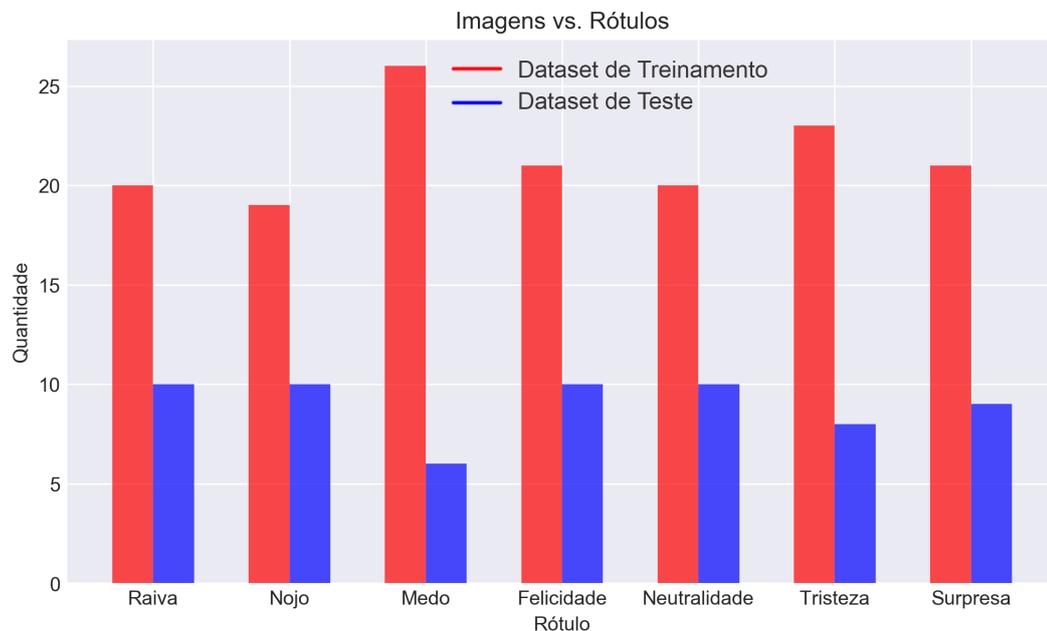


Figura 36 – Distribuição de amostras do conjunto de dados visuais JAFFE. Fonte: Elaborado pela autora.

- **Extended CK+**: *Extended Cohn-Kanade* (CK+) (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010), consiste em 593 seqüências de vídeo gravadas de 123 indivíduos. Seleccionamos 10.263 imagens para o conjunto de treinamento, enquanto para o conjunto de teste, seleccionamos 565 imagens. Para este experimento, definimos o tamanho das imagens neste conjunto de dados visuais como  $48 \times 48$  pixels. Na Figura 37 é possível observar algumas amostras no conjunto de dados visuais Extended CK+. Ainda, a distribuição de amostras do conjunto de dados visuais Extended CK+ é apresentada na Figura 38.

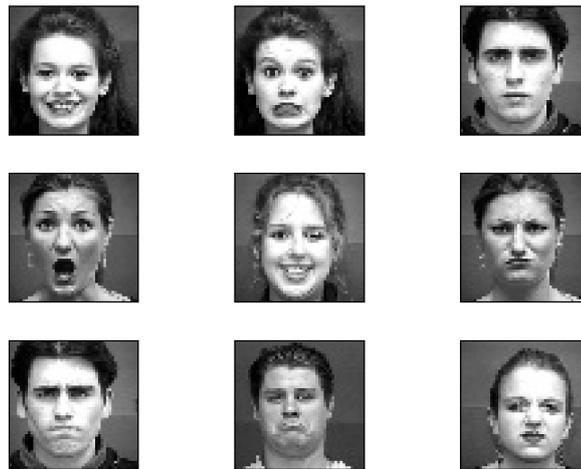


Figura 37 – Amostras presente no conjunto de dados visuais Extended CK+. Fonte: Adaptado de (KANADE; COHN; TIAN, 2000; LUCEY et al., 2010)

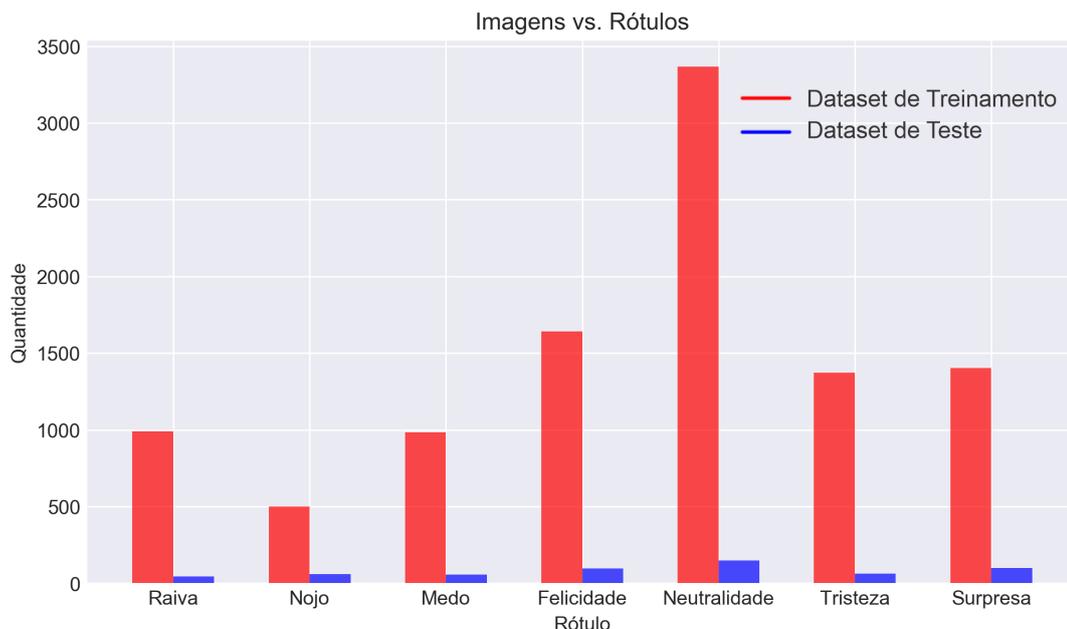


Figura 38 – Distribuição de amostras do conjunto de dados visuais Extended CK+. Fonte: Elaborado pela autora.

- **FEI:** *FEI Face Database* (FEI) (THOMAZ; GIRALDI, 2010), é um conjunto de dados visuais brasileiros e consiste em 2.800 imagens faciais separadas em 14 imagens por 200 indivíduos, sendo 100 imagens masculinas e 100 imagens femininas. Seleccionamos um subconjunto, que consiste em 400 imagens de faces frontais separadas em 2 classes distintas (feliz e neutro), ambas classes contendo 100 imagens. Separamos 280 e 120 imagens respectivamente, para o conjunto de treinamento e teste. O tamanho das imagens neste conjunto de dados visuais é  $120 \times 120$  pixels. Na Figura 39 é possível observar algumas amostras no conjunto de dados visuais FEI. Ainda, a distribuição de amostras do conjunto de dados visuais FEI é apresentada na Figura 40.



Figura 39 – Amostras presente no conjunto de dados visuais FEI. Fonte: Adaptado de (THOMAZ; GIRALDI, 2010)

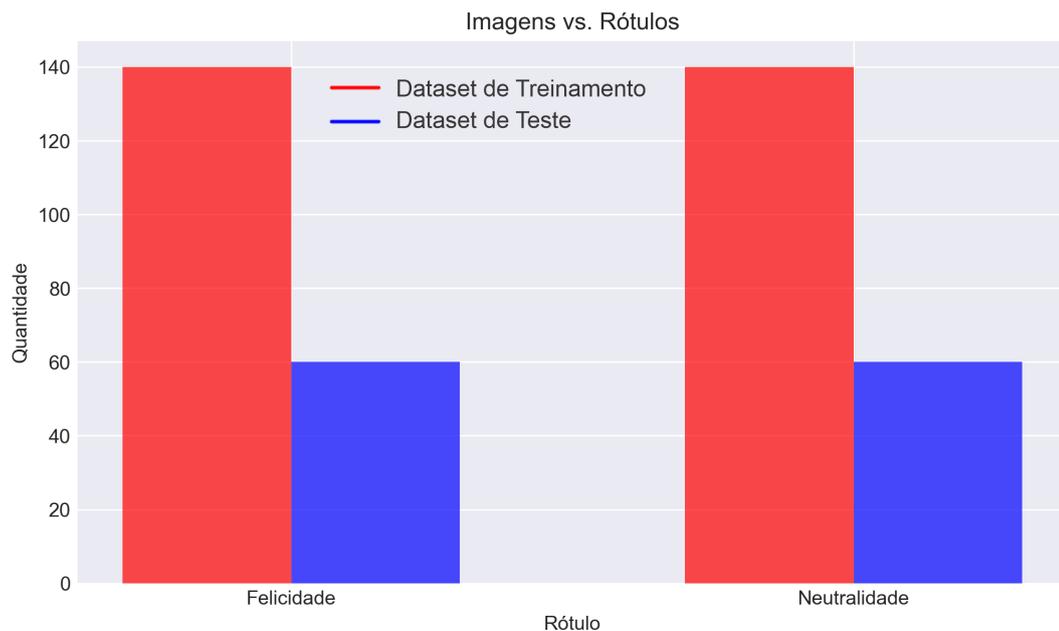


Figura 40 – Distribuição de amostras do conjunto de dados visuais FEI. Fonte: Elaborado pela autora.

- CIFAR-10:** *Canadian Institute For Advanced Research* (CIFAR-10) (KRIZHEVSKY; HINTON et al., 2009) consiste em 60.000 imagens separadas em 10 classes diferentes (aviões, carros, pássaros, gatos, veados, cães, sapos, cavalos, navios e caminhões), cada uma dessas classes contém 6.000 imagens, onde 50.000 imagens fazem parte do conjunto de treinamento e 10.000 imagens do conjunto de teste. O tamanho das imagens neste conjunto de dados visuais é de  $32 \times 32$  pixels. Na Figura 41 é possível observar algumas amostras no conjunto de dados visuais CIFAR-10. Ainda, a distribuição de amostras do conjunto de dados visuais CIFAR-10 é apresentada na Figura 42.

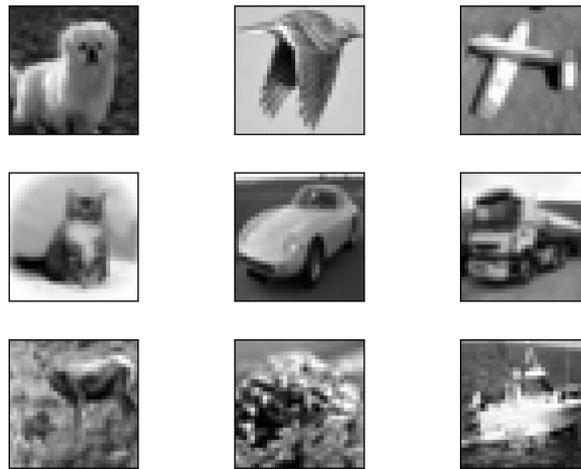


Figura 41 – Amostras presente no conjunto de dados visuais CIFAR-10. Fonte: Adaptado de (KRIZHEVSKY; HINTON et al., 2009)

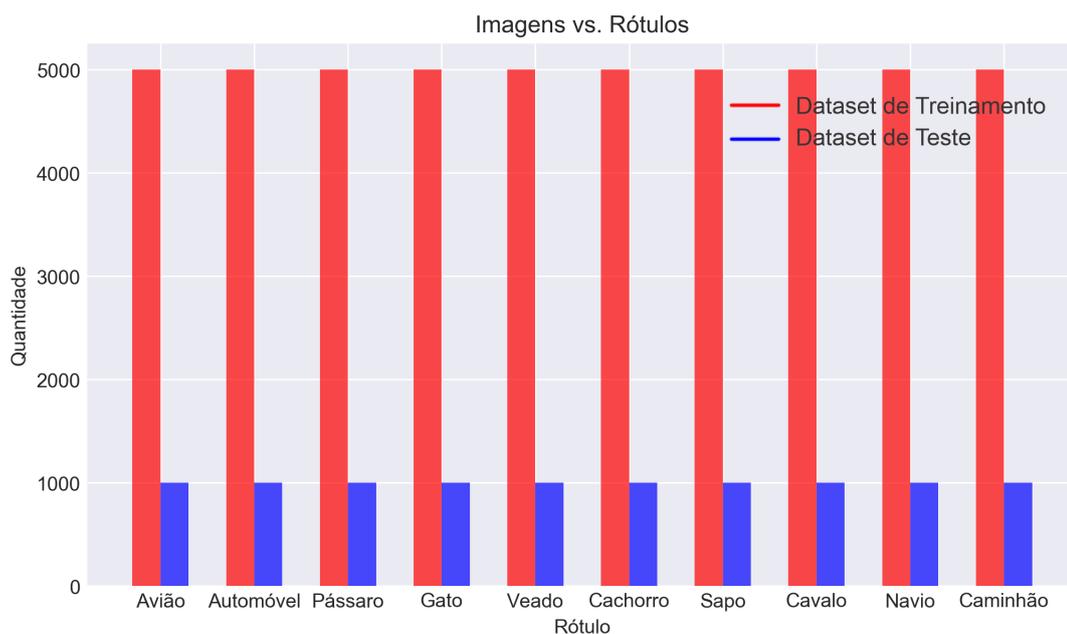


Figura 42 – Distribuição de amostras do conjunto de dados visuais CIFAR-10. Fonte: Elaborado pela autora.

- **FER-2013:** *Facial Expression Recognition 2013* (FER-2013) foi apresentado no desafio *2013 Challenges in Representation Learning* (ICML 2013) (GOODFELLOW et al., 2013). É composto por 28.709 imagens no conjunto de treinamento e 3.589 imagens no conjunto de testes criadas com a API do Google — *Google Images*, suas imagens possuem diversos tipos de variações, tornando a tarefa de classificação neste conjunto de dados visuais um desafio. O tamanho das imagens neste conjunto de dados visuais é  $48 \times 48$  pixels. Na Figura 43 é possível observar algumas amostras no conjunto de dados visuais FER-2013. Ainda, a distribuição de amostras do conjunto de dados visuais FER-2013 é apresentada na Figura 44.

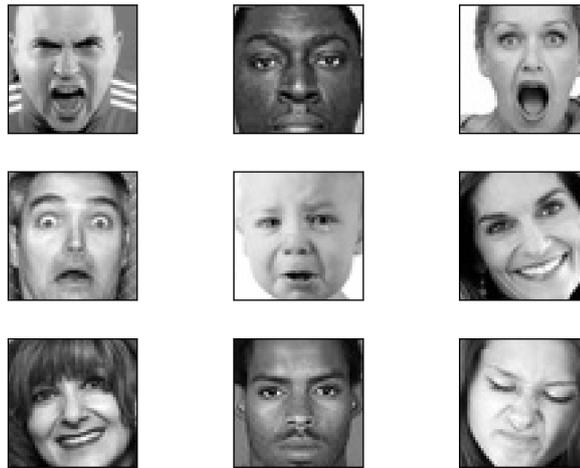


Figura 43 – Amostras presente no conjunto de dados visuais FER-2013. Fonte: Adaptado de (GOODFELLOW et al., 2013)

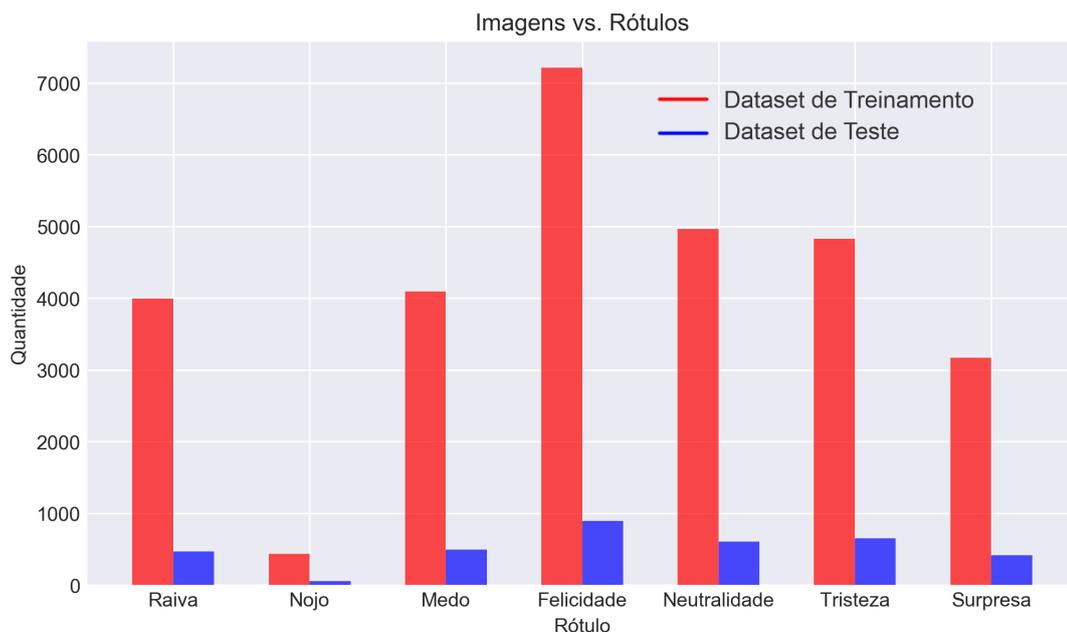


Figura 44 – Distribuição de amostras do conjunto de dados visuais FER-2013. Fonte: Elaborado pela autora.

Por fim, neste Capítulo, foi apresentada uma breve descrição das metodologias adotadas nos experimentos e simulações realizados no desenvolvimento desta Dissertação. A seguir, serão apresentados os resultados dos experimentos e simulações realizados no decorrer do desenvolvimento desta Dissertação.

## 6 EXPERIMENTOS E RESULTADOS

Neste Capítulo, apresentamos os experimentos e simulações realizados no decorrer do desenvolvimento desta Dissertação, onde cada etapa foi importante para alcançar o objetivo final. Portanto, passaremos por alguns tópicos, a saber:

- **Avaliação comparativa entre Descritores de Características por meio da abordagem de Saco de Características Visuais com *Perceptron* Multicamadas:** Experimentos e avaliação da abordagem de Saco de Características Visuais com *Perceptron* Multicamadas;
- **Rede Neural Convolutacional de Descritores (DescNet):** Experimentos e avaliação da abordagem proposta nesta Dissertação de mestrado.

### 6.1 Avaliação comparativa entre Descritores de Características por meio da abordagem de Saco de Características Visuais com *Perceptron* Multicamadas

A linguagem Python 3.6, bem como os pacotes OpenCV 4.1 e Scikit-Learn 0.19, são usados para realizar os experimentos apresentados nesta Seção, junto ao microcomputador Jetson Nano (conforme detalhado na Seção 5.1). Implementação e futuras atualizações estarão disponíveis no repositório do GitHub que pode ser consultado no Apêndice C.

#### 6.1.1 Abordagem Proposta

Os experimentos apresentados nesta Seção consistem na comparação da precisão, eficiência computacional e desempenho de cada um dos algoritmos dos Descritores de Características Locais e dos Descritores Binários Locais (apresentados na Seção 2.1) por meio da tarefa de reconhecimento e classificação pelo classificador *Perceptron* Multicamadas. É realizada também a comparação entre o tempo de processamento na geração do índice de Saco de Características Visuais nas etapas de treinamento e teste. Ao analisar os resultados de cada algoritmo, esperamos poder selecionar um Descritor Binário Local eficiente que será abordado nas próximas etapas deste trabalho.

#### 6.1.2 Resultados

Conforme especificado anteriormente, o tamanho das imagens do MNIST são  $28 \times 28$  pixels, as imagens do CIFAR-10 são  $32 \times 32$  pixels e as imagens do JAFFE, Extended

CK+ e FER-2013 são 48x48 píxeis, enquanto o tamanho das imagens do FEI são  $120 \times 120$  píxeis. Alguns descritores como SIFT e SURF podem obter resultados excelentes em *patches* menores, mas a maioria dos descritores obtém resultados excelentes em um *patch* de  $32 \times 32$  para encontrar pontos-chave e calcular descritores, bem como ORB. Como MNIST, JAFFE, Extended CK+, CIFAR-10 e FER-2013 têm suas imagens menores, não foi possível obter resultados nesses conjuntos de dados visuais usando os descritores BRIEF, AKAZE e FREAK. Isso porque esses descritores não se ajustam adequadamente nas imagens presentes nesses conjuntos de dados visuais, ou seja, para evitar o retorno de pontos-chave sem descritores, o BRIEF, AKAZE e FREAK removem os pontos-chave. Porém, com os descritores BRIEF, AKAZE e FREAK, foi possível obter resultados no conjunto de dados visuais FEI, pois o tamanho das imagens do FEI são  $120 \times 120$  pixels, ou seja, o tamanho das imagens são mais significativas para estes descritores. Quando os descritores tentam extrair características dos conjuntos de dados visuais JAFFE e FEI, o resultado é obtido quase que instantaneamente (há uma pequena variação entre zero a sete segundos). Este caso ocorre para ambos os conjuntos de dados visuais porque eles possuem muito poucas amostras, no caso 150 e 280 imagens para o conjunto de treinamento, respectivamente, e 63 e 120 imagens para o conjunto de teste. Na Figura 45 pode ser observado os conjuntos de dados visuais utilizados neste trabalho reorganizarmos por tamanho (quantidade de amostras  $\times$  resolução), neste sentido, podemos classificar o conjunto de dados visuais JAFFE como o conjunto de dados mais leve e o conjunto de dados visuais FER-2013 como o conjunto de dados mais pesado.

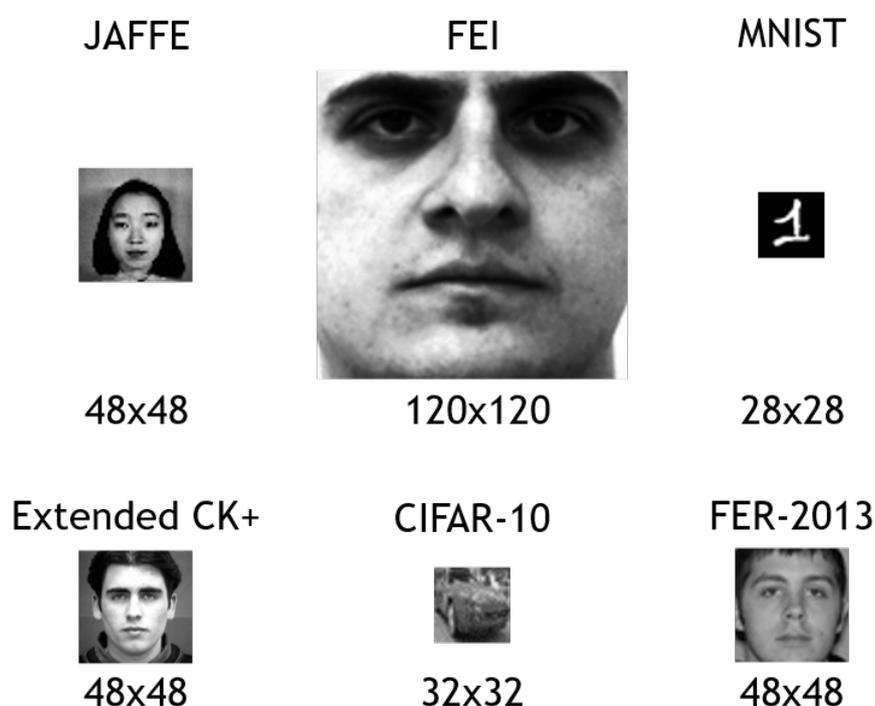


Figura 45 – Conjuntos de dados visuais reorganizarmos por tamanho (quantidade de amostras  $\times$  resolução). Fonte: Elaborado pela autora.

Não obstante da abordagem de Saco de Características Visuais, quando os descritores operam em conjuntos de dados visuais JAFFE e FEI, os resultados nas etapas (c) Representação de Característica, (d) Geração de Vocabulário Visual e (e) Representação de Imagem, em relação à etapa de treinamento, mostrados na Figura 16, e os resultados nas etapas (c) Representação de Característica e (d) Representação de Imagem, em relação à etapa de teste, mostrados na Figura 17, são alcançados quase que instantaneamente.

Em relação à abordagem de Saco de Características Visuais na etapa de treinamento mostrada na Figura 16, na Tabela 5, é possível observar o tempo de processamento na etapa (c) Representação de Característica para os descritores SIFT, SURF, KAZE, ORB e BRISK, trabalhando com os conjuntos de dados visuais MNIST, Extended CK+, CIFAR-10 e FER-2013 apresentados neste trabalho.

Tabela 5 – Tempo de processamento (min.) na etapa de Representação de Características da abordagem Saco de Características Visuais da etapa de treinamento. Fonte: Elaborado pela autora.

Algoritmos	Dataset	Conjunto de dados visuais			
		MNIST	Extended CK+	CIFAR-10	FER-2013
SIFT	Treinamento	02:37	01:01	02:59	02:51
SURF		00:49	<b>00:16</b>	00:54	<b>00:25</b>
KAZE		02:59	01:02	02:59	02:52
ORB		<b>00:32</b>	00:17	<b>00:46</b>	00:39
BRISK		-	00:29	00:57	01:00

É possível observar na Tabela 5, um alto desempenho do microcomputador Jetson Nano neste experimento, pois o tempo máximo para a etapa de Extração de Características foi de apenas 02:59 minutos. Entre os Descritores de Características Locais, SURF leva menos tempo para extrair características do que outros Descritores de Características Locais, enquanto o Descritor Binário Local ORB excede outros descritores nesta etapa para cada conjunto de dados visuais. Como MNIST possui as menores imagens, o mesmo caso que ocorre com os descritores BRIEF, AKAZE e FREAK acontece com BRISK, e, neste caso, não foi possível obter resultados.

Ainda com relação à abordagem de Saco de Características Visuais, na etapa de treinamento mostrada na Figura 16, a próxima etapa (d) Geração de Vocabulário Visual, consiste em usar o algoritmo de agrupamento K-Means para gerar o possível dicionário de vocabulário visual sobre a coleção de vetores de características extraídos na etapa anterior. Na Tabela 6, é possível observar o tempo de processamento na etapa (d) Geração de Vocabulário Visual para os descritores SIFT, SURF, KAZE, ORB e BRISK nos conjuntos de dados visuais MNIST, Extended CK+, CIFAR-10, e FER-2013 apresentados neste trabalho.

Tabela 6 – Tempo de processamento (min.) na etapa de Geração de Vocabulário Visual da abordagem Saco de Características Visuais da etapa de treinamento. Fonte: Elaborado pela autora.

Algoritmos	Dataset	Conjunto de dados visuais			
		MNIST	Extended CK+	CIFAR-10	FER-2013
<b>SIFT</b>	<b>Treinamento</b>	02:30	00:48	05:14	02:50
<b>SURF</b>		<b>00:01</b>	<b>00:05</b>	00:03	00:20
<b>KAZE</b>		01:45	00:39	05:09	04:13
<b>ORB</b>		02:39	01:46	03:22	02:43
<b>BRISK</b>		-	00:06	<b>00:00</b>	<b>00:06</b>

Entre os Descritores de Características Locais, SURF levou menos tempo para gerar o dicionário de vocabulário visual. Em contraste, em relação aos Descritores Binários Locais, BRISK foi o mais rápido a fazer isso. Como já comentado, como o MNIST possui as menores imagens, o mesmo caso que ocorre com os descritores BRIEF, AKAZE e FREAK também ocorre com BRISK, e, neste caso, não foi possível obter nenhum resultado.

Ainda em relação à abordagem de Saco de Características Visuais da etapa de treinamento mostrada na Figura 16, a próxima etapa (e) Representação de Imagem, consiste em calcular os retornos de frequências de cada *cluster* na etapa anterior, resultando em um Histograma de Características Visuais. O Histograma de Características Visuais quando realizado sobre conjuntos de dados visuais JAFFE e FEI, foi obtido instantaneamente, conforme comentado anteriormente. O tempo de processamento para obter o histograma de características visuais para os descritores SIFT, SURF, KAZE, ORB e BRISK nos conjuntos de dados visuais MNIST, Extended CK+, CIFAR-10 e FER-2013, respectivamente, pode ser observado na Tabela 7.

Tabela 7 – Tempo de processamento (min.) na etapa de Representação de Imagem da abordagem Saco de Características Visuais da etapa de treinamento. Fonte: Elaborado pela autora.

Algoritmos	Dataset	Conjunto de dados visuais			
		MNIST	Extended CK+	CIFAR-10	FER-2013
<b>SIFT</b>	<b>Treinamento</b>	01:04	00:11	00:56	00:33
<b>SURF</b>		<b>00:13</b>	<b>00:10</b>	00:20	00:29
<b>KAZE</b>		01:04	00:11	00:56	00:33
<b>ORB</b>		01:02	00:12	00:55	00:33
<b>BRISK</b>		-	<b>00:10</b>	<b>00:01</b>	<b>00:19</b>

SURF leva menos tempo para obter o histograma de características visuais do que os outros Descritores de Características Locais, enquanto o BRISK obteve os melhores resultados em comparação com a maioria dos Descritores Binários Locais. Após gerar um histograma de características visuais, o mesmo é passado para a etapa (f) Treinamento, para treinar o modelo *Perceptron* Multicamadas.

Em relação à abordagem de Saco de Características Visuais da etapa de teste mostrada na Figura 17, podemos observar na Tabela 8 o tempo de processamento na etapa (c) Representação de Característica e (d) Representação de Imagem para os descritores SIFT, SURF, KAZE, ORB e BRISK nos conjuntos de dados visuais MNIST, Extended CK+, CIFAR-10 e FER-2013 apresentados neste trabalho.

Tabela 8 – Tempo de processamento (min.) nas etapas de Representação de Características e Representação de Imagem da abordagem Saco de Características Visuais da etapa de teste. Fonte: Elaborado pela autora.

Algoritmos	Dataset	Conjunto de dados visuais			
		MNIST	Extended CK+	CIFAR-10	FER-2013
<b>SIFT</b>	Teste	00:35	00:04	00:46	00:25
<b>SURF</b>		<b>00:09</b>	<b>00:01</b>	<b>00:08</b>	<b>00:06</b>
<b>KAZE</b>		00:39	00:04	00:46	00:25
<b>ORB</b>		00:16	<b>00:01</b>	00:20	00:09
<b>BRISK</b>		-	00:02	00:11	00:09

SURF leva menos tempo para extrair características e obter o histograma de características visuais do que outros Descritores de Características Locais e Descritores Binários Locais. Após gerar um Histograma de Características Visuais, é realizada a inferência, na etapa (e) Classificação.

Na Tabela 9 é possível observar os resultados alcançados por meio o classificador *Perceptron* Multicamadas, onde o desempenho e a eficiência de seis conjuntos diferentes de parâmetros foram avaliados junto aos descritores BRIEF, AKAZE e FREAK no conjunto de dados visuais da FEI.

Tabela 9 – Taxa de acurácia (%) nas etapas de teste do classificador *Perceptron* Multicamadas no conjunto de dados visuais FEI. Fonte: Elaborado pela autora.

Algoritmos	Dataset	Modelo Perceptron Multicamadas					
		MLP1	MLP2	MLP3	MLP4	MLP5	MLP6
<b>BRIEF</b>	FEI	0.78	0.74	0.77	0.76	0.82	0.85
<b>AKAZE</b>		<b>0.85</b>	<b>0.87</b>	<b>0.83</b>	<b>0.84</b>	<b>0.83</b>	<b>0.86</b>
<b>FREAK</b>		0.47	0.47	0.47	0.51	0.51	0.54

Deste modo, na Tabela 10, é possível observar os resultados alcançados por meio o classificador *Perceptron* Multicamadas, onde o desempenho e a eficiência de seis conjuntos diferentes de parâmetros foram avaliados junto aos descritores SIFT, SURF, KAZE, ORB e BRISK em cada conjunto de dados visuais.

Tabela 10 – Taxa de acurácia (%) na etapa do teste classificador *Perceptron* Multicamadas em cada conjunto de dados visuais. Fonte: Elaborado pela autora.

Algoritmos	Conjunto de Dados Visuais	Modelo Perceptron Multicamadas					
		MLP1	MLP2	MLP3	MLP4	MLP5	MLP6
SIFT	MNIST	0.67	0.67	0.67	0.68	0.67	0.68
	JAFFE	0.10	0.10	0.10	0.21	0.21	0.25
	Extended CK+	0.38	0.38	0.40	0.41	0.44	0.39
	FEI	0.69	0.67	0.63	0.75	0.76	0.72
	CIFAR-10	0.25	0.25	0.25	0.25	0.25	0.24
	FER-2013	0.26	0.27	0.27	0.26	0.26	0.26
SURF	MNIST	0.68	0.68	0.67	0.69	0.68	0.68
	JAFFE	0.10	0.10	0.10	0.21	0.22	0.21
	Extended CK+	0.40	0.37	0.40	0.43	0.49	0.44
	FEI	0.72	0.74	0.61	0.74	0.72	0.74
	CIFAR-10	0.11	0.11	0.11	0.18	0.19	0.18
	FER-2013	0.26	0.26	0.26	0.25	0.24	0.24
KAZE	MNIST	0.65	0.65	0.66	0.67	0.68	0.66
	JAFFE	0.08	0.19	0.24	0.22	0.21	0.22
	Extended CK+	0.36	0.36	0.41	0.44	0.42	0.43
	FEI	0.62	0.62	0.63	0.58	0.67	0.68
	CIFAR-10	0.25	0.25	0.25	0.25	0.24	0.24
	FER-2013	0.26	0.25	0.26	0.26	0.25	0.25
ORB	MNIST	0.53	0.52	0.53	0.52	0.53	0.52
	JAFFE	0.14	0.13	0.10	0.11	0.19	0.14
	Extended CK+	0.31	0.31	0.30	0.30	0.27	0.29
	FEI	0.50	0.57	0.58	0.65	0.64	0.70
	CIFAR-10	0.23	0.23	0.23	0.22	0.22	0.21
	FER-2013	0.26	0.25	0.25	0.26	0.25	0.26
BRISK	MNIST	-	-	-	-	-	-
	JAFFE	0.11	0.11	0.11	0.19	0.19	0.21
	Extended CK+	0.26	0.29	0.31	0.27	0.30	0.34
	FEI	0.82	0.82	0.82	0.85	0.84	0.82
	CIFAR-10	0.13	0.13	0.13	0.13	0.13	0.20
	FER-2013	0.25	0.25	0.25	0.24	0.24	0.25

Verificou-se que muitos algoritmos tiveram resultados abaixo do esperado. Acreditamos que isso ocorreu porque a maioria dos conjuntos de dados visuais usados, possuem amostras de imagens muito pequenas. Porém, conforme descrito em (CSURKA et al., 2004), para a abordagem de Saco de Características Visuais distinguir mudanças relevantes em partes da imagem, o tamanho do dicionário deve ser grande o suficiente; no entanto, não é tão grande a ponto de distinguir variações irrelevantes, como ruído. Com o método de agrupamento escolhido (algoritmo de agrupamento K-Means), é comum definir empiricamente o valor para  $k$ , ao modificar este parâmetro, acreditamos que os resultados alcançados podem ser aprimorados. Acreditamos também que a modificação de alguns parâmetros dos descritores podem aprimorá-los. Em relação ao modelo *Perceptron* Multicamadas, acreditamos que a utilização de outras técnicas (e.g., *dropout*) podem ser exploradas para melhorar a precisão de classificação do modelo.

A respeito disso, é possível observar que os melhores resultados alcançados com cada descritor sobre os classificadores *Perceptron* Multicamadas foram sobre o conjunto de dados visuais FEI, que, conforme já mencionado, o tamanho das imagens deste conjunto de dados visuais são um pouco mais significativas. A pontuação mais alta obtida foi com o classificador *Perceptron* Multicamadas denominado ML6 sobre o conjunto de dados visuais FEI com o Descritor Binário Local AKAZE. Em muitos casos, usando outros classificadores *Perceptron* Multicamadas, é possível observar que os modelos com cada conjunto de dados visuais não produziram resultados satisfatórios para tarefas de reconhecimento e classificação. Em geral, eles produziram uma baixa taxa de precisão, ou uma alta taxa de *overfitting* ocorre principalmente com o conjunto de dados visuais JAFFE. Os classificadores *Perceptron* Multicamadas chamados MLP3, MLP4, MLP5 e MLP6 sobre o conjunto de dados visuais FEI com o Descritor Binário Local ORB apresentam uma alta taxa de *underfitting*.

Outros Descritores Binários Locais que se destacam com o conjunto de dados visuais FEI foram o BRIEF e BRISK; no entanto, o descritor que alcançou proeminência e fornece resultados excelentes em relação à invariância à rotação, escala, iluminação e ponto de vista em muitos outros trabalhos, é o BRISK. Desta forma, optamos por utilizar o descritor BRISK nas próximas etapas deste trabalho. Na Figura 46, é possível observar a Matriz de Confusão com o modelo MLP5 sobre o conjunto de dados visuais FEI com o Descritor Binário Local BRISK.

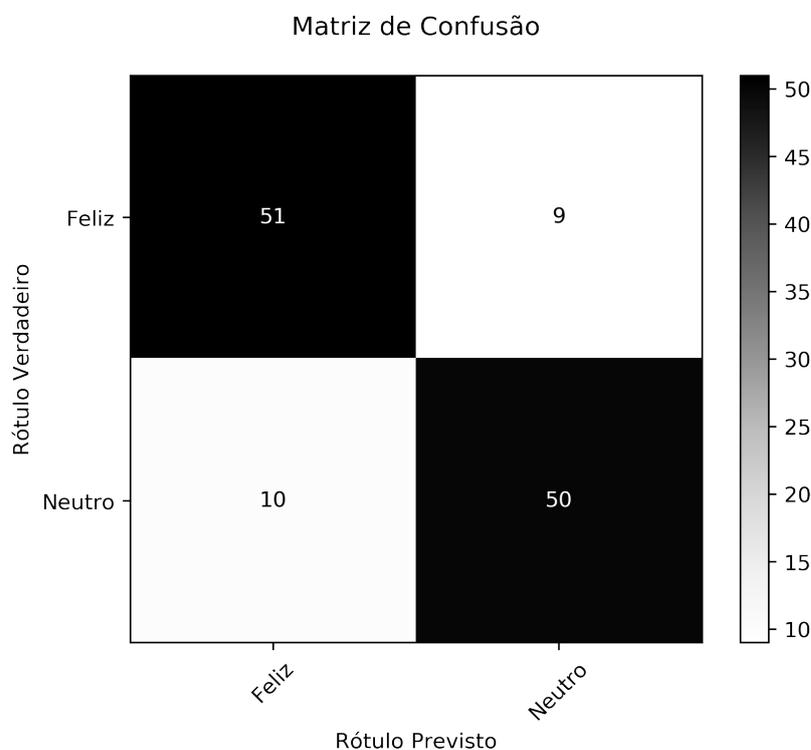


Figura 46 – Matriz de Confusão sobre conjunto de dados visuais FEI com o descritor BRISK com o modelo ML5. Fonte: Elaborado pela autora.

Assim, o modelo MLP5 sobre o conjunto de dados visuais FEI com BRISK previu corretamente a emoção feliz 51 vezes (verdadeiro positivo) e neutro 50 vezes corretamente (verdadeiro negativo). Em contraste, previu incorretamente a emoção feliz 9 vezes (falso positivo) e neutro 10 vezes incorretamente (falso negativo). Neste contexto, o modelo MLP5 sobre o conjunto de dados visuais FEI com BRISK teve uma precisão de 84%, pois obteve 101 das 120 previsões corretas. Na Tabela 11, é possível visualizar os valores alcançados das principais métricas de classificação: Precisão, *Recall*, *F1-Score* e Suporte. Com o último, fica claro que o conjunto de testes do conjunto de dados visuais FEI está bem balanceado.

Tabela 11 – Média (%) dos valores de Precisão, *Recall*, *F1-Score* e Suporte sobre o conjunto de dados visuais FEI com o descritor BRISK com o modelo ML5.

Fonte: Elaborado pela autora.

Algoritmos	Dataset	Rótulo	Modelo <i>Perceptron</i> Multicamadas			
			MLP5			
			Precisão	<i>Recall</i>	<i>F1-Score</i>	Suporte
BRISK	FEI	Feliz	0.84	0.85	0.84	60
		Neutro	0.85	0.85	0.84	60
		Média	<b>0.84</b>	<b>0.84</b>	<b>0.84</b>	<b>120</b>

Além disto, é possível analisar a curva da taxa de perda de treinamento dos seis classificadores *Perceptron* Multicamadas sobre o conjunto de dados visuais FEI com o Descritor Binário Local BRISK na Figura 47.

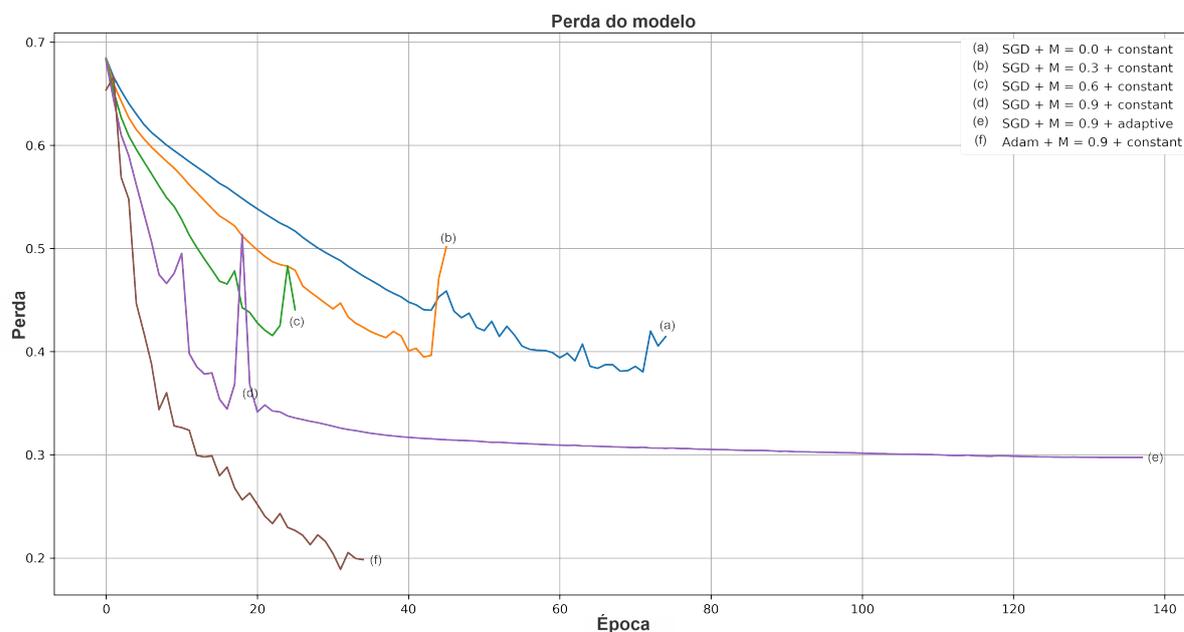


Figura 47 – Taxa de perda de treinamento sobre o conjunto de dados visuais FEI com o descritor BRISK com o modelo ML5. Fonte: Elaborado pela autora.

Note que, na Figura 47, não é possível visualizar a curva (d) referente ao modelo ML4, isso ocorre porque a curva (d) segue o mesmo caminho da curva (e) referente ao

modelo ML5 até cerca de 20 épocas, pois os parâmetros de configuração de ambos os modelos são semelhantes.

Ressaltamos que o objetivo deste experimento não foi simplesmente atingir o melhor índice de acurácia pelo classificador *Perceptron* Multicamadas, mas conhecer o microcomputador Jetson Nano e obter uma visão geral de seu funcionamento e poder computacional, além de avaliar o desempenho dos descritores ao lado do classificador *Perceptron* Multicamadas aqui discutido. Além disto, experimentalmente, demonstramos a viabilidade de produzir resultados promissores combinando a abordagem de Saco de Características Visuais com classificador *Perceptron* Multicamadas. Portanto, empiricamente, podemos assumir que os descritores computados gerados por um Descritor Binário Local ao lado de uma arquitetura de Redes Neurais Convolucionais podem alcançar resultados satisfatórios na etapa Detecção de Característica, e reformulados na etapa Extração de Característica em filtros convolucionais de arquitetura de Rede Neural Convolucional proposta nesta Dissertação.

## 6.2 Rede Neural Convolucional de Descritores

A linguagem Python 3.7.12, bem como os pacotes OpenCV 4.1.2 e Keras 2.6.0 com TensorFlow 2.6.0 como *back-end* são usados para realizar os experimentos apresentados nesta Seção, junto a um hardware composto por uma CPU Intel (R) Xeon (R) @ 2,20 GHz, GPU NVIDIA Tesla P100-PCIE-16GB, 25.5 GB de memória RAM. Implementação e futuras atualizações estarão disponíveis no repositório do GitHub que pode ser consultado no Apêndice C.

### 6.2.1 Abordagem Proposta

Em nosso último experimento, estávamos preocupados em realizar a avaliação de uma abordagem de Saco de Características Visuais, extraindo características através de Descritores de Características Locais e Descritores Binários Locais no microcomputador Jetson Nano para as tarefas de reconhecimento e classificação em seis conjuntos de dados visuais por meio do classificador *Perceptron* Multicamadas.

Este trabalho preliminar foi imprescindível, pois esperávamos que o experimento e simulações realizadas nos levassem a uma escolha exitosa no que diz respeito a seleção de um descritor, que seria, posteriormente, utilizado em trabalhos futuros na etapa de Detecção de Características e reformulado na etapa de Extração de Características em filtros convolucionais da arquitetura de Rede Neural Convolucional proposta.

Portanto, ao final do experimento e simulações realizadas anteriormente o descritor a ser abordado em trabalhos futuros foi definido e, com isto, foi possível avançar para as próximas etapas, dando assim, continuidade as tarefas previstas para o desenvolvimento

desta Dissertação de mestrado. Optamos em recorrer ao algoritmo BRISK como principal descritor a ser utilizado. Entretanto, um ponto importante acerca da nossa abordagem é que todos os descritores, sejam eles Descritores de Características Locais ou Descritores Binários Locais, podem ser integrados a camada de Detecção de Características Locais da Desc-Net.

Outro ponto importante acerca da nossa abordagem é que a arquitetura Desc-Net opera apenas com imagens em escala de cinza, visto que existe uma limitação ao utilizar a camada de Detecção de Características Locais, onde os detectores de pontos-chave e descritores de características só manipulam este tipo de imagem. Portanto, ao supor que temos uma imagem de entrada em escala de cinza e que seu tamanho é  $10 \times 10$ , a arquitetura verá uma matriz representativa de dimensão  $10 \times 10 \times 1$ , onde o valor 1 refere-se a um componente pré-estabelecido de uma imagem — chamado Canal e conhecido em inglês como *Channel* — neste caso, o valor 1 refere-se a escala de cinza, onde, para este canal, será atribuído valores de intensidade de pixels no intervalo de 0 à 255 (onde 0 representa pontos pretos e 255 representa pontos brancos) para cada ponto da matriz representativa de uma imagem. Para o propósito deste trabalho, de modo a reduzir ao máximo o poder computacional necessário na etapa de processamento de imagens, consideramos este ponto interessante, ao passo que é gerando apenas uma única matriz  $2D$  representativa para cada imagem ao contrário de uma imagem colorida, onde sua matriz representativa seria expressa pela dimensão  $10 \times 10 \times 3$ , onde o valor 3 refere-se ao padrão de modo de cor RGB (em inglês, *Red*, *Green*, *Blue*).

## 6.2.2 Detalhes da Implementação

Optamos em avaliar a eficiência das camadas de Detecção de Características Locais e Convolução de Descritor Local propostas, acopladas em uma arquitetura de ResNet-152, de modo a comparar o seu desempenho com camadas convolucionais tradicionais, nos conjuntos de dados visuais MNIST e CIFAR-10. A ResNet é composta principalmente por dois blocos, o bloco padrão usados em ResNets, conhecido como bloco de identidade e o bloco convolucional. O bloco de identidade é usado quando a ativação de entrada tem a mesma dimensão que a ativação de saída, enquanto o bloco convolucional é usado quando as dimensões de entrada e saída não coincidem. Para estes primeiros experimentos, optamos em reformular apenas o bloco convolucional com blocos DescNet.

Conceitualmente, a DescNet proposta neste trabalho pode ser facilmente implementada em qualquer estrutura de Aprendizado Profundo existente. Além disso, como os pesos são esparsos e binários, a operação de convolução pode ser realizada puramente por meio de adições e subtrações. Isso leva à economia do ponto de vista computacional e também da memória. Como topologia da rede, foi utilizado a arquitetura ResNet apresentado por HE et al., portanto, a mesma arquitetura utilizada para implementar a Rede Neural

Convolutacional tradicional serviu como linha de base para a implementação da DescNet proposta neste trabalho. Portanto, para fazer uma comparação junta entre as arquiteturas, o número de filtros convolucionais, camadas convolucionais, unidades escondidas na camada totalmente conectada, foram os mesmos em ambos os modelos de redes, embora que, com pesos esparsos e binários para a DescNet, gerados seguindo o procedimento descrito na Seção 4.2.1.2, e com pesos densos para a ResNet-152.

Aplicamos a técnica de Validação Cruzada (em inglês *Cross Validation*) dividindo aleatoriamente as amostras de treinamento em conjuntos disjuntos de tamanhos iguais contendo aproximadamente as mesmas proporções de classes de cada conjunto de dados visuais, formando assim, um conjunto de dados visuais de validação. Além disto, devido à natureza escassa dos conjuntos de dados visuais utilizados, aplicamos o processo de *Data Augmentation* — DA, processo este sendo aplicado ao conjunto de treinamento com a finalidade de gerar sub-amostras diferentes para cada imagem, com a finalidade de alimentar, e conseqüentemente, melhorar os resultados do classificador, além de ajudar a prevenir *Overfitting* que consiste em um modelo que se ajusta bem ao conjunto de treinamento, entretanto, é ineficaz de generalizar-se bem a dados até então não vistos.

Inicialmente foi realizado teste com diversos algoritmos de otimização, como *RMS-prop*, *SGB* e *Adagrad*, entretanto o que demonstrou melhores resultados foi o *Adam*. *Adam Optimizer* é um algoritmo de otimização como alternativa ao algoritmo Gradiente Estocástico de Descida (em inglês, *Stochastic Gradient Descent* — SGD) utilizado para realizar a atualização dos pesos de forma iterativa de acordo como o conjunto de dados de treinamento. Também usamos os mesmos dados e parâmetros em termos de taxa de aprendizado inicial  $1e - 3$  e programação da taxa de aprendizado. Ainda, aplicamos a técnica de Taxas Aprendizado Cíclicos (em inglês, *Cyclical Learning Rate* — CLR), ou seja, durante a etapa de treinamento, a taxa de aprendizado oscila para frente e para trás entre dois limites aumentando lentamente a taxa de aprendizado após cada atualização em lote.

Desta forma, usamos uma técnica para encontrar uma taxa de aprendizado ideal, chamada Teste de Variação da Taxa de Aprendizado (em inglês *Learning Rate Tange Test* — LRRT), sendo uma técnica que nos auxilia a descobrir os melhores valores de taxa de aprendizado que podem ser usados para treinar um modelo sem divergência. Deste modo, configuramos um intervalo entre  $1e - 8$  e  $1e0 = 1$ , carregamos os modelos pré-treinados, e após o treinamento de 10 épocas, chegamos à conclusão de que uma diminuição na perda (ou seja, melhoria do modelo), começando aproximadamente na taxa de aprendizado  $1e - 4$  e terminando aproximadamente na taxa de aprendizado  $1e - 2$ , logo após, há um aumento excessivo na perda. Portanto, selecionamos o valor de  $1e - 4$  como o limite inferior do ciclo, que nada mais é do que a taxa de aprendizado inicial, e o valor de  $1e - 2$  como o limite superior do ciclo, valor este que define a amplitude do ciclo. Na Figura 48 pode ser

observado o gráfico gerado do Teste de Variação da Taxa de Aprendizado.

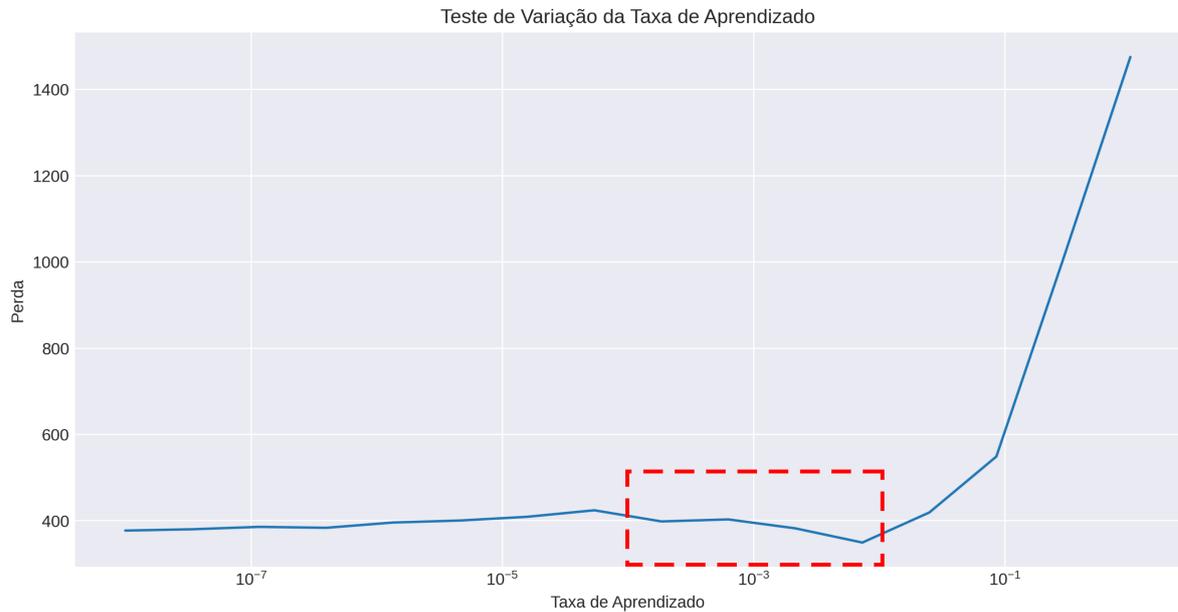


Figura 48 – Teste de Variação da Taxa de Aprendizado. Fonte: Elaborado pela autora.

Além disto, usamos ReLU (vide Equação 5.1) em vez de sigmóide como função não-linear para convergência mais rápida dos experimentos.

### 6.2.3 Resultados

Com os resultados obtidos neste experimento, avaliamos o desempenho de classificação usando as acurácias Top-1 e Top-5. Onde, a acurácia Top-1 é caracterizada pela precisão convencional, ou seja, exatamente a resposta esperada (com maior probabilidade). Enquanto a acurácia Top-5, considera que, qualquer uma das cinco primeiras respostas com maior probabilidade devem corresponder à resposta esperada. Isto porque, a priori, a resposta esperada encontra-se entre as 5 primeiras estimativas. A acurácia Top-5 torna-se uma métrica razoável, isto porque uma única imagem pode conter mais detalhes do que um objeto de interesse.

De modo a avaliar o desempenho do modelo, analisaremos as curvas de aprendizado, que consistem em gráficos que mostram as mudanças de desempenho do modelo ao longo de épocas, onde o modelo será avaliado no conjunto de dados de treinamento e em um conjunto de dados de validação.

#### 6.2.3.1 Resultados no conjunto de dados visuais MNIST

A Figura 49 apresenta a curva de precisão da classificação para as acurácias Top-1 e Top-5, e a curva de perda no conjunto de dados visuais MNIST na etapa de treinamento no conjunto de dados de treinamento e validação.

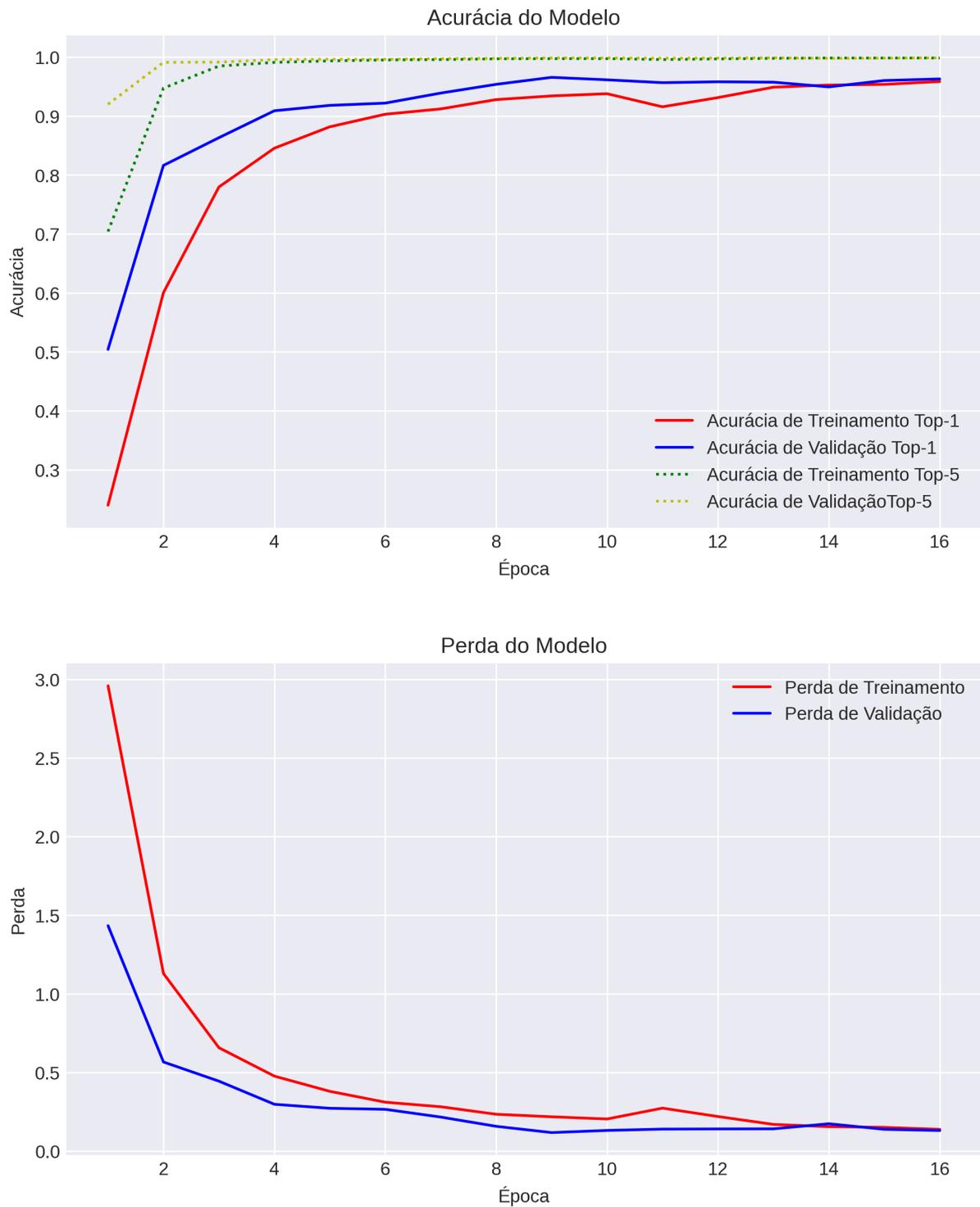


Figura 49 – Curvas de aprendizado no conjunto de dados visuais MNIST. Fonte: Elaborado pela autora.

A partir das curvas de aprendizado observadas na Figura 49, as curvas de precisão de classificação entre os conjuntos de dados de treinamento e validação ao longo das épocas encontram-se bem próximas uma da outra, desta forma, o desempenho do modelo cresce ao longo das épocas, indicando que o mesmo está melhorando com a experiência.

Ainda na Figura 49, ao analisar a curva da taxa de perda, podemos observar o efeito da taxa de aprendizado na perda, e podemos dizer que a taxa de aprendizado definida, foi uma boa escolha de parâmetros, isto porque a perda diminui com o tempo, e com isso, o modelo está aprendendo, ou seja, isto indica quão bem o modelo está se ajustando os dados de treinamento. Enquanto a perda de validação indica quão bem o modelo está se ajustando os dados de validação, ou seja, aos novos dados. Podemos observar ainda, que entre as épocas 14 à 16 ambas as perdas (de treinamento e validação) diminuiriam a um ponto de estabilidade, onde a lacuna de generalização do modelo é mínima, ou seja, quase zero. Com isto, podemos dizer que os parâmetros de treinamento escolhidos levou o modelo para MNIST, um modelo de bom ajuste.

Ao avaliar o modelo no conjunto de dados de teste, o modelo DescNet no conjunto de dados visuais MNIST alcança acurácias Top-1 e Top-5 de 56% e 90% respectivamente, enquanto atinge uma taxa de perda de 1.94%. Para este conjunto de dados visuais, foi definido um tamanho do lote (em inglês *batch size*) de 64, desta forma, isto significa que o modelo conseguirá enxergar todo o conjunto de dados de treinamento em 843 lotes (em inglês *batches*), ou seja, para executar 1 época, é necessário que ocorra 843 lotes. Portanto, o modelo executa aproximadamente 13.488 iterações por época. Aplicando a técnica de Validação Cruzada, foi formado um conjunto de dados visuais de validação com 6.000 amostras. Enquanto a quantidade de amostras para o conjunto de dados visuais de treinamento foi formado por 54.000 amostras e o conjunto de dados visuais de teste sendo formado por 10.000 amostras.

Comparando os resultados de precisão de classificação entre conjunto de dados visuais de validação e teste, observamos uma grande lacuna de generalização do modelo. Podemos dizer que este comportamento pode estar relacionado pela quantidade de amostras presentes em cada conjunto de dados visuais, neste caso, a quantidade de amostras presentes no conjunto de dados visuais de validação é menor que a quantidade de amostras presentes no conjunto de dados visuais de teste. Outro ponto a ser considerado, é que talvez o modelo esteja reproduzindo *overfitting* sobre o conjunto de dados visuais de validação, ou seja, o modelo está se ajustando muito bem neste conjunto de dados, entretanto, se torna ineficaz ao prever novos resultados. Estes casos, ajudam explicar as diferentes precisões de classificação obtidas. Podemos observar a matriz de confusão do modelo, na Figura 50, onde, a diagonal principal apresenta as previsões classificadas corretamente. Enquanto o restante da matriz apresenta as previsões classificadas erroneamente.

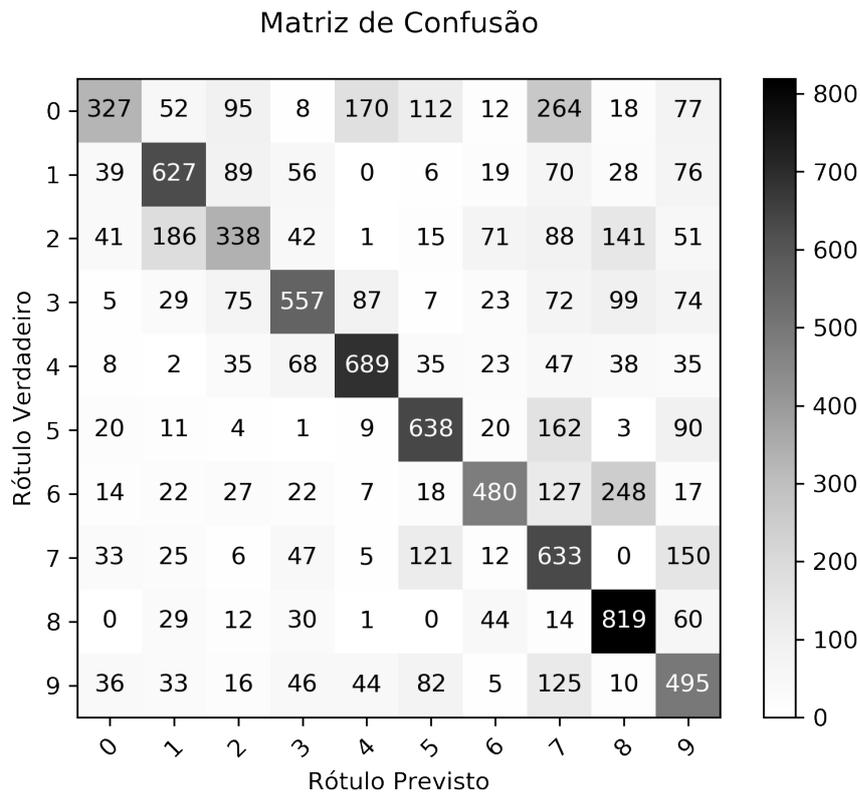


Figura 50 – Matriz de Confusão sobre conjunto de dados visuais MNIST. Fonte: Elaborado pela autora.

Na Tabela 12, é possível visualizar os valores alcançados pelas principais métricas de classificação: Precisão, *Recall*, *F1-Score* e Suporte.

Tabela 12 – Média (%) das métricas de precisão, *Recall*, *F1-Score* e suporte sobre o conjunto de dados visuais MNIST. Fonte: Elaborado pela autora.

Rótulo	Precisão	<i>Recall</i>	<i>F1-Score</i>	Suporte
0	0.63	0.29	0.39	1135
1	0.62	0.62	0.62	1010
2	0.48	0.35	0.40	974
3	0.64	0.54	0.58	1028
4	0.68	0.70	0.69	980
5	0.62	0.67	0.64	958
6	0.68	0.49	0.57	982
7	0.40	0.61	0.48	1032
8	0.58	0.81	0.68	1009
9	0.44	0.55	0.49	892
<b>Acurácia</b>	<b>0.56</b>	<b>0.56</b>	<b>0.56</b>	<b>0.56</b>
<b>Média</b>	<b>0.58</b>	<b>0.56</b>	<b>0.55</b>	<b>10000</b>

## 6.2.3.2 Resultados no conjunto de dados visuais CIFAR-10

A Figura 51 apresenta a curva de precisão da classificação para as acurácias Top-1 e Top-5, e a curva de perda no conjunto de dados visuais CIFAR-10 na etapa de treinamento no conjunto de dados de treinamento e validação.

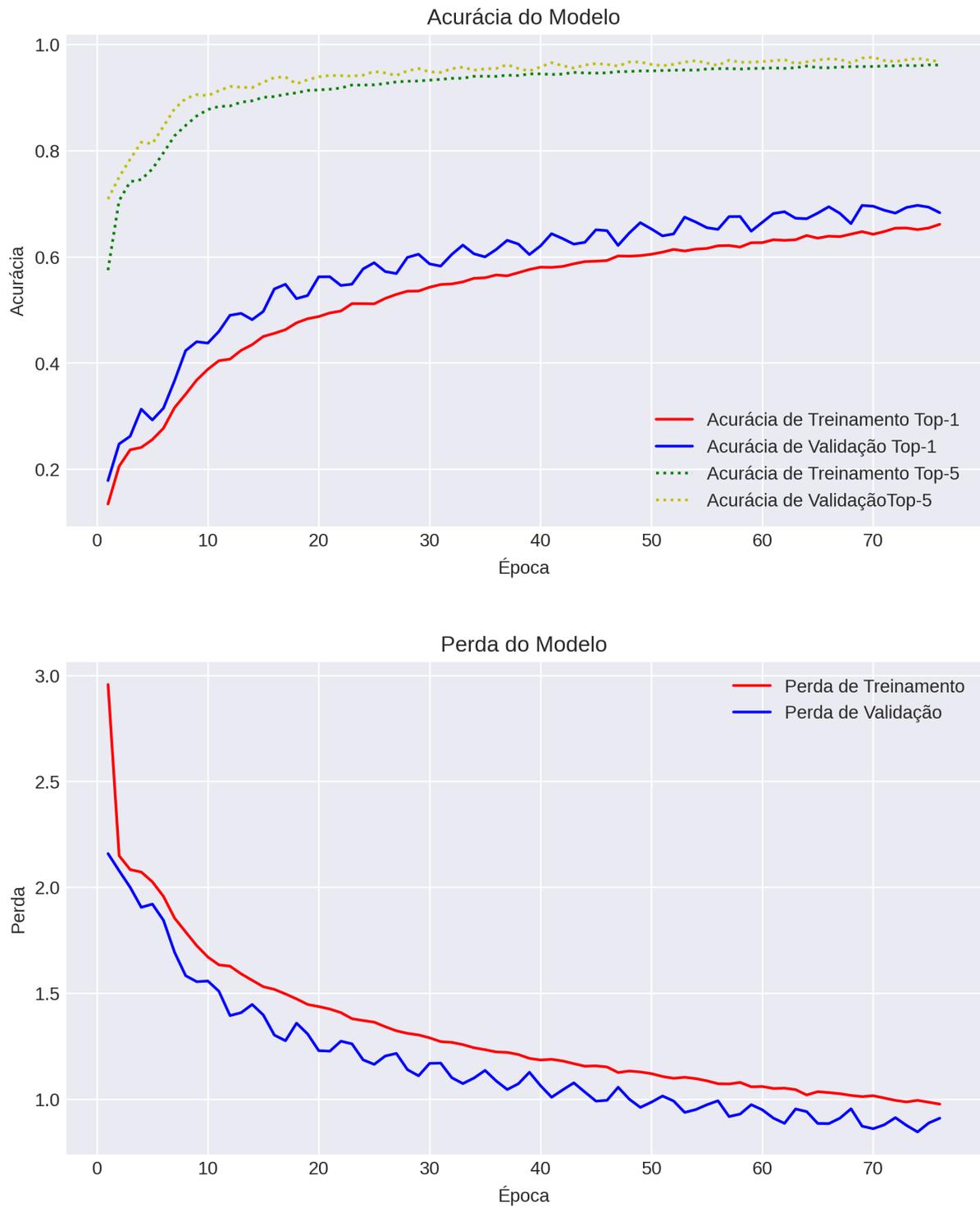


Figura 51 – Curvas de aprendizado no conjunto de dados visuais CIFAR-10. Fonte: Elaborado pela autora.

A partir das curvas de aprendizado observadas na Figura 51, as curvas de precisão de classificação entre os conjuntos de dados de treinamento e validação ao longo das épocas encontram-se bem próximas uma da outra, desta forma, o desempenho do modelo cresce ao longo das épocas, indicando que o mesmo está melhorando com a experiência. Entretanto, após 70 épocas, o modelo não conseguiu superar, para a acurácia Top-1, cerca de 65%.

Ainda na Figura 51, ao analisar a curva da taxa de perda, podemos observar o efeito da taxa de aprendizado na perda, e podemos dizer que a taxa de aprendizado definida parece ter relativamente sido uma boa escolha de parâmetros, isto porque a perda diminui com o tempo. Portanto, a curva da taxa de perda do conjunto de dados visuais de treinamento está boa, em contrapartida, a taxa de perda do conjunto de dados visuais de validação se move com ruídos ao longo das épocas. Isto indica que a taxa de perda do conjunto de dados visuais de validação é melhor em comparação com a taxa de perda do conjunto de dados visuais de treinamento, portanto, apesar de a quantidade de amostras no conjunto de dados visuais de validação ser pequena, o modelo consegue ter um bom desempenho nestas poucas amostras.

Ao avaliar o modelo no conjunto de dados de teste, o modelo DescNet no conjunto de dados visuais CIFAR-10 alcança acurácias Top-1 e Top-5 de 23% e 65% respectivamente, enquanto atinge uma taxa de perda de 3.20%. Para este conjunto de dados visuais, foi definido um tamanho do lote de 64. Isto significa que o modelo conseguirá enxergar todo o conjunto de dados de treinamento em 703 lotes, ou seja, para executar 1 época, é necessário que ocorra 843 lotes. Portanto, o modelo executa aproximadamente 49.210 iterações por época. Aplicando a técnica de Validação Cruzada, foi formado um conjunto de dados visuais de validação com 5.000 amostras. Enquanto a quantidade de amostras para o conjunto de dados visuais de treinamento foi formado por 45.000 amostras e o conjunto de dados visuais de teste sendo formado por 10.000 amostras.

Comparando os resultados de precisão de classificação entre conjunto de dados visuais de validação e teste, observamos uma grande lacuna de generalização do modelo, assim como ocorreu com o conjunto de dados visuais MNIST. Podemos dizer que este comportamento pode estar relacionado pela quantidade de amostras presentes em cada conjunto de dados visuais, neste caso, a quantidade de amostras presentes no conjunto de dados visuais de validação é menor que a quantidade de amostras presentes no conjunto de dados visuais de teste. Outro ponto a ser considerado, é que talvez o modelo esteja reproduzindo *overfitting* sobre o conjunto de dados visuais de validação, ou seja, o modelo está se ajustando muito bem neste conjunto de dados, entretanto, se torna ineficaz ao prever novos resultados, assim como ocorreu com o conjunto de dados visuais MNIST. Estes casos, ajudam explicar as diferentes precisões de classificação obtidas. Podemos observar a matriz de confusão do modelo, na Figura 52.

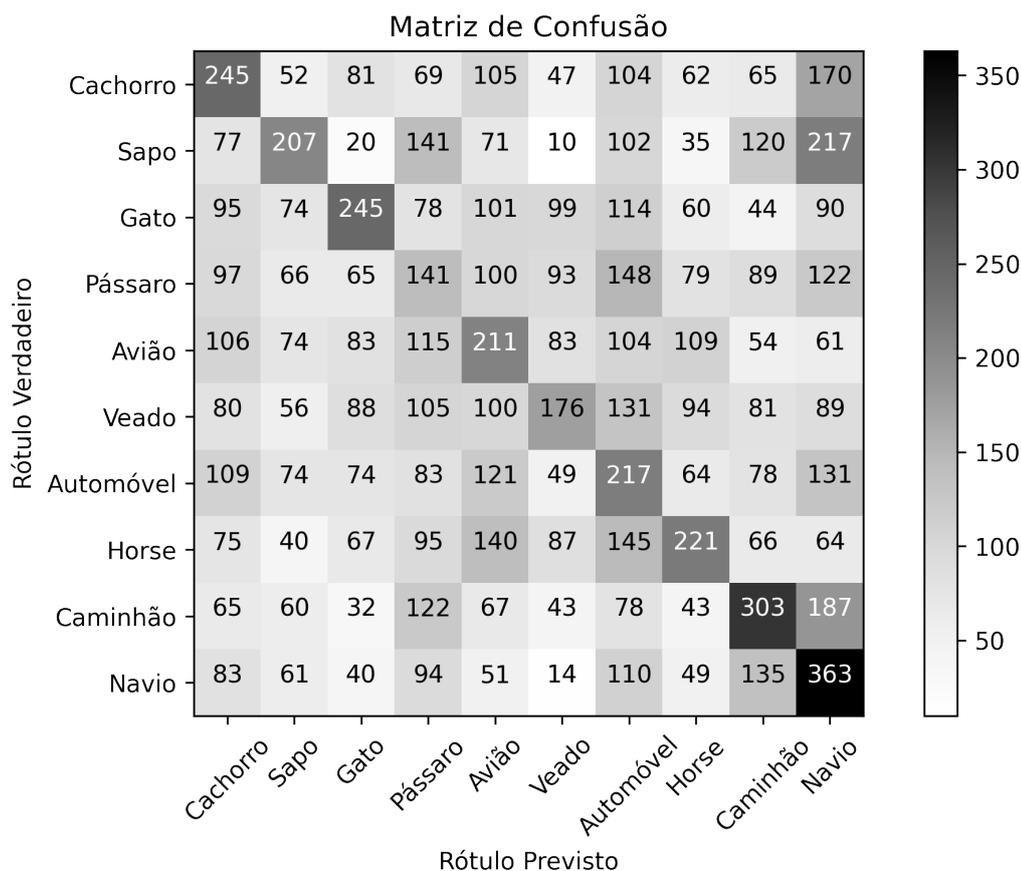


Figura 52 – Matriz de Confusão sobre conjunto de dados visuais CIFAR-10. Fonte: Elaborado pela autora.

Na Tabela 13, é possível visualizar os valores alcançados pelas principais métricas de classificação: Precisão, *Recall*, *F1-Score* e Suporte.

Tabela 13 – Média (%) das métricas de precisão, *Recall*, *F1-Score* e suporte sobre o conjunto de dados visuais CIFAR-10. Fonte: Elaborado pela autora.

Rótulo	Precisão	<i>Recall</i>	<i>F1-Score</i>	Suporte
<b>Cachorro</b>	0.24	0.25	0.24	1000
<b>Sapo</b>	0.27	0.21	0.23	1000
<b>Gato</b>	0.31	0.25	0.27	1000
<b>Pássaro</b>	0.14	0.14	0.14	1000
<b>Avião</b>	0.20	0.21	0.20	1000
<b>Veado</b>	0.25	0.18	0.21	1000
<b>Automóvel</b>	0.17	0.22	0.19	1000
<b>Cavalo</b>	0.27	0.22	0.24	1000
<b>Caminhão</b>	0.29	0.30	0.30	1000
<b>Navio</b>	0.24	0.36	0.29	1000
<b>Acurácia</b>	<b>0.23</b>	<b>0.23</b>	<b>0.23</b>	<b>0.23</b>
<b>Média</b>	<b>0.24</b>	<b>0.23</b>	<b>0.23</b>	<b>10000</b>

### 6.2.3.3 Comparação com Linha de Base

A Tabela 14 consolida os resultados obtidos da DescNet de melhor desempenho em comparação com sua linha de base (ResNet-152) correspondente.

Tabela 14 – Comparação entre DescNet e linha de Base. Fonte: Elaborado pela autora.

Conjunto de dados visuais	DescNet				Linha de Base			
	Top-1	Top-5	Parâmetros aprendíveis	Tempo de Treinamento	Top-1	Top-5	Parâmetros aprendíveis	Tempo de Treinamento
MNIST	56%	90%	50,307,978	01:01:02	67%	89%	58,295,178	02:59:54
CIFAR-10	23%	65%	50,307,978	05:29:46	38%	77%	58,295,178	07:44:53

Comparando o número de parâmetros aprendíveis entre DescNet e sua linha de base, a DescNet economiza aproximadamente 14% de parâmetros aprendíveis nas camadas convolucionais, e executa a etapa de treinamento de forma acelerada em comparação com sua linha de base.

Entretanto, nos experimentos executados, não foi possível obter taxas de acurácias Top-1 e Top-5 superiores à linha de base, exceto no conjunto de dados visuais MNIST, enquanto para a acurácia Top-5 com o modelo DescNet foi atingido 1% a mais na taxa de classificação. Concluímos que as baixas taxas de acurácia obtidas pela DescNet em comparação com sua linha de base, se deu principalmente pela limitação ao se trabalhar com conjuntos de dados visuais onde o tamanho de suas amostras são pequenas, onde alguns descritores, assim como o BRISK obtêm resultados ínfimos em *patches* menores como comentado anteriormente na Seção 6.1.2.

## 6.3 Comentários Gerais

De forma geral, entendemos que ainda podem ser aplicadas diversas técnicas de modo a aprimorar os resultados e performance da DescNet proposta. Como, por exemplo, (a) retreinar o modelo com uma taxa de aprendizado mais alta do que a última utilizada; (b) aplicar um treinamento utilizando o método RigL (EVCÍ et al., 2020); (c) aplicar uma função de ativação com saídas negativas utilizando o método ELU (CLEVERT; UNTERTHINER; HOCHREITER, 2015); (d) regular os parâmetros do modelo até encontrar um bom ajuste; (e) avaliar outros conjuntos de Descritores de Características Locais e Descritores Binários Locais junto a DescNet aplicados a diferentes tamanhos de *patches*, entre outras técnicas.

Exploramos ainda, a possibilidade de obter resultados em conjuntos de dados visuais maiores, como, por exemplo, *Canadian Institute For Advanced Research* — CIFAR-100 (KRIZHEVSKY; HINTON et al., 2009) e *ImageNet Large Scale Visual Recognition Challenge 2017* — ILSVRC-2017 (RUSSAKOVSKY et al., 2015). E futuramente, obter resultados em conjuntos de dados visuais voltados para o problema de Detecção de Fechamento de *Loop*, como, por exemplo, *New College* e *City Centre* (CUMMINS; NEWMAN,

2008), *LipIndoor* e *LipOutdoor* (ANGELI et al., 2008), *Bovisa* (LABBE; MICHAUD, 2013), e *KITTI* (XIE et al., 2016).

Por fim, neste Capítulo, foram apresentados os resultados dos experimentos e simulações realizados no decorrer do desenvolvimento desta Dissertação. A seguir, serão apresentadas as considerações finais desta Dissertação, bem como, as contribuições científicas realizadas durante a evolução deste trabalho, a conclusão e sugestões para trabalhos futuros.

## 7 CONSIDERAÇÕES FINAIS

As contribuições científicas, fruto desta Dissertação de mestrado, podem ser consultadas no Anexo A.

### 7.1 Conclusão

Neste trabalho, propomos a reformulação das camadas convolucionais de Redes Neurais Convolucionais tradicionais por meio de Descritores Binários Locais. Desta forma, apresentamos duas novas camadas para arquiteturas profundas — Detecção de Características Locais (em inglês, *Local Feature Detection* — LFD), como uma primeira camada, e Convolução de Descritor Local (em inglês, *Local Descriptor Convolution* — LDC), como uma alternativa viável e eficiente às camadas convolucionais. Os Descritores Binários Locais inspiram os fundamentos do *design* de camadas das camadas de Detecção de Características Locais e Convolução de Descritor Local, denominada neste trabalho de Rede Neural Convolucional de Descritores (em inglês, *Descriptor Convolutional Neural Network* — DescNet).

As bases teóricas que sustentaram a adaptação foram apresentadas. Experimentos foram realizados para comparar sua eficiência com um modelo de Rede Neural Convolucional tradicional. Alcançamos resultados significativos sobre a camada convolucional padrão em conjuntos de dados visuais competitivos (MNIST e CIFAR-10) enquanto permite economia no número de parâmetros do modelo e, conseqüentemente, economia computacional significativa, tornando a DescNet um modelo aplicável em ambientes reais com recursos escassos e limitados.

Nossa abordagem é promissora, onde os métodos propostos possuem o potencial de realizar a tarefa de Detecção de Fechamento de *Loop* de um sistema VSLAM eficaz e com desempenho eficiente nas próximas etapas deste trabalho.

### 7.2 Trabalhos Futuros

Apresentamos como proposta de trabalho futuro, a integração de um modelo híbrido de arquitetura de Rede Neural Artificial — a Rede Convolucional Recorrente de Longo Prazo com blocos DescNet com um sistema VSLAM capaz de solucionar o problema de Detecção de Fechamento de *Loop*. Dado que o principal objetivo da Detecção de Fechamento de *Loop* está em corrigir a deriva, ou seja, corrigir a incerteza nos cálculos estimados acumulada no decorrer da trajetória de uma plataforma robótica móvel através da obtenção de imagens sequenciais (vídeo) e processadas pela adaptação do modelo

híbrido de arquitetura de Rede Neural Artificial proposta. Na Figura 53, é possível verificar o fluxograma do sistema proposto a ser desenvolvido em trabalho futuros, que consiste nas seguintes etapas:

- **Entrada:** O objetivo desta etapa consiste na configuração e calibração da câmera monocular que será usada para obter as imagens sequenciais (vídeo).

Esta etapa consiste no pré-processamento das imagens sequenciais obtidas, como alteração do padrão de modo de cor RGB (mais comum) para Escala de cinza (Descritores de Características Locais e Descritores Binários Locais operam apenas com imagens de entrada no padrão de modo de cor em escala de cinza). Consiste ainda na aplicação de redimensionamento das imagens sequenciais obtidas em diferentes tamanhos.

- **Representação e Extração de Característica:** O objetivo desta etapa consiste em executar a gerar a representação de característica por meio da camada de Detecção de Características Locais da DescNet, seguida pela extração de características obtida por meio da camada de Convolação de Descriptor Local da DescNet.

O objetivo final desta etapa consiste na adaptação de um modelo híbrido de arquitetura de Rede Neural Artificial — Rede Convolutacional Recorrente de Longo Prazo com blocos DescNet. Esperamos projetar, ao mesmo tempo, um modelo com eficiência e desempenho computacional eficiente;

- **Fechamento de *Loop*:** O objetivo desta etapa consiste na implementação da Detecção de Fechamento de *Loop* para um sistema VSLAM. Portanto, a Detecção de Fechamento de *Loop* será baseada no modelo híbrido de arquitetura de Rede Neural Artificial desenvolvido na etapa anterior.

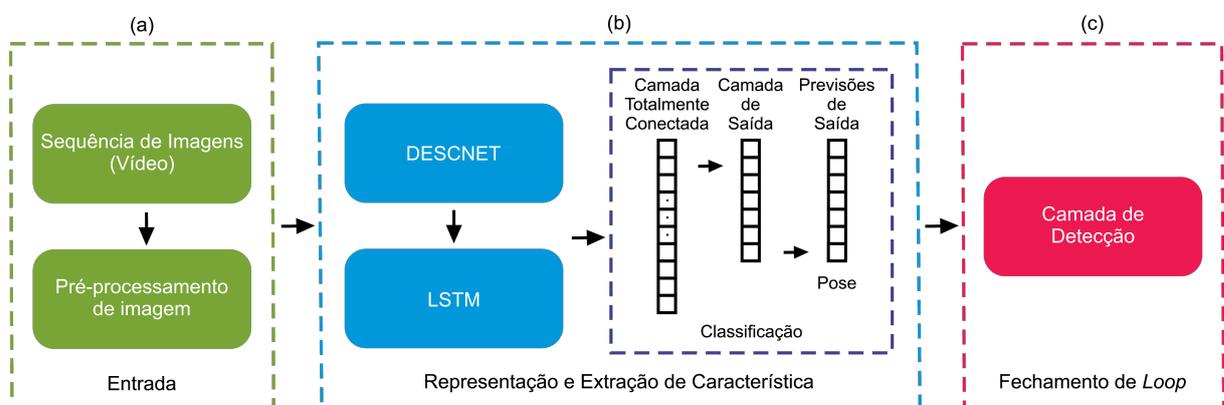


Figura 53 – Sistema proposto de Detecção de Fechamento de *Loop* baseado no modelo híbrido de arquitetura de Rede Neural Artificial proposta. Fonte: Elaborado pela autora.

## Agradecimentos

Este estudo foi parcialmente financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior — Brasil (CAPES) — Código Financeiro 001.

Este estudo foi realizado com o apoio do Programa de Cooperação Acadêmica em Defesa Nacional (PROCAD-DEFESA).

Agradecemos ao Simpósio Latino-Americano de Robótica/Simpósio Brasileiro de Robótica (LARS/SBR 2020) pelo prêmio de Terceiro Melhor Artigo concedido ao artigo intitulado: “Comparative Evaluation of Feature Descriptors Through Bag of Visual Features with Multilayer Perceptron on Embedded GPU System” (RAIBOLT; ANGONESE; ROSA, 2020).

## REFERÊNCIAS

- ABDEL-HAKIM, A. E.; FARAG, A. A. C sift: A sift descriptor with color invariant characteristics. In: IEEE. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. [S.l.], 2006. v. 2, p. 1978–1983. 34
- ALAHY, A.; ORTIZ, R.; VANDERGHEYNST, P. Freak: Fast retina keypoint. In: IEEE. *2012 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.], 2012. p. 510–517. 9, 34, 45, 46
- ALCANTARILLA, P. F.; BARTOLI, A.; DAVISON, A. J. Kaze features. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2012. p. 214–227. 9, 34, 39, 40
- ALCANTARILLA, P. F.; SOLUTIONS, T. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Trans. Patt. Anal. Mach. Intell.*, TrueVision Solutions, v. 34, n. 7, p. 1281–1298, 2011. 34, 46
- ANGELI, A.; FILLIAT, D.; DONCIEUX, S.; MEYER, J.-A. Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, IEEE, v. 24, n. 5, p. 1027–1037, 2008. 105
- ANWER, R. M.; KHAN, F. S.; WEIJER, J. van de; MOLINIER, M.; LAAKSONEN, J. Binary patterns encoded convolutional neural networks for texture recognition and remote sensing scene classification. *ISPRS journal of photogrammetry and remote sensing*, Elsevier, v. 138, p. 74–85, 2018. 60, 65
- BAILEY, T.; DURRANT-WHYTE, H. Simultaneous localization and mapping (slam): Part ii. *IEEE robotics & automation magazine*, IEEE, v. 13, n. 3, p. 108–117, 2006. 20
- BARROSO-LAGUNA, A.; RIBA, E.; PONSA, D.; MIKOLAJCZYK, K. Key. net: Keypoint detection by handcrafted and learned cnn filters. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 5836–5844. 61, 65
- BASULTO-LANTSOVA, A.; PADILLA-MEDINA, J. A.; PEREZ-PINAL, F. J.; BARRANCO-GUTIERREZ, A. I. Performance comparative of opencv template matching method on jetson tx2 and jetson nano developer kits. In: IEEE. *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.], 2020. p. 0812–0816. 63
- BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. In: SPRINGER. *European conference on computer vision*. [S.l.], 2006. p. 404–417. 9, 34, 37, 38
- BAYRAKTAR, E.; BOYRAZ, P. Analysis of feature detector and descriptor combinations with a localization experiment for various performance metrics. *Turkish Journal of Electrical Engineering and Computer Science*, v. 25, n. 3, p. 2444–2454, 2017. 59
- BEALE, R.; JACKSON, T. *Neural Computing-an introduction*. [S.l.]: CRC Press, 1990. 9, 50

- BEKELE, D.; TEUTSCH, M.; SCHUCHERT, T. Evaluation of binary keypoint descriptors. In: IEEE. *2013 IEEE International Conference on Image Processing*. [S.l.], 2013. p. 3652–3656. 59
- BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, IEEE, v. 5, n. 2, p. 157–166, 1994. 56
- BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent dirichlet allocation. *Journal of machine Learning research*, v. 3, n. Jan, p. 993–1022, 2003. 30, 46
- BRESSON, G.; RAHAL, M.-C.; GRUYER, D.; REVILLOU, M.; ALSAYED, Z. A cooperative fusion architecture for robust localization: Application to autonomous driving. In: IEEE. *2016 IEEE 19th international conference on intelligent transportation systems (ITSC)*. [S.l.], 2016. p. 859–866. 21
- CALONDER, M.; LEPETIT, V.; OZUYSAL, M.; TRZCINSKI, T.; STRECHA, C.; FUA, P. Brief: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 34, n. 7, p. 1281–1298, 2011. 34, 41
- CALONDER, M.; LEPETIT, V.; STRECHA, C.; FUA, P. Brief: Binary robust independent elementary features. In: SPRINGER. *European conference on computer vision*. [S.l.], 2010. p. 778–792. 9, 34, 41
- CHATOUX, H.; LECELLIER, F.; FERNANDEZ-MALOIGNE, C. Comparative study of descriptors with dense key points. In: IEEE. *2016 23rd International Conference on Pattern Recognition (ICPR)*. [S.l.], 2016. p. 1988–1993. 59
- CHEN, H.; LIU, S.; ZHANG, H.; CHENG, W. Handwriting numerals recognition using convolutional neural network implemented on nvidia’s jetson nano. In: SPRINGER. *International Conference in Communications, Signal Processing, and Systems*. [S.l.], 2019. p. 529–535. 63
- CHENG, Y.; MAIMONE, M.; MATTHIES, L. Visual odometry on the mars exploration rovers. In: IEEE. *2005 IEEE International Conference on Systems, Man and Cybernetics*. [S.l.], 2005. v. 1, p. 903–910. 21
- CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 1251–1258. 24, 52
- CLEMENTE, L. A.; DAVISON, A. J.; REID, I. D.; NEIRA, J.; TARDÓS, J. D. Mapping large loops with a single hand-held camera. In: *Robotics: Science and Systems*. [S.l.: s.n.], 2007. v. 2, n. 2. 9, 23
- CLEVERT, D.-A.; UNTERTHINER, T.; HOCHREITER, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. 104
- CORTES, C.; VAPNIK, V. Support-vector networks. *Machine learning*, Springer, v. 20, n. 3, p. 273–297, 1995. 48

- COURBARIAUX, M.; BENGIO, Y.; DAVID, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, v. 28, p. 3123–3131, 2015. 61, 65
- CSORBA, M. *Simultaneous localisation and map building*. Tese (Doutorado) — University of Oxford Oxford, 1997. 20
- CSURKA, G.; DANCE, C.; FAN, L.; WILLAMOWSKI, J.; BRAY, C. Visual categorization with bags of keypoints. In: PRAGUE. *Workshop on statistical learning in computer vision, ECCV*. [S.l.], 2004. v. 1, n. 1-22, p. 1–2. 46, 91
- CUMMINS, M.; NEWMAN, P. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, SAGE Publications Sage UK: London, England, v. 27, n. 6, p. 647–665, 2008. 105
- DAI, Z.; HUANG, X.; CHEN, W.; HE, L.; ZHANG, H. A comparison of cnn-based and hand-crafted keypoint descriptors. In: IEEE. *2019 International Conference on Robotics and Automation (ICRA)*. [S.l.], 2019. p. 2399–2404. 60
- DONAHUE, J.; HENDRICKS, L. A.; GUADARRAMA, S.; ROHRBACH, M.; VENUGOPALAN, S.; SAENKO, K.; DARRELL, T. Long-term recurrent convolutional networks for visual recognition and description. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 2625–2634. 10, 57, 58
- DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, IEEE, v. 13, n. 2, p. 99–110, 2006. 20
- EVCI, U.; GALE, T.; MENICK, J.; CASTRO, P. S.; ELSSEN, E. Rigging the lottery: Making all tickets winners. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2020. p. 2943–2952. 104
- FARLEY, K. A.; WILLIFORD, K. H.; STACK, K. M.; BHARTIA, R.; CHEN, A.; TORRE, M. de la; HAND, K.; GOREVA, Y.; HERD, C. D.; HUESO, R. et al. Mars 2020 mission overview. *Space Science Reviews*, Springer, v. 216, n. 8, p. 1–41, 2020. 21
- FILLIAT, D. A visual bag of words method for interactive qualitative localization and mapping. In: IEEE. *Proceedings 2007 IEEE International Conference on Robotics and Automation*. [S.l.], 2007. p. 3921–3926. 62
- FLOSOS, G.; ZANDER, B. V. D.; LEIBE, B. Openstreetslam: Global vehicle localization using openstreetmaps. In: IEEE. *2013 IEEE International Conference on Robotics and Automation*. [S.l.], 2013. p. 1054–1059. 21
- FRIEDMAN, N.; GEIGER, D.; GOLDSZMIDT, M. Bayesian network classifiers. *Machine learning*, Springer, v. 29, n. 2-3, p. 131–163, 1997. 48
- GÁLVEZ-LÓPEZ, D.; TARDOS, J. D. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, IEEE, v. 28, n. 5, p. 1188–1197, 2012. 62
- GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 580–587. 52

- GOODFELLOW, I. J.; ERHAN, D.; CARRIER, P. L.; COURVILLE, A.; MIRZA, M.; HAMNER, B.; CUKIERSKI, W.; TANG, Y.; THALER, D.; LEE, D.-H. et al. Challenges in representation learning: A report on three machine learning contests. In: SPRINGER. *International Conference on Neural Information Processing*. [S.l.], 2013. p. 117–124. 11, 84
- GREFF, K.; SRIVASTAVA, R. K.; KOUTNÍK, J.; STEUNEBRINK, B. R.; SCHMIDHUBER, J. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, IEEE, v. 28, n. 10, p. 2222–2232, 2016. 10, 56
- GROOT, R. de. *Data Mining for Tweet Sentiment Classification: Twitter Sentiment Analysis*. [S.l.]: LAP LAMBERT Academic Publishing, 2012. 48
- GUIVANT, J.; NEBOT, E.; BAIKER, S. Autonomous navigation and map building using laser range sensors in outdoor. *Journal of Robotics system*, v. 17, n. 10, p. 565, 2000. 20
- GUIVANT, J.; NEBOT, E.; BAIKER, S. Localization and map building using laser range sensors in outdoor applications. *Journal of robotic systems*, Wiley Online Library, v. 17, n. 10, p. 565–583, 2000. 21
- GUIZILINI, V. C. *Localização e mapeamento simultâneos com auxílio visual omnidirecional*. Tese (Doutorado) — Universidade de São Paulo, 2008. 9, 20
- HAI, S. *The AI Index Report — Artificial Intelligence Index*. Stanford HAI: Stanford Institute for Human-Centered Artificial Intelligence, 2021. Acesso em: 17 nov. de 2021. Disponível em: <<https://aiindex.stanford.edu/report>>. 12, 27
- HÄNE, C.; HENG, L.; LEE, G. H.; FRAUNDORFER, F.; FURGALE, P.; SATTLER, T.; POLLEFEYS, M. 3d visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. *Image and Vision Computing*, Elsevier, v. 68, p. 14–27, 2017. 21
- HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. 24, 52, 95
- HEINLY, J.; DUNN, E.; FRAHM, J.-M. Comparative evaluation of binary features. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2012. p. 759–773. 59
- HOCHREITER, S. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, v. 91, n. 1, 1991. 56
- HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. 56
- HOTELLING, H. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, Warwick & York, v. 24, n. 6, p. 417, 1933. 70
- HUBARA, I.; COURBARIAUX, M.; SOUDRY, D.; EL-YANIV, R.; BENGIO, Y. Binarized neural networks. *Advances in neural information processing systems*, v. 29, p. 4107–4115, 2016. 62, 65
- HWANG, K.; SUNG, W. Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1. In: IEEE. *2014 IEEE Workshop on Signal Processing Systems (SiPS)*. [S.l.], 2014. p. 1–6. 61

- IFR. *Executive Summary World Robotics 2017 Industrial Robots*. IFR: International Federation of Robotics, 2017. Acesso em: 25 ago. de 2021. Disponível em: <[https://ifr.org/downloads/press/Executive\\_Summary\\_WR\\_2017\\_Industrial\\_Robots.pdf](https://ifr.org/downloads/press/Executive_Summary_WR_2017_Industrial_Robots.pdf)>. 26
- IFR. *Executive Summary World Robotics 2019 Industrial Robots*. IFR: International Federation of Robotics, 2019. Acesso em: 25 ago. de 2021. Disponível em: <[https://ifr.org/downloads/press2018/Executive\\_Summary\\_WR\\_2019\\_Industrial\\_Robots.pdf](https://ifr.org/downloads/press2018/Executive_Summary_WR_2019_Industrial_Robots.pdf)>. 25
- IFR. *Executive Summary World Robotics 2019 Service Robots*. IFR: International Federation of Robotics, 2019. Acesso em: 25 ago. de 2021. Disponível em: <[https://ifr.org/downloads/press2018/Executive\\_Summary\\_WR\\_Service\\_Robots\\_2019.pdf](https://ifr.org/downloads/press2018/Executive_Summary_WR_Service_Robots_2019.pdf)>. 26
- IFR. *Executive Summary World Robotics 2020 Industrial Robots*. IFR: International Federation of Robotics, 2021. Acesso em: 25 ago. de 2021. Disponível em: <[https://ifr.org/img/worldrobotics/Executive\\_Summary\\_WR\\_2020\\_Industrial\\_Robots.pdf](https://ifr.org/img/worldrobotics/Executive_Summary_WR_2020_Industrial_Robots.pdf)>. 25
- IFR. *Executive Summary World Robotics 2020 Service Robots*. IFR: International Federation of Robotics, 2021. Acesso em: 25 ago. de 2021. Disponível em: <[https://ifr.org/img/worldrobotics/Executive\\_Summary\\_WR\\_2020\\_Service\\_Robots.pdf](https://ifr.org/img/worldrobotics/Executive_Summary_WR_2020_Service_Robots.pdf)>. 26
- JUEFEI-XU, F.; BODDETI, V. N.; SAVVIDES, M. Local binary convolutional neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2017. p. 19–28. 61, 65
- KANADE, T.; COHN, J. F.; TIAN, Y. Comprehensive database for facial expression analysis. In: IEEE. *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*. [S.l.], 2000. p. 46–53. 11, 81
- KARPATY, A.; TODERICI, G.; SHETTY, S.; LEUNG, T.; SUKTHANKAR, R.; FEI-FEI, L. Large-scale video classification with convolutional neural networks. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2014. p. 1725–1732. 52
- KE, Y.; SUKTHANKAR, R. et al. Pca-sift: A more distinctive representation for local image descriptors. *CVPR (2)*, v. 4, p. 506–513, 2004. 34
- KIM, J.; HWANG, K.; SUNG, W. X1000 real-time phoneme recognition vlsi using feed-forward deep neural networks. In: IEEE. *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2014. p. 7510–7514. 61
- KIM, P.; CHEN, J.; KIM, J.; CHO, Y. K. Slam-driven intelligent autonomous mobile robot navigation for construction applications. In: SPRINGER. *Workshop of the European Group for Intelligent Computing in Engineering*. [S.l.], 2018. p. 254–269. 20
- KRIG, S. *Computer vision metrics: Survey, taxonomy, and analysis*. [S.l.]: Springer Nature, 2014. 9, 43
- KRIG, S. *Computer vision metrics*. [S.l.]: Springer, 2016. 33
- KRIZHEVSKY, A.; HINTON, G. et al. Learning multiple layers of features from tiny images. Citeseer, 2009. 11, 24, 62, 83, 104

- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105. 23, 24, 52
- KUMAR, A. C.; BHANDARKAR, S. M.; PRASAD, M. Depthnet: A recurrent neural network architecture for monocular depth prediction. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2018. p. 283–291. 23
- LABBE, M.; MICHAUD, F. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, IEEE, v. 29, n. 3, p. 734–745, 2013. 105
- LANGLEY, P.; IBA, W.; THOMPSON, K. et al. An analysis of bayesian classifiers. In: *Aaai*. [S.l.: s.n.], 1992. v. 90, p. 223–228. 48
- LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Taipei, Taiwan, v. 86, n. 11, p. 2278–2324, 1998. 10, 23, 24, 52, 62, 79
- LEE, G. H.; FRAUNDORFER, F.; POLLEFEYS, M. Structureless pose-graph loop-closure with a multi-camera system on a self-driving car. In: IEEE. *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.], 2013. p. 564–571. 21
- LEUTENEGGER, S.; CHLI, M.; SIEGWART, R. Brisk: Binary robust invariant scalable keypoints. In: IEEE. *2011 IEEE international conference on computer vision (ICCV)*. [S.l.], 2011. p. 2548–2555. 9, 34, 43, 44
- LEVINSON, J.; MONTEMERLO, M.; THRUN, S. Map-based precision vehicle localization in urban environments. In: CITESEER. *Robotics: Science and Systems*. [S.l.], 2007. v. 4, p. 1. 21
- LIU, B.; WANG, M.; FOROOSH, H.; TAPPEN, M.; PENSKY, M. Sparse convolutional neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 806–814. 62, 65
- LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 3431–3440. 52
- LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, Springer, v. 60, n. 2, p. 91–110, 2004. 34
- LOWE, D. G. et al. Object recognition from local scale-invariant features. In: *iccv*. [S.l.: s.n.], 1999. v. 99, n. 2, p. 1150–1157. 9, 34, 35, 36, 42
- LUCEY, P.; COHN, J. F.; KANADE, T.; SARAGIH, J.; AMBADAR, Z.; MATTHEWS, I. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In: IEEE. *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. [S.l.], 2010. p. 94–101. 11, 81
- LYONS, M.; AKAMATSU, S.; KAMACHI, M.; GYOBA, J. Coding facial expressions with gabor wavelets. In: IEEE. *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*. [S.l.], 1998. p. 200–205. 10, 80

- MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. [S.l.], 1967. v. 1, n. 14, p. 281–297. 47
- MAIMONE, M.; CHENG, Y.; MATTHIES, L. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, Wiley Online Library, v. 24, n. 3, p. 169–186, 2007. 21
- MAIR, E.; HAGER, G. D.; BURSCHKA, D.; SUPPA, M.; HIRZINGER, G. Adaptive and generic corner detection based on the accelerated segment test. In: SPRINGER. *European conference on Computer vision*. [S.l.], 2010. p. 183–196. 43
- MAKI, J.; GRUEL, D.; MCKINNEY, C.; RAVINE, M.; MORALES, M.; LEE, D.; WILLSON, R.; COPLEY-WOODS, D.; VALVO, M.; GOODSALL, T. et al. The mars 2020 engineering cameras and microphone on the perseverance rover: A next-generation imaging system for mars exploration. *Space Science Reviews*, Springer, v. 216, n. 8, p. 1–48, 2020. 21
- MCLACHLAN, G. J. *Discriminant analysis and statistical pattern recognition*. [S.l.]: John Wiley & Sons, 2004. v. 544. 70
- MINSKY, M.; PAPER, S. Perceptrons. 1969. *Cited on*, p. 1, 1990. 30
- MOREIRA, L. A. S. *Análise de similaridade visual em fechamento de loop através de redução de dimensionalidade de dados via mapeamentos por difusão*. Tese (Doutorado) — Programa de Pós-graduação em Engenharia de Defesa (PGED) do Instituto Militar de Engenharia (IME), apr 2017. 9, 22
- MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, IEEE, v. 31, n. 5, p. 1147–1163, 2015. 63
- NAIXIN, Q.; XIAOGANG, Y.; CHUANXIANG, L.; XIAOFENG, L.; SHENGXIU, Z.; LIJIA, C. Monocular semidirect visual odometry for large-scale outdoor localization. *IEEE Access*, IEEE, v. 7, p. 57927–57942, 2019. 21
- NETZER, Y.; WANG, T.; COATES, A.; BISSACCO, A.; WU, B.; NG, A. Y. Reading digits in natural images with unsupervised feature learning. 2011. 62
- NISTÉR, D.; NARODITSKY, O.; BERGEN, J. Visual odometry. In: IEEE. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004*. [S.l.], 2004. v. 1, p. I–I. 21
- NISTÉR, D.; NARODITSKY, O.; BERGEN, J. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, Wiley Online Library, v. 23, n. 1, p. 3–20, 2006. 21
- NVIDIA. *Jetson Nano Developer Kit*. NVIDIA Corporation, 2019. Acesso em: 13 jun. de 2021. Disponível em: <[https://developer.download.nvidia.com/embedded/L4T/r32-3-1\\_Release\\_v1.0/Jetson\\_Nano\\_Developer\\_Kit\\_User\\_Guide.pdf](https://developer.download.nvidia.com/embedded/L4T/r32-3-1_Release_v1.0/Jetson_Nano_Developer_Kit_User_Guide.pdf)>. 10, 75
- OLAH, C. *Understanding LSTM Networks*. 2015. Acesso em: 14 jul. de 2021. Disponível em: <<http://colah.github.io/posts/2015-08-Understanding-LSTMs>>. 10, 54, 55

- ONO, Y.; TRULLS, E.; FUA, P.; YI, K. M. Lf-net: Learning local features from images. *arXiv preprint arXiv:1805.09662*, 2018. 61, 65
- PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A. et al. Deep face recognition. In: *bmvc*. [S.l.: s.n.], 2015. v. 1, n. 3, p. 6. 52
- PATEL, A.; KASAT, D.; JAIN, S.; THAKARE, V. Performance analysis of various feature detector and descriptor for real-time video based face tracking. *International Journal of Computer Applications*, Foundation of Computer Science, v. 93, n. 1, 2014. 59
- PEEMEN, M.; MESMAN, B.; CORPORAAL, H. Efficiency optimization of trainable feature extractors for a consumer platform. In: SPRINGER. *International Conference on Advanced Concepts for Intelligent Vision Systems*. [S.l.], 2011. p. 293–304. 9, 53
- PENG, T.; ZHANG, D.; LIU, R.; ASARI, V. K.; LOOMIS, J. S. Evaluating the power efficiency of visual slam on embedded gpu systems. In: IEEE. *2019 IEEE National Aerospace and Electronics Conference (NAECON)*. [S.l.], 2019. p. 117–121. 63
- RAIBOLT, A.; ANGONESE, A.; ROSA, P. Comparative evaluation of feature descriptors through bag of visual features with multilayer perceptron on embedded gpu system. In: IEEE. *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*. [S.l.], 2020. p. 1–6. 108
- RASTEGARI, M.; ORDONEZ, V.; REDMON, J.; FARHADI, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 525–542. 62, 65
- ROSA, P.; SILVEIRA, O.; MELO, J. de; MOREIRA, L.; RODRIGUES, L. Development of embedded algorithm for visual simultaneous localization and mapping. In: SBC. *Anais Estendidos da XXXII Conference on Graphics, Patterns and Images*. [S.l.], 2019. p. 160–163. 63
- ROSTEN, E.; DRUMMOND, T. Machine learning for high-speed corner detection. In: SPRINGER. *European conference on computer vision*. [S.l.], 2006. p. 430–443. 42
- ROWEIS, S. T.; SAUL, L. K. Nonlinear dimensionality reduction by locally linear embedding. *science*, American Association for the Advancement of Science, v. 290, n. 5500, p. 2323–2326, 2000. 71
- RUBLEE, E.; RABAUD, V.; KONOLIGE, K.; BRADSKI, G. R. Orb: An efficient alternative to sift or surf. In: CITESEER. *ICCV*. [S.l.], 2011. v. 11, n. 1, p. 2. 34, 42
- RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation*. [S.l.], 1985. 23, 54
- RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015. 104
- SAHAMI, M. Learning limited dependence bayesian classifiers. In: *KDD*. [S.l.: s.n.], 1996. v. 96, n. 1, p. 335–338. 48

- SCOVANNER, P.; ALI, S.; SHAH, M. A 3-dimensional sift descriptor and its application to action recognition. In: ACM. *Proceedings of the 15th ACM international conference on Multimedia*. [S.l.], 2007. p. 357–360. 34
- SERMANET, P.; EIGEN, D.; ZHANG, X.; MATHIEU, M.; FERGUS, R.; LECUN, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 63
- SIEGWART, R.; NOURBAKHSI, I. R.; SCARAMUZZA, D. *Introduction to autonomous mobile robots*. [S.l.]: MIT press, 2011. 20
- SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 24
- SIVIC, J.; ZISSERMAN, A. Video google: A text retrieval approach to object matching in videos. In: IEEE. *null*. [S.l.], 2003. p. 1470. 62
- SOUDRY, D.; HUBARA, I.; MEIR, R. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In: *NIPS*. [S.l.: s.n.], 2014. v. 1, p. 2. 61
- STANKOVIĆ, R. S.; FALKOWSKI, B. J. The haar wavelet transform: its status and achievements. *Computers & Electrical Engineering*, Elsevier, v. 29, n. 1, p. 25–44, 2003. 39
- SUGIYAMA, M. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *Journal of machine learning research*, v. 8, n. 5, 2007. 70
- SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCHE, V.; RABINOVICH, A. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 1–9. 24, 52
- TARAWNEH, A. S.; CELIK, C.; HASSANAT, A. B.; CHETVERIKOV, D. Detailed investigation of deep features with sparse representation and dimensionality reduction in cbir: A comparative study. *Intelligent Data Analysis*, IOS Press, v. 24, n. 1, p. 47–68, 2020. 60
- TAREEN, S. A. K.; SALEEM, Z. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In: IEEE. *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. [S.l.], 2018. p. 1–10. 59
- TENENBAUM, J. B.; SILVA, V. D.; LANGFORD, J. C. A global geometric framework for nonlinear dimensionality reduction. *science*, American Association for the Advancement of Science, v. 290, n. 5500, p. 2319–2323, 2000. 70
- THOMAZ, C. E.; GIRALDI, G. A. A new ranking method for principal components analysis and its application to face image analysis. *Image and Vision Computing*, Elsevier, v. 28, n. 6, p. 902–913, 2010. 11, 82
- THRUN, S.; BURGARD, W.; FOX, D. Probabilistic robotics. 2005. *Massachusetts Institute of Technology, USA*, 2005. 9, 20

- TOSHEV, A.; SZEGEDY, C. Deeppose: Human pose estimation via deep neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 1653–1660. 52
- TOSIC, I.; FROSSARD, P. Dictionary learning. *IEEE Signal Processing Magazine*, v. 28, n. 2, p. 27–38, 2011. 68
- VALENTE, M.; JOLY, C.; FORTELLE, A. de L. An lstm network for real-time odometry estimation. *arXiv preprint arXiv:1902.08536*, 2019. 23
- VIJITKUNSAWAT, W.; CHANTNGARM, P. Comparison of machine learning algorithm's on self-driving car navigation using nvidia jetson nano. In: *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*. [S.l.: s.n.], 2020. p. 201–204. 63
- VINAY, A.; KUMAR, C. A.; SHENOY, G. R.; MURTHY, K. N. B.; NATARAJAN, S. Orb-pca based feature extraction technique for face recognition. *Procedia Computer Science*, v. 58, p. 614–621, 2015. ISSN 1877-0509. Second International Symposium on Computer Vision and the Internet (VisionNet'15). Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1877050915021912>>. 60, 65
- WANG, C.-C.; THORPE, C.; THRUN, S. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In: IEEE. *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*. [S.l.], 2003. v. 1, p. 842–849. 21
- WANG, N.; YEUNG, D.-Y. Learning a deep compact image representation for visual tracking. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2013. p. 809–817. 52
- WANG, S.; CLARK, R.; WEN, H.; TRIGONI, N. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In: IEEE. *2017 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2017. p. 2043–2050. 23
- WEI, X.; YU, X.; LIU, B.; ZHI, L. Convolutional neural networks and local binary patterns for hyperspectral image classification. *European Journal of Remote Sensing*, Taylor & Francis, v. 52, n. 1, p. 448–462, 2019. 61, 65
- WEICKERT, J.; ROMENY, B. T. H.; VIERGEVER, M. A. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE transactions on image processing*, IEEE, v. 7, n. 3, p. 398–410, 1998. 40
- WILLIAMS, S. B.; NEWMAN, P.; DISSANAYAKE, G.; DURRANT-WHYTE, H. Autonomous underwater simultaneous localisation and map building. In: IEEE. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. [S.l.], 2000. v. 2, p. 1793–1798. 20
- XIE, J.; KIEFEL, M.; SUN, M.-T.; GEIGER, A. Semantic instance annotation of street scenes by 3d to 2d label transfer. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016. 105

- YANG, X.; CHENG, K.-T. Ldb: An ultra-fast feature for scalable augmented reality on mobile devices. In: IEEE. *2012 IEEE international symposium on mixed and augmented reality (ISMAR)*. [S.l.], 2012. p. 49–57. 46
- ZHANG, M.-L.; ZHOU, Z.-H. A k-nearest neighbor based algorithm for multi-label classification. In: IEEE. *Granular Computing, 2005 IEEE International Conference on*. [S.l.], 2005. v. 2, p. 718–721. 48
- ZHANG, M.-L.; ZHOU, Z.-H. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, Elsevier, v. 40, n. 7, p. 2038–2048, 2007. 48
- ZHANG, X.; SU, Y.; ZHU, X. Loop closure detection for visual slam systems using convolutional neural network. In: IEEE. *2017 23rd International Conference on Automation and Computing (ICAC)*. [S.l.], 2017. p. 1–6. 62, 63, 65
- ZHANG, Z.; LYONS, M.; SCHUSTER, M.; AKAMATSU, S. Comparison between geometry-based and gabor-wavelets-based facial expression recognition using multi-layer perceptron. In: IEEE. *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*. [S.l.], 1998. p. 454–459. 23
- ZHUO, L.; CHENG, B.; ZHANG, J. A comparative study of dimensionality reduction methods for large-scale image retrieval. *Neurocomputing*, v. 141, p. 202–210, 2014. ISSN 0925-2312. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0925231214004238>>. 60

## APÊNDICE A – JETSON NANO: ETAPAS DE INSTALAÇÃO

Neste Apêndice, apresentamos as etapas de instalação necessárias para utilização do microcomputador Jetson Nano, a saber:

- **Gravação da imagem no cartão MicroSD:** Nesta Seção são apresentadas as etapas realizadas para executar a gravação da imagem do Kit de Desenvolvimento NVIDIA Jetson Nano no cartão MicroSD.

### A.1 Gravação da imagem no cartão MicroSD

Como dispositivo de inicialização e armazenamento principal, o microcomputador Jetson Nano faz uso de um cartão MicroSD. Segundo sua documentação, a recomendação é que seja utilizado um cartão UHS-1 de no mínimo 16GB. Para o desenvolvimento desta proposta empregamos um MicroSDXC de 64GB Classe 10 da HP Inc., garantindo, desta forma, a utilização de um cartão com armazenamento suficiente e rápido para o desenvolvimento de experimentos e simulações futuras.

Em sua documentação, é requisito para gravar o sistema operacional e o *software* no cartão MicroSD: um computador com conexão à Internet e com um slot ou adaptador de cartão SD embutido. Para esta tarefa, utilizamos um notebook Ultrafino Samsung Ativ Book 4 NP470R4E-KD1 com Intel® Core™ i5- 3230M, 4GB, 500GB e *Windows* 10.

Realizamos o *download* da imagem do Kit de Desenvolvimento NVIDIA Jetson Nano que pesa cerca de 5,64GB em um arquivo de imagem compactado, através do site oficial da NVIDIA<sup>1</sup>. Para executar a gravação no MicroSD, foram seguidas as instruções conforme o tipo de plataforma utilizada (em nosso caso, a plataforma *Windows*, como já mencionado). Portanto, abaixo é descrita as etapas a serem executadas:

#### A.1.1 Formatação do cartão MicroSD

Primeiramente, é recomendado fazer o *download*, instalação e inicialização do *software SD Memory Card Formatter* para *Windows*. A interface do *software SD Memory Card Formatter* pode ser observado na Figura 54.

---

<sup>1</sup> <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>

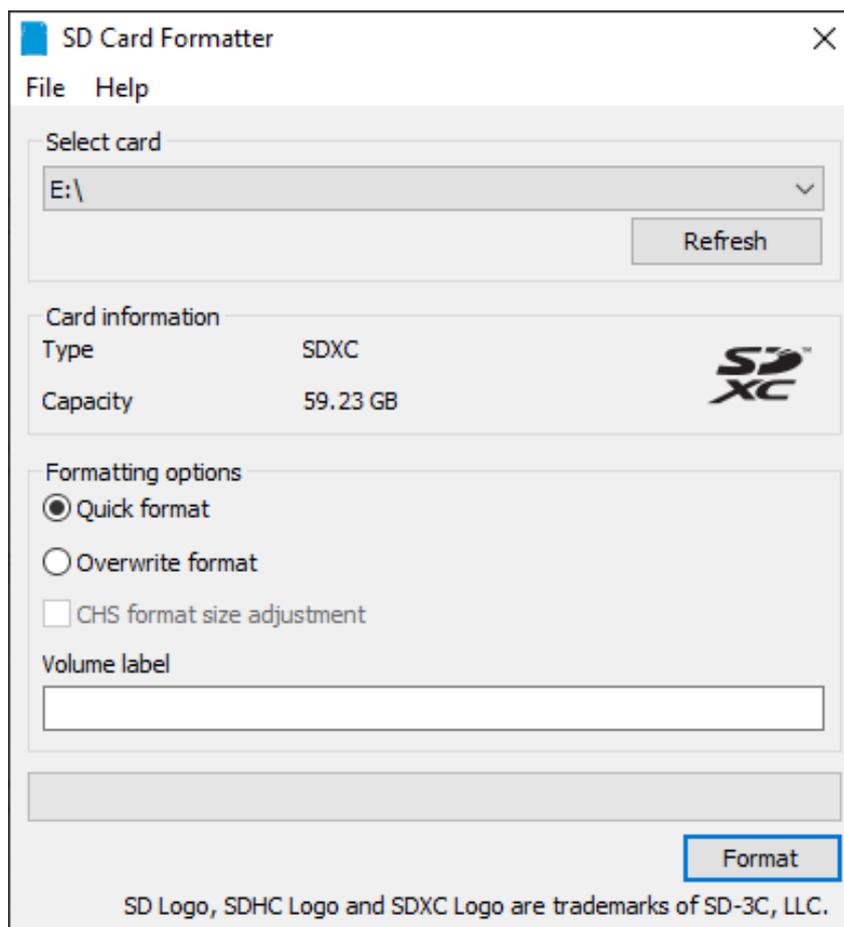


Figura 54 – Interface do *software SD Memory Card Formatter*. Fonte: Elaborado pela autora.

Em seguida, é necessário:

1. Selecionar a unidade do cartão em “*Select card*” (em português, Selecionar cartão).
2. Selecionar a opção “*Quick format*” (em português, Formatação rápida).
3. Deixar a opção “*Volume label*” (em português, Rótulo do volume) em branco.
4. Clicar em “*Format*” (em português, Formatar) para iniciar a formatação e ao final, na caixa de diálogo de aviso clicar na opção “*Yes*” (em português, Sim).

A próxima etapa consiste na gravação propriamente dita, da imagem do Kit de Desenvolvimento NVIDIA Jetson Nano no cartão MicroSD.

### A.1.2 Gravação da imagem no cartão MicroSD

É recomendado realizar o *download*, instalação e inicialização do *software* Etcher para *Windows*. A interface do *software* Etcher pode ser observada na Figura 55.

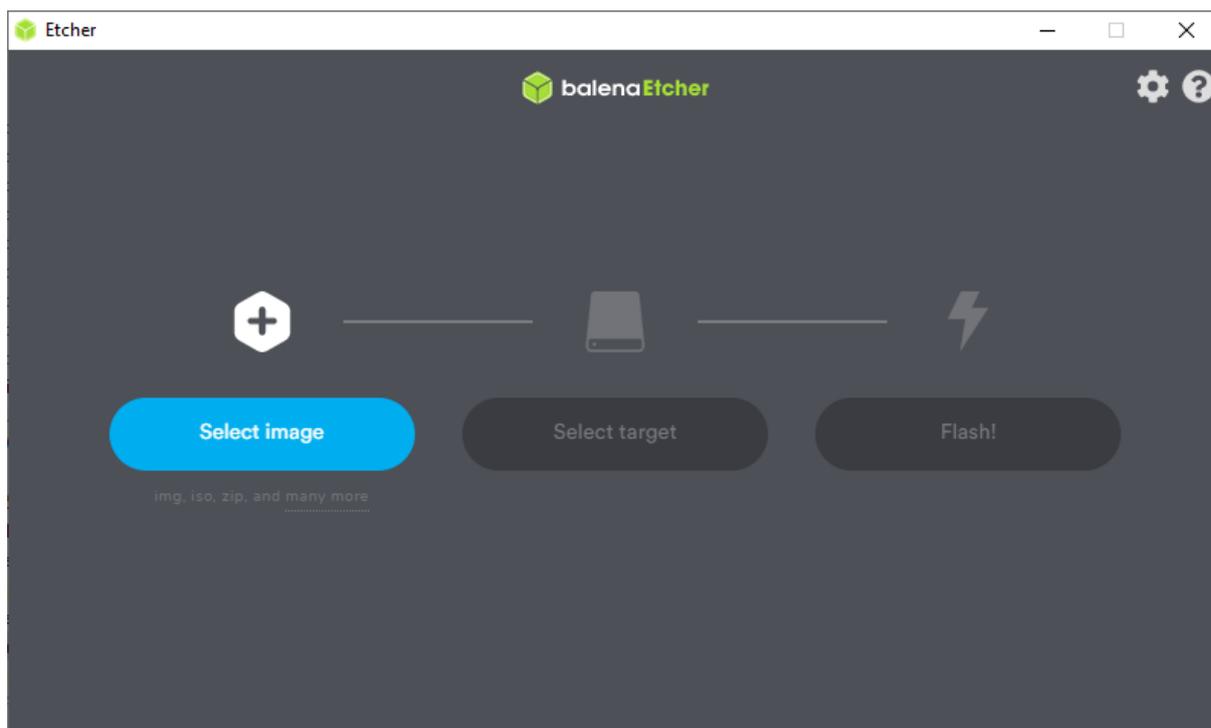


Figura 55 – Interface do *software* Etcher. Fonte: Elaborado pela autora.

Em seguida, é necessário:

1. Selecionar a opção “*Select image*” (em português, Selecionar imagem) e selecionar o arquivo de imagem compactado (em nosso caso: **nv-jetson-nano-sd-card-image-r32.3.1.zip**).
2. Caso o cartão MicroSD ainda não esteja inserido, insira-o no slot ou adaptador de cartão SD embutido.
3. Selecionar a opção “*Select drive*” (em português, Selecionar unidade) e escolha o dispositivo que será atualizado com a imagem.
4. Selecionar a opção “*Flash!*”.

Na Figura 56 é possível observar os parâmetros selecionados nesta etapa.

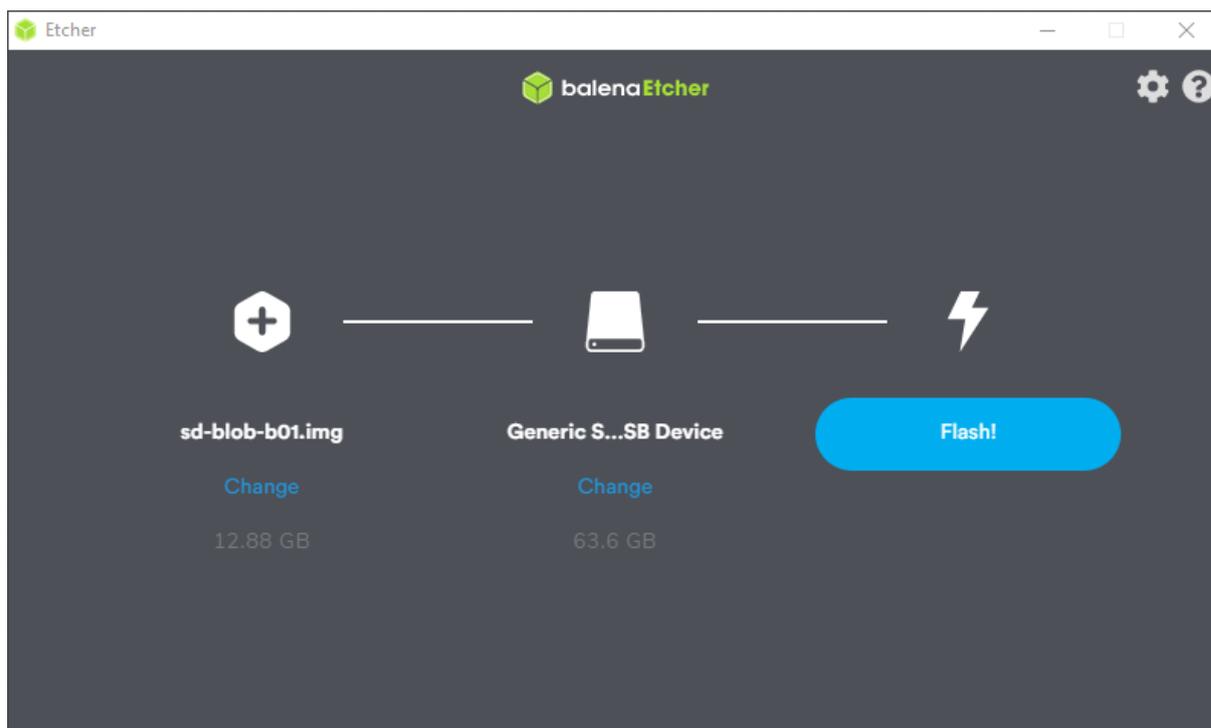


Figura 56 – Interface Etcher — parâmetros selecionados. Fonte: Elaborado pela autora.

O processo de gravação e validação da imagem no cartão MicroSD pode demorar entre 10 a 15 minutos. Ao término desta etapa, é possível que o *Windows* informe que é necessário formatar a unidade de disco para poder usá-lo, isto, porque, o *Windows* não é capaz de ler o cartão MicroSD com a imagem do Kit de Desenvolvimento NVIDIA Jetson Nano. Entretanto, é necessário selecionar a opção “*Cancel*” (em português, Cancelar) e remover o cartão MicroSD. A mensagem de alerta do *Windows* pode ser observada na Figura 57.

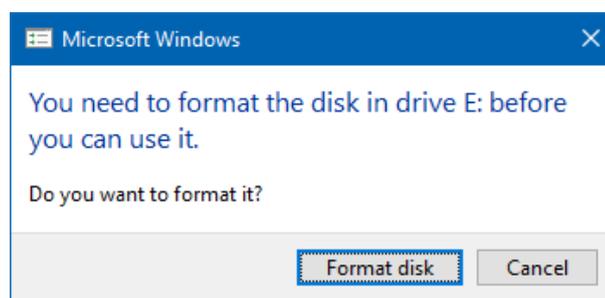


Figura 57 – Mensagem de alerta do *Windows*. Fonte: Elaborado pela autora.

Com a realização das etapas apresentadas, o cartão MicroSD está pronto para uso.

## APÊNDICE B – JETSON NANO: ETAPAS DE CONFIGURAÇÃO

Neste Apêndice, apresentamos as etapas de configuração do microcomputador Jetson Nano, a saber:

- **Expansão da memória *Swap*:** São apresentadas nesta Seção as etapas realizadas para a criação de 4GB de memória *Swap*;
- **Instalação da biblioteca *OpenCV*:** Nesta Seção são apresentadas as etapas realizadas para a instalação e configuração da biblioteca *OpenCV* para a realização dos experimentos e simulações;
- **Instalação de outras dependências:** São apresentadas nesta Seção as etapas realizadas para a instalação e configuração de outras dependências para a realização dos experimentos e simulações.

### B.1 Expansão da memória *Swap*

Com o cartão MicroSD já instalado no microcomputador Jetson Nano, para a instalação, compilação e utilização de bibliotecas como *TensorFlow*, *Keras* e *OpenCV* é necessária uma quantidade excessiva de memória, o microcomputador Jetson Nano possui apenas 4GB de RAM disponível, o que não será o bastante para a compilação das bibliotecas citadas acima, portanto, faz-se necessário uma expansão de memória *swap* no microcomputador Jetson Nano. Os seguintes comandos podem ser utilizados no Terminal do microcomputador para a criação de 4GB de memória *swap*:

```
sudo fallocate -l 4G /var/swapfile
sudo chmod 600 /var/swapfile
sudo mkswap /var/swapfile
sudo swapon /var/swapfile
sudo bash -c 'echo "/var/swapfile swap swap defaults 0 0" >> /etc/fstab'
```

Para verificar se o processo ocorreu adequadamente, é possível utilizar o seguinte comando no Terminal:

```
free -h
```

Caso seja apresentado o tamanho 4,0G, ou o tamanho escolhido para a expansão de memória *swap*, significa que o processo ocorreu de forma esperada, ou seja, com sucesso.

## B.2 Instalação da biblioteca *OpenCV*

Optamos em instalar a biblioteca *OpenCV* 4.1.0 de modo a realizar alguns experimentos e simulações. Para isto, realizamos os seguintes procedimentos:

### B.2.1 Atualização de pacotes

```
$ sudo apt update
$ sudo apt install -y build-essential cmake git libgtk2.0-dev
  pkg-config libswscale-dev libtbb2 libtbb-dev
$ sudo apt install -y python-dev python3-dev python-numpy
  python3-numpy python3-scipy
$ sudo apt install -y curl
```

### B.2.2 Pacotes necessários para formatos de imagem e vídeo

```
$ sudo apt install -y libjpeg-dev libpng-dev libtiff-dev
  libjasper-dev
$ sudo apt install -y libavcodec-dev libavformat-dev
$ sudo apt install -y libgstreamer1.0-dev libgstreamer-plugins-
  base1.0-dev
$ sudo apt install -y libv4l-dev v4l-utils qv4l2 v4l2ucp
  libdc1394-22-dev
```

### B.2.3 Download dos módulos do *OpenCV* e *Contribs*

```
$ curl -L https://github.com/opencv/opencv/archive/4.1.0.zip -o
  opencv-4.1.0.zip
$ curl -L https://github.com/opencv/opencv_contrib/archive
  /4.1.0.zip -o opencv_contrib-4.1.0.zip
```

### B.2.4 Descompactação dos pacotes

```
$ unzip opencv-4.1.0.zip
$ unzip opencv_contrib-4.1.0.zip
```

### B.2.5 Renomeando os diretórios

```
$ mv opencv-4.1.0 OpenCV
$ mv opencv_contrib-4.1.0 OpenCV_Contrib
```

### B.2.6 Criação de um novo diretório

```
$ cd OpenCV
$ mkdir release
$ cd release/
```

## B.2.7 Criando o *OpenCV* através do CMake

```
$ cmake -D WITH_CUDA=ON \
-D WITH_CUBLAS=ON \
-D WITH_LIBV4L=ON \
-D WITH_GSTREAMER=ON \
-D WITH_GTK=ON \
-D CUDA_ARCH_PTX="" \
-D CUDA_ARCH_BIN="5.3,6.2,7.2" \
-D OPENCV_ENABLE_NONFREE=ON \
-D OPENCV_EXTRA_MODULES_PATH=../../OpenCV_Contrib/
modules \
-D BUILD_opencv_java=OFF \
-D BUILD_opencv_python2=ON \
-D BUILD_opencv_python3=ON \
-D BUILD_EXAMPLES=ON \
-D BUILD_TESTS=OFF \
-D BUILD_PERF_TESTS=OFF \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D INSTALL_C_EXAMPLES=OFF \
-D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local ..
```

## B.2.8 Compilação do *OpenCV* com os módulos Contribs

```
$ make -j4
$ sudo make install
```

Após a realização destes procedimentos a instalação do *OpenCV* 4.1.0 está finalizada.

## B.3 Instalação de outras dependências

Como muitas das dependências precisam ser compiladas a partir do código-fonte, a instalação pode demorar.

### B.3.1 PIP3

A ferramenta para gerenciamento de pacotes Python — PIP3 não vem pré-instalado no microcomputador Jetson Nano, sendo necessário fazer a sua instalação. Para tal:

```
$ sudo apt-get install python3-pip
$ sudo pip3 install -U pip testresources setuptools==49.6.0
```

### B.3.2 Cython

```
$ sudo pip3 install Cython
```

### B.3.3 Numpy

```
$ sudo pip3 install numpy
```

### B.3.4 Matplotlib

```
$ sudo apt-get install python3-matplotlib
```

### B.3.5 Pandas

```
$ sudo pip3 uninstall enum34
$ sudo pip3 install pandas
```

### B.3.6 Scipy

```
$ wget https://github.com/scipy/scipy/releases/download
  {}/v1.3.3/scipy-1.3.3.tar.gz
$ tar -xzvf scipy-1.3.3.tar.gz scipy-1.3.3
$ cd scipy-1.3.3/
$ python3 setup.py install --user
```

### B.3.7 Scikit-Learn

```
$ sudo apt-get install -y build-essential libatlas-base-dev
  gfortran
$ sudo apt-get install python3-sklearn
```

### B.3.8 Keras

```
$ sudo apt-get install libhdf5-serial-dev hdf5-tools
$ sudo apt-get install zlib1g-dev zip libjpeg8-dev libhdf5-dev
$ sudo pip3 install -U grpcio absl-py py-cpuinfo psutil
  portpicker six mock requests gast h5py astor termcolor
$ sudo pip3 install -U keras
```

### B.3.9 TensorFlow

```
$ sudo pip3 install --extra-index-url https://developer.download
.nvidia.com/compute/redis/jp/v42 tensorflow-gpu
```

## APÊNDICE C – REPOSITÓRIOS

Neste Apêndice, apresentamos os links de acesso aos repositórios *git* relacionados aos códigos-fontes desenvolvidos nesta Dissertação, a saber:

- **Repositório com a implementação das técnicas de Detecção e Descrição de Características:**

<<https://github.com/whoisraibolt/Feature-Detection-and-Description>>

- **Repositório com a implementação das técnicas de Detecção e Correspondência de Características:**

<<https://github.com/whoisraibolt/Feature-Detection-and-Matching>>

- **Repositório com a implementação da abordagem Saco de Características Visuais:**

<<https://github.com/whoisraibolt/BoVF-with-MLP-classifier>>

- **Repositório com a implementação da reformulação de Descritores através de filtros convolucionais:**

<<https://github.com/whoisraibolt/Reform-Conv-Filters-through-Descriptors>>

- **Repositório com a implementação da DescNet:**

<<https://github.com/whoisraibolt/DescNet>>

## ANEXO A – CONTRIBUIÇÕES CIENTÍFICAS

Neste Anexo, apresentamos as publicações realizadas durante a evolução deste trabalho de Dissertação, a saber:

- RAIBOLT; ANGONESE; ROSA, **Comparative Evaluation of Feature Descriptors Through Bag of Visual Features with Multilayer Perceptron on Embedded GPU System** 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), 2020, pp. 1-6, doi: 10.1109/LARS/SBR/WRE51543.2020.9306931. Artigo escolhido entre os 3 melhores artigos publicados no Simpósio Latino-Americano de Robótica/Simpósio Brasileiro de Robótica (LARS/SBR 2020) em novembro de 2020;
- RAIBOLT; ANGONESE; ALVES; ROSA, **Towards Loop Closure Detection for SLAM Applications using Bag of Visual Features: Experiments and Simulation** Computational Neuroscience: Third Latin American Workshop, LAWCN 2021. (aceito para publicação nos Anais da Conferência);
- RAIBOLT; ANGONESE; ALVES; ROSA, **Descriptor Convolutional Neural Network (DescNet)**. (a ser submetido para Periódico).