UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

LEANDRO YUJI KANNO

Desenvolvimento de algoritmo de visão computacional com redes neurais para navegação de robô aquático

São Carlos
2024

LEANDRO YUJI KANNO

Desenvolvimento de algoritmo de visão computacional com redes neurais para navegação de robô aquático

Monografia apresentada ao Curso de Engenharia Mecatrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Mecatrônico.

Orientadora: Profª. Dra. Maíra Martins da Silva

São Carlos

2024

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

K16d

Kanno, Leandro Yuji
Desenvolvimento de algoritmo de visão computacional com redes neurais para navegação de robô aquático / Leandro Yuji Kanno; orientadora Maíra Martins da Silva. São Carlos, 2023.

Monografia (Graduação em Engenharia Mecatrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2023.

1. Visão computacional. 2. TensorFlow. 3. Detecção de Objetos. 4. Rastreamento de Objetos. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE AVALIAÇÃO

Candidato: Leandro Yuji Kanno

Título: Desenvolvimento de algoritmo de visão computacional com redes neurais para navegação de robô aquático

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos da Universidade de São Paulo Curso de Engenharia Mecatrônica.

BANCA EXAMINADORA

Professora Maíra Martins da Silva (Orientadora)

Nota atribuída: 10,0 (dez) Maíra M. da Silva

(assinatura)

Professor Adriano Almeida Gonçalves Siqueira

Nota atribuída: 10,0 (dez) Maíra M. da Silva

(assinatura)

Professor Alberto Cliquet Junior

Nota atribuída: 10,0 (dez) Maíra M. da Silva

(assinatura)

Média: 10,0 (dez)

Resultado: APROVADO

Data: 21/12/2023.

Este trabalho tem condições de ser hospedado no Portal Digital da Biblioteca da EESC

SIM ☑ NÃO ☐ Visto do orientador Maíra M. da Silva

Este trabalho é dedicado ao laboratório de Dinâmica,
como uma contribuição para o desenvolvimento de novas ideias
e oportunidades de aprendizados.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, que sempre estiveram presentes e incentivaram minhas escolhas e fizeram seu máximo para ajudar a segui-las. Ao meu pai, agradeço por sempre ter sido uma pessoa exemplar que zelou pela minha segurança, futuro e sonhos. À minha mãe, agradeço por ter sido motivação e ter confiado em mim até seu último dia. Suas últimas palavras dirigidas a mim ainda ecoam em meu coração, e não seria possível seguir em frente sem elas.

Minha eterna gratidão à Luna, que em sua inocência e pureza sempre deu à família alegria em bons momentos e conforto nos momentos de maiores pesares.

Agradeço também às amizades que fiz dentro e fora da universidade, que não só me apoiaram academicamente, como também em meu luto. Breno, Felippe, João Guilherme, Kamila, Samanta e Vinicius.

Sou grato também a minha grande amiga e companheira Bruna, que colaborou de diversas formas ao longo do desenvolvimento do trabalho e em múltiplos campos da vida, manteve tanto meus pés no chão e meu foco para frente.

Por fim, agradeço à professora doutora Maíra Martins, não apenas por ter feito muito pela comunidade de discentes da USP, como também por ter me dado a oportunidade de aprender mais sobre assuntos que me interesso muito e, ainda mais, permitiu que a liberdade criativa orientasse o desenvolvimento deste projeto.

“Divida cada dificuldade em quantas partes for viável
e necessário para a resolver.”
René Descartes

RESUMO

KANNO, L. 124p. **Desenvolvimento de algoritmo de visão computacional com redes neurais para navegação de robô aquático**. 2023. Monografia (Trabalho de Conclusão de Curso) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2023.

O crescente progresso de tecnologias de processamento em escalas cada vez menores possibilitou o uso de tarefas que antes eram computacionalmente dispendiosas em sistemas embarcados de dimensões reduzidas. Em diversas situações a portabilidade e baixo peso são parâmetros desejáveis, e, com o objetivo de complementar o estudo de robôs macios aquáticos bioinspirados, esta tese busca desenvolver um código que colete a localização da posição relativa dos objetos no ambiente em relação à câmera acoplada ao sistema robótico. Para tal, com o requisito da versatilidade em termos de modos de atuação e tipos de objetos que o algoritmo é capaz de detectar, serão utilizados modelos de redes neurais para realizar a inferência das imagens coletadas. Dentre as vantagens do uso de aprendizado profundo estão a possibilidade de alternar entre modelos pré existentes e variar entre a velocidade, ou seja, maior ou menor exigência de recursos computacionais, e a acurácia, o que permite a adequação a diferentes hardwares. A linguagem de programação Python será utilizada junto à biblioteca TensorFlow, que fornece APIs de alto nível para aprendizado de máquina.

Palavras-chave: Visão computacional. TensorFlow. Detecção de Objetos. Rastreamento de Objetos.

ABSTRACT

KANNO, L. P. **Development of a computer vision algorithm with neural networks for aquatic robot navigation.**. 2023. 124p. Monograph (Conclusion Course Paper) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2023.

The growing progress of processing technologies on increasingly smaller scales has enabled the use of tasks that were previously computationally expensive in embedded systems of reduced dimensions. In various situations, portability and low weight are desirable parameters. With the aim of complementing the study of bioinspired aquatic soft robots, this thesis seeks to develop code that collects the location of the relative position of objects in the environment concerning the camera attached to the robotic system. To achieve this, with the requirement of versatility in terms of operating modes and types of objects that the algorithm can detect, neural network models will be used to infer the collected images. Among the advantages of using deep learning are the possibility of switching between pre-existing models and varying between speed, i.e., higher or lower computational resource requirements, and accuracy, allowing adaptation to different hardware. The Python programming language will be used alongside the TensorFlow library, which provides high-level APIs for machine learning.

Keywords: Computer Vision. TensorFlow. Object Detection. Object Tracking.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface (Interface de Programação de Aplicação)
ROS	Robot Operating System
MS COCO	Microsoft Common Objects in Context
mAP	Mean Average Precision

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto e Motivação

A coleta de dados em ambientes subaquáticos se beneficia substancialmente de dispositivos com boa manobrabilidade e rendimento energético. Nesse contexto, exploram-se novas alternativas de mecanismos de locomoção, com a robótica macia bioinspirada visando a eficiência de uma estrutura capaz de realizar movimentos ondulatórios e oscilatórios para deslocar-se. Com o intuito de avaliar a eficácia dessa abordagem, é fundamental analisar a dinâmica do robô ao executar tarefas em que a performance de trajetórias variáveis se torna um requisito para o bom desempenho do sistema. Isso se torna possível por meio da execução da função de perseguição de um objeto alvo.

A viabilidade dessa abordagem é facilitada pelo campo de estudo da visão computacional, que oferece diversas possibilidades e soluções para a detecção de objetos. Utilizando técnicas de processamento e operações sobre as imagens, bibliotecas em Python são empregadas para extrair coordenadas de objetos. O laboratório conduziu estudos iniciais utilizando técnicas de visão computacional com detecção de objetos baseada na análise de formas e padrões geométricos. Agora, esta tese propõe a utilização de redes neurais para detectar uma ampla gama de objetos.

Entre as vantagens das redes neurais destaca-se a versatilidade, pois conseguem lidar com diferentes formas e objetos que possuem variações expressivas de características, dependendo do ângulo e das condições de iluminação. Esses são fatores que, em outras técnicas, dificultariam significativamente a tarefa da inferência de objetos na imagem. No entanto, por meio do aprendizado profundo, a viabilidade da operação não é comprometida.

1.2 Objetivos

Este trabalho consiste em desenvolver um código que realiza a inferência de objetos nas imagens coletadas e, não só isso, o rastreamento dos objetos presentes. Para atender a diferentes interesses de operações e testes, alguns requisitos de versatilidade são desejados, de forma que seja possível priorizar diferentes classes de objetos e, no caso de existirem múltiplos objetos da mesma categoria na imagem coletada pela câmera, estabelecer critérios para definir um deles em específico.

1.3 Organização do Trabalho

Este trabalho foi dividido em 5 capítulos, seguidos da seção de Referências. O presente capítulo apresenta a contextualização e objetivos. O segundo capítulo abordará a Revisão da Literatura que conterá conceitos imprescindíveis para a compreensão do desenvolvimento do trabalho. O terceiro capítulo lista os materiais utilizados, suas especificações, limitações para

execução dos objetivos propostos e processo de desenvolvimento a partir deles. A quarta seção consta os resultados obtidos através dos procedimentos descritos desenvolvimento. A quinta seção contém as conclusões, considerações e perspectivas para o desenvolvimento de futuros trabalhos.

2 REVISÃO BIBLIOGRÁFICA

2.1 Visão Geral

Esta seção tem o objetivo de fornecer os alicerces teóricos para a compreensão do trabalho desenvolvido, através do entendimento dos principais fundamentos que compõem os algoritmos de visão computacional.

A apresentação dos conceitos primordiais será de suma importância para que a abordagem das principais soluções base existentes possa ser comparada através do mesmo escopo de premissas, desempenho e propósitos, como nos casos em que há diferentes alternativas para realizar cálculos e tarefas dentro do algoritmo.

Grande parte das escolhas leva em consideração os recursos computacionais disponíveis e a viabilização da coleta de dados e elaboração de uma resposta em tempo real, visto que dentro da visão computacional existem operações em que não é necessária uma resposta imediata, como no caso de análises de imagens e vídeos coletados previamente, o que não é o caso de um sistema que visa controlar um robô em movimento.

O funcionamento da aplicação em questão, a locomoção até um objeto, pode ser dividido nas etapas da detecção do objeto, o rastreamento dos elementos presentes e a escolha de um deles como alvo do robô. As etapas da detecção e rastreamento são as mais complexas e possuem soluções já trabalhadas que devem ser analisadas e adaptadas, enquanto o estabelecimento de critérios para definir um alvo é uma questão específica e será abordada na seção sobre o desenvolvimento do código.

2.2 Visão Computacional

A visão computacional é um campo que visa fornecer aos computadores a capacidade de extrair informações de imagens de forma autônoma e possui diversas aplicações cotidianas como reconhecimento de objetos, detecção de padrões, reconhecimento de impressões digitais, reconhecimento facial, entre outros.

O procedimento para realizar essas tarefas pode variar bastante dependendo de qual o objetivo a ser cumprido e dispõe de uma grande variedade de técnicas e métodos que permitem a análise e interpretação de vídeos por sistemas computacionais.

Através de princípios fundamentais da geometria, estatística, álgebra linear e processamento de sinais, os pixels que compõem os quadros analisados têm seus padrões e características relevantes extraídas. Tipicamente, os sistemas de visão computacional passam pelas etapas descritas a seguir. (GONZALEZ, 2008)

2.2.1 **Pré-processamento**

Antes da análise dos pixels presentes, o pré-processamento da imagem realiza tarefas como a redução de ruídos, normalização da iluminação, ajuste de contraste e outros procedimentos que visam a melhora da qualidade dos dados e a facilitação da extração de características.

2.2.2 **Segmentação de imagens**

A segmentação divide uma imagem em regiões que contém padrões com indicativos de que possuem elementos de interesse. Essa segmentação pode ocorrer através de procedimentos que levam em conta as coordenadas da região, o contorno de objetos detectados por varreduras que analisam o gradiente dos valores ao longo dos pixels, segmentação semântica, entre outros métodos. Uma segmentação bem sucedida identifica objetos individuais.

2.2.3 **Extração de Características**

Na etapa da extração de características são identificados padrões distintos nas regiões de interesse previamente segmentadas, com técnicas como a detecção de bordas, linhas, regiões e pontos de interesse e extração de descritores (GONZALEZ, 2008). Podemos citar como exemplos de descritores analisados:

- **Histogramas de cores**

  Análise da distribuição de cores na porção da imagem através do padrão do histograma.

- **Textura**

  Descritores que se baseiam em padrões de textura, como, por exemplo, filtros Gabor ou características Haralick.

- **Gradientes e bordas**

  Representam a intensidade das mudanças de valores armazenados nos pixels, como o Histograma de Gradientes Orientados (HOG).

- **Momentos de imagem**

  Descreve a forma e distribuição dos pixels e morfologias encontradas na imagem.

- **Descritores de forma**

  Parâmetros e padrões geométricos que caracterizam a forma do elemento de interesse.

- **Descritores de borda**

  Analisa as bordas detectadas na imagem e seus padrões quando relacionadas.

- **Descritores locais**

  Descreve características localizadas como, por exemplo, SIFT (Scale-Invariant Feature Transform) e SURF (Speeded Up Robust Features).

- **Descritores de frequência**

  Relaciona-se às características da frequência da imagem. Usualmente a partir da conversão da imagem para sua correspondente no domínio da frequência através da Transformada de Fourier.

- **Momentos invariantes**

  Descritores que, mesmo com transformações geométricas como rotação, translação e alterações na escala, permanecem inalterados.

- **Histograma de gradientes de cores**

  Descreve a distribuição dos gradientes de cores de acordo com o sistema de codificação da imagem e seus canais.

- **Redes neurais pré-treinadas**

  Descritores pautados em redes neurais treinadas para tarefas específicas de visão computacional.

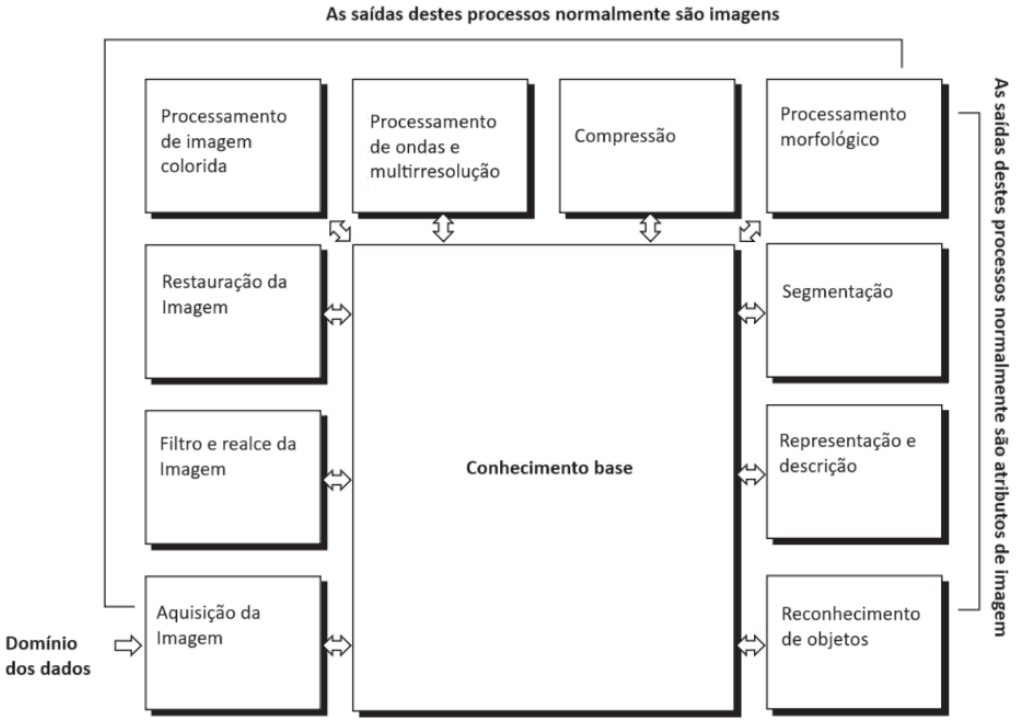
2.2.4 **Reconhecimento**

Através das características extraídas é feita a associação de objetos a classes ou identidades. Os elementos detectados passam então por uma análise a fim de que seja inferido se aquele conjunto de pixels possui evidências suficientes para que seja classificado como pertencente a uma das categorias de interesse do algoritmo.

2.3 Redes Neurais Convolucionais Profundas

O aprendizado profundo é um subconjunto dentro do aprendizado de máquina e se baseia em redes neurais profundas. Ao contrário do aprendizado de máquina tradicional, em que os dados se caracterizam por serem estruturados e bem definidos, o aprendizado profundo é adequado para tarefas complexas com dados não estruturados, como em aplicações que envolvem o reconhecimento de padrões em sons e imagens.

Figura 1 – Etapas do processamento de imagem e visão computacional

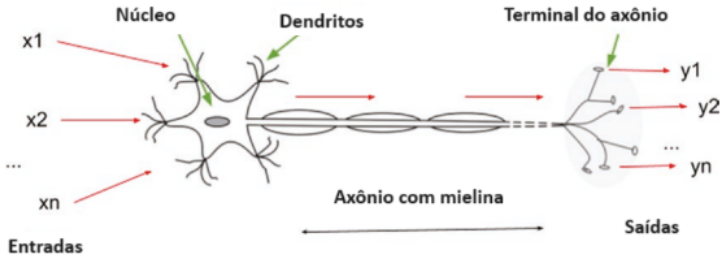


Modificado de Rafael C. Gonzalez, 2008

2.3.1 **Rede Neural Artificial**

Uma rede neural artificial (artificial neural network, ANN) é projetada para fazer a mímese de como funciona um cérebro humano: aprende a reconhecer e classificar de acordo com a análise de padrões. Cientistas da computação se inspiraram no sistema de neurônios, humanos que se conectam e cruzam informações, para projetar as redes neurais artificiais.

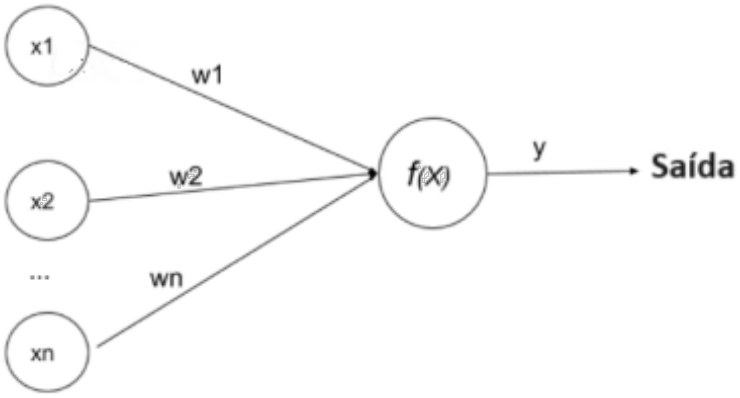
Figura 2 – Modelo lógico de um neurônio humano



Modificado de (ANSARI, 2020).

Tanto nos neurônios biológicos quanto nos das redes neurais artificiais, os sinais de entrada x1, x2, ..xn são associados a pesos w1, w2, ..wn, e esses sinais são então processados através de funções para gerar saídas. A unidade de processamento que combina esses sinais de entrada é chamada de **neurônio** e sua função matemática é chamada de **função de ativação**.

Figura 3 – Neurônio artificial



Modificado de (ANSARI, 2020).

2.3.2 **Perceptron**

Um único neurônio de uma rede neural é chamado de perceptron, ele implementa uma função matemática que opera nos sinais de entrada e gera saídas. Isoladamente, forma a rede neural mais simples, como no caso da Figura 3. As entradas para o neurônio são coletadas do ambiente através de dispositivos como câmeras ou outros aparatos de sensoriamento, mas também podem vir das saídas de outros neurônios.

2.3.3 **Aprendizado de um Perceptron**

O objetivo de aprendizado de um perceptron é a determinação dos pesos ideais para cada sinal de entrada. O algoritmo de aprendizado atribui arbitrariamente pesos a cada sinal de entrada, que são multiplicados pelos seus pesos correspondentes. O resultado, peso vezes valor do sinal, é somado para calcular uma saída. A computação é representada pelas seguintes equações:

$$f(x) = w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

Em alguns casos é interessante fornecer pesos iniciais para direcionar o resultado através de um viés x0

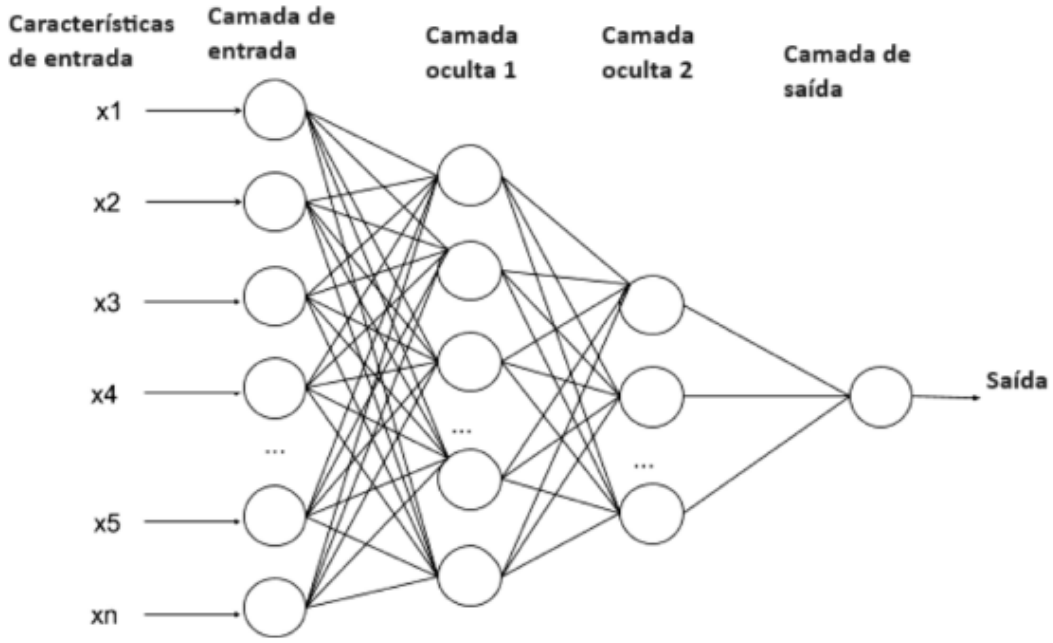
$$f(x) = x_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

O neurônio recebe um grande número de entradas, e então uma função de otimização adequa os pesos usando funções matemáticas chamadas otimizadores, e a computação é repetida com os novos pesos. Essa iteração continuamente otimiza os valores até que o resultado seja satisfatório para o conjunto de entradas fornecido. Esse processo constitui o aprendizado do neurônio.

2.3.4 **Perceptron de Múltiplas Camadas**

Uma rede neural artificial típica contém vários perceptrons. As entradas são processadas por um grupo de neurônios e cada um deles processa as entradas de forma independente. As saídas deste grupo de neurônios são alimentadas para outro neurônio ou camada neurônios. Desta forma, a saída de uma camada atua como entrada para a próxima camada, e é possível adicionar indefinidas camadas para treinar a rede neural. Essa organização de neurônios em camadas é comumente conhecida como Perceptron de Múltiplas Camadas (MLP), como mostra a Figura 4.

Figura 4 – Perceptron multicamadas



Modificado de (ANSARI, 2020)

MLPs são úteis pois, por exemplo, ao levarmos em consideração um único neurônio com uma entrada, ele terá uma função de ativação que se assemelha a:

$$f(x) = w_1 x_1 + x_0$$

O que representa uma função linear. Entretanto, a maioria dos problemas encontrados no mundo real não exibem comportamentos lineares, e Perceptrons de múltiplas camadas modelam a não linearidade e podem abordar situações do cotidiano de forma mais precisa do que modelos baseados em neurônios individuais e algoritmos de aprendizado de máquina como regressão linear e regressão logística.

Aprendizado profundo é outra nomenclatura dada a uma rede neural artificial de várias camadas ou perceptron de várias camadas, e diferentes tipos de sistemas são empregados dependendo da arquitetura da rede neural (CONVOLUTIONAL. . . ,) e seu tipo de operação.

Por exemplo, redes neurais feed-forward, redes convolucionais, redes neurais recorrentes, autoencoders e deep beliefs são tipos distintos de sistemas de aprendizado profundo.

Um MLP consiste em pelo menos três tipos de camadas: camada de entrada, camadas ocultas e camada de saída. Pode haver mais de uma camada oculta e cada uma contém um ou mais neurônios.

- **Camada de entrada**

  Essa camada recebe a entrada de uma fonte externa, como, por exemplo, imagens. Os inputs para esta camada são as características. Os neurônios na camada de entrada não realizam nenhuma computação e apenas passam seus inputs para a próxima camada. O número de neurônios na camada de entrada é igual ao número de características, que, no caso da visão computacional, equivale ao número de pixels. A Figura 4 mostra uma arquitetura de rede neural genérica.

- **Camadas ocultas**

  As camadas entre as camadas de entrada e saída são chamadas de camadas ocultas. Uma rede neural deve ter pelo menos uma camada oculta pois é essa camada que gera o aprendizado e há os cálculos necessários nos neurônios para o aprendizado. A complexidade e exigência de poder computacional aumenta com o número de camadas.

- **Camada de saída**

  A camada de saída é a última camada, e o número de neurônios na camada de saída depende do tipo de situação para a qual a rede neural foi projetada. Para regressão, onde a rede deve fazer a previsão de um valor contínuo, a camada de saída tem apenas um neurônio. Já para problemas de classificação entre classes, a camada de saída tem o mesmo número de neurônios que o número de classes.

2.3.5 **Funções de ativação**

A função de ativação determina se o neurônio associado deve ser ativado ou não com base na relevância da entrada do neurônio para o modelo e normaliza a saída de cada neurônio para um valor na faixa [0,1] ou [-1,1].

Várias funções matemáticas são usadas como ativação. As seguintes funções de ativação são utilizadas pelo TensorFlow:

- **Função de Ativação Linear**

  A função de ativação linear não é utilizada no aprendizado profundo pois apresenta problema com a derivação. É comum utilizar o método de retropropagação, que emprega uma técnica chamada descida de gradiente. Nesta técnica, há o cálculo da

derivada de primeira ordem da entrada, o que, no caso da função linear, resulta em zero e torna impossível retroceder aos pesos das entradas.

Outro problema é a restrição à Linearidade: a última camada será uma função linear da primeira camada, logo a rede é equivalente a apenas uma camada, o que não é adequado para resolver problemas complexos.

$$f(x) = x_0 + w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

- **Função de Ativação Sigmóide ou Logística**

  A função sigmóide resulta em um valor entre 0 e 1, o que torna a saída menos sucinta à variações abruptas da entrada. Outra vantagem é que não gera um valor constante a partir de uma derivada de primeira ordem, comportamento adequado para deep learning com retropropagação. A maior desvantagem da função sigmóide é que a saída não muda entre valores de entrada grandes ou pequenos, o que a torna inadequada para casos em que o vetor de características recebidas pela função contém valores grandes ou pequenos. Uma alternativa é normalizar seu vetor de características para ter valores entre -1 e 1 ou entre 0 e 1.

$$s(z) = \frac{1}{1 + e^{-z}}$$

- **TanH/Tangente Hiperbólica**

  Semelhante à função de ativação sigmóide, com a diferença de que o TanH é centrado em zero, e, como consequência, ela modela entradas com valores pequenos, grandes e neutros.

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

- **Unidade Linear Retificada (Rectified Linear Unit, ReLU)**

  Caso z seja positivo, a função ReLU considera esse valor como saída, e, se for negativo, a saída é zero. A saída varia entre 0 e infinito, e a vantagem desta função de ativação que ela é computacionalmente eficiente e permite que a rede convirja, ou seja, encontre os pesos ideais para operação do modelo, rapidamente. Além disso, o ReLU é não linear.

  A maior desvantagem da função ReLU é que o gradiente da função se torna zero para entradas zero ou negativas, comportamento inadequado para retropropagação quando a entrada possui valores negativos. Esta funçãoé amplamente utilizado no treinamento da maioria dos modelos de visão computacional visto que os pixels de imagem não têm valores negativos.

$$f(z) = \max(0, z)$$

- **Leaky ReLU**

  Leaky ReLU oferece uma pequena variação do ReLU. Em vez de anular o valor negativo de z, ele o multiplica por um número pequeno, como 0,01. O Leaky ReLU tem pequenos valores em sua parte negativa e permite a retropropagação para entradas negativas, com a desvantagem de que o resultado do Leaky ReLU não é consistente nesses valores.

- **Unidade Linear Exponencial Escalonada (Scaled Exponential Linear Unit, SELU)**

  A função SELU gera saídas "auto-normalizadas", com média 0 e desvio padrão 1. Isso implica que toda a rede exibe comportamento normalizado até a saída na última camada.

  Com esta função, o aprendizado é altamente robusto e permite treinar redes com muitas camadas, visto que a auto-normalização torna-se eficiente em termos de computação e tende a convergir mais rapidamente. Outra vantagem é que ela supera os problemas de gradientes com variações abruptas ou que diminuem bruscamente quando as características de entrada são muito altas ou muito baixas.

$$f(x) = 1.05070098 \cdot \begin{cases} 1.67326324(e^x - 1) & \text{para } x \leq 0 \\ 0 & \text{para } x > 0 \end{cases}$$

- **Função de Ativação Softplus**

  A função de ativação softplus aplica suavização ao valor da função de ativação z e utiliza o logaritmo do exponencial, também é chamado de função SmoothReLU. A primeira derivada da função softplus é a mesma que a função de ativação sigmoidal.

$$\ln(1 + e^z)$$

- **Softmax**

  Função que recebe um vetor de números reais como entrada, normaliza os valores presentes em uma distribuição probabilística e gera saídas no intervalo (0,1). É frequentemente usado como a ativação para a camada de saída de uma rede neural de classificação, e seu resultado é interpretado como a probabilidade prevista de cada classe.

$$f(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

2.3.6 Função de Erro

No aprendizado de máquina o erro é a diferença entre o resultado esperado e o resultado previsto.

Erro = Resultado Esperado - Resultado Previsto.

O objetivo do aprendizado da rede é calcular valores otimizados de pesos, ou seja, que resultam em erros mínimos. Durante o processo de aprendizado há o ajuste de pesos de forma iterativa.

O ponto em que a derivada primeira da função de erro é zero é o objetivo ideal que indica um mínimo e encontrar os pesos onde a função de erro é mínima é tarefa das funções de erro, também são conhecidas como funções de perda, e são enquadradas em três principais categorias:

Funções de Perda para Regressão: Para treinar modelos que desejam prever resultados contínuos.

Funções de Perda para Classificação Binária: Treina modelos que preveem entre dois resultados, útil em casos como, por exemplo, detectar a presença de doenças.

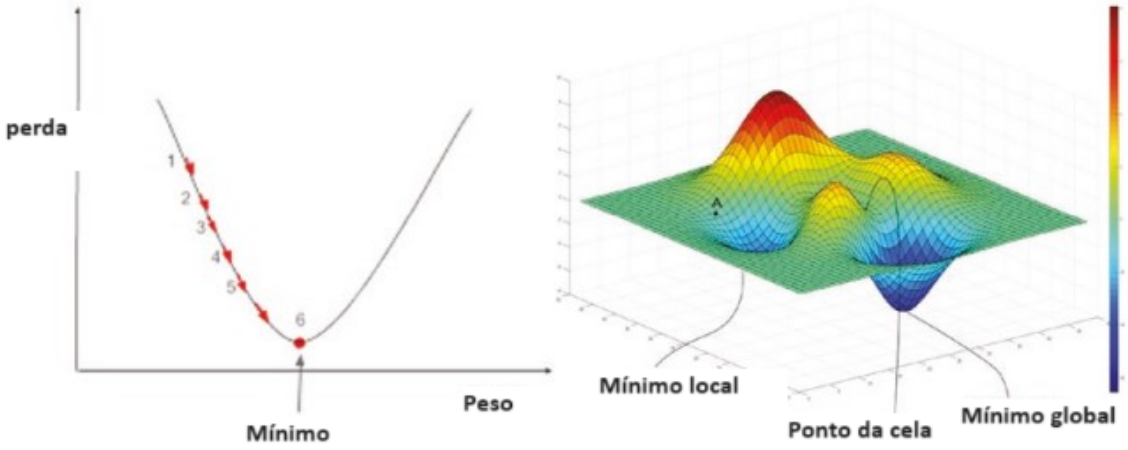
Funções de Perda para Classificação Multiclasse: Usadas quando os modelos precisam prever várias classes, como por exemplo na detecção de objetos.

2.3.7 Algoritmos de Otimização

O algoritmo de aprendizado otimiza a função de perda para encontrar pesos que minimizem o valor da perda iterativamente. A função que otimiza a função de perda é chamada **algoritmo de otimização** ou **otimizador**, que oferecem diferentes graus de precisão e velocidade. Alguns conceitos são comuns aos algoritmos de otimização. (GOODFELLOW; BENGIO; COURVILLE, 2016)

Mínimos Locais e Globais ocorrem pois há situações em que podem haver dezenas ou até centenas de características para as quais os pesos precisam ser aprendidos, e, nesses casos, a curva da função pode ter vários pontos que pareceriam mínimos, chamados mínimos locais. O objetivo do algoritmo de descida de gradiente é encontrar o mínimo global.

Figura 5 – Função de perda com o gradiente em direção ao mínimo



Modificado de (ANSARI, 2020)

A **Taxa de Aprendizado** precisa ser cautelosamente selecionada, pois um valor grande pode fazer com que o algoritmo oscile e não encontre o mínimo, enquanto um valor pequeno torna o aprendizado lento. Um bom valor inicial para a taxa de aprendizado está entre 0,01 e 0,1, e deve ser ajustado conforme se faz necessário.

Regularização é uma maneira de controlar o efeito de um ou mais pesos grandes, que interfeririam na previsão geral. O parâmetro chamado de regularização é adicionado na função de custo para equilibrar os pesos que podem impactar a previsão, penalizando os pesos grandes para reduzir seu impacto. A seguir estão alguns algoritmos de otimização e suas características:

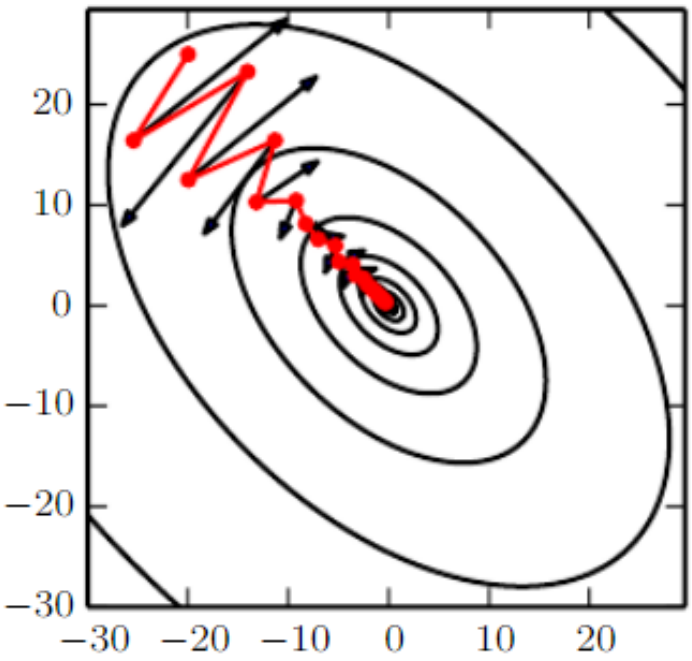
- **Descida de Gradiente (Gradient Descent)** Encontra pesos nos quais a função de perda, ou função de custo, é mínima. O algoritmo calcula a derivada e se move ao longo da curva, com sentido de movimento decidido pelo gradiente negativo, e a taxa de aprendizado determina o tamanho desse deslocamento a cada iteração até que o algoritmo encontre o custo mínimo final.

- **Descida de Gradiente Estocástica (SGD)** A descida de gradiente calcula os gradientes de todas as amostras de treinamento em cada iteração, o que pode ser computacionalmente custoso e até mesmo pode não ser viável. A SGD contorna esse problema ao calcular os gradientes de um pequeno subconjunto de um conjunto do treinamento que pode caber facilmente na memória através da randomização do conjunto de dados de entrada para eliminar qualquer viés. É então calculado o gradiente de uma única amostra de dados selecionado aleatoriamente ou de um pequeno segmento dos dados.

- **SGD com Momentum** A SGD com momentum é uma extensão que controla a oscilação e acelera a convergência, especialmente em torno de mínimos locais profundos, visto que o momentum é um método que controla a oscilação através do deslocamento do gradiente com operações distribuídas e paralelas que ajudam a convergir o SGD mais rapidamente.

  Segundo (GOODFELLOW; BENGIO; COURVILLE, 2016), o Momentum tem como objetivo resolver a variância no gradiente estocástico. Na Figura 5, as linhas de contorno representam uma função de perda quadrática. O caminho vermelho que corta as curvas indica o caminho seguido pela regra de aprendizado do momentum ao minimizar essa função. Em cada etapa ao longo do caminho, é desenhada uma seta indicando a direção que o gradiente desceria naquele ponto. Podemos observar que uma função objetivo quadrática mal condicionada se assemelha a um vale ou desfiladeiro longo e estreito com lados íngremes. O momentum atravessa corretamente o comprimento do desfiladeiro, enquanto os passos do gradiente perdem tempo se movendo de um lado para o outro ao longo do eixo estreito do desfiladeiro.

Figura 6 – Momentum utilizado para encontrar o mínimo do erro



Fonte: (GOODFELLOW; BENGIO; COURVILLE, 2016)

- **Adaptive Gradient Algorithm (Adagrad)** O algoritmo Adagrad aborda o problema de definir a taxa de aprendizado calculando um valor adequada para cada parâmetro, atribuindo uma taxa maior para características pouco frequentes e uma taxa de aprendizado menor para características mais frequentes, o que melhora o desempenho em problemas com gradientes esparsos, como em visão computacional ou processamento de linguagem natural (NLP).

  Uma desvantagem é que a taxa de aprendizado adaptativa tende a ficar muito pequena ao longo do tempo, o que aumenta o tempo de treinamento do modelo consideravelmente.

- **RMSProp** O RMSProp apresenta uma melhoria em relação ao SGD com momentum, pois restringe o movimento dos gradientes verticalmente, visto que, em uma curva íngreme, um pequeno movimento na direção horizontal causará um grande movimento na direção vertical. Assim, o movimento em ambas as direções não será desigual, convergindo ao ponto mínimo mais rapidamente.

- **Adaptive Moment (Adam)** Projetado especificamente para aprendizado profundo, é um dos otimizadores mais utilizados, pois combina o SGD com momentum e o RMSProp: atualiza os pesos da rede iterativamente com base nos dados de treinamento, ao invés de adaptar as taxas com base na média do primeiro momento, como no RMSProp, esse otimizador utiliza a média dos segundos momentos dos gradientes.

De forma geral, o algoritmo Adam é fácil de implementar, eficiente computacionalmente, com requisitos de memória baixos, invariante à escalonagem diagonal dos gradientes, adequado para problemas grandes em termos de dados, parâmetros, objetivos não estacionários e problemas com gradientes ruidosos ou esparsos com parâmetros que geralmente exigem pouco ajuste. Fonte: https://arxiv.org/pdf/1412.6980.pdf

2.3.8 **Retropropagação**

O treinamento de uma rede neural necessita dos seguintes elementos: dados ou características de entrada; uma rede neural multicamada; uma função de erro. A rede atribui pesos iniciais a cada característica de entrada, e, através do algoritmo de otimização, como SGD ou Adam, a função de erro é otimizada para calcular o erro mínimo e atualizar os pesos.

É então estimado o erro para que os pesos sejam atualizados. No método de retropropagação, os gradientes dos pesos são calculados primeiro na última camada e os gradientes da primeira camada são calculados por último. Os cálculos parciais do gradiente são reutilizados no cálculo do gradiente para a camada anterior. Esse fluxo reverso dos dados de erro resulta em uma computação eficiente do gradiente em cada camada, os cálculos de gradientes não são feitos independentemente em cada camada.

O erro da última camada é calculado primeiro pois mapeia as variáveis alvo do conjunto de dados rotulado e, na camada oculta, não há variáveis alvo.

2.3.9 **Redes Neurais Convolucionais (CNNs)**

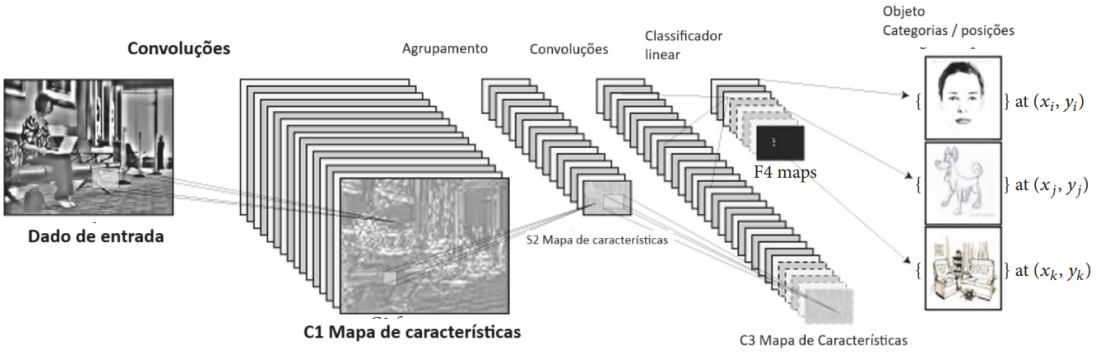
Uma Rede Neural Convolucional (CNN ou ConvNet) é um tipo de rede neural profunda projetada para reconhecer padrões em dados que possuem uma forma de grade, como imagens, e é particularmente eficaz em tarefas relacionadas à visão computacional, como reconhecimento e segmentação de objetos, visto sua capacidade de aprender características hierárquicas e invariantes de translação em dados de grade, problema comumente encontrado em outras abordagens. São compostas por camadas convolucionais, de pooling (agrupamento) e totalmente conectadas. (VOULODIMOS, 2018)

- **Camada Convolucional (Convolutional Layer)**: aplica filtros, chamados de kernels, às partes da entrada, procurando padrões locais. Essa operação de convolução permite que a rede aprenda características específicas, como bordas, texturas ou padrões mais complexos.

- **Camada de Pooling (Pooling Layer)**: reduz as dimensões da entrada e torna a representação mais compacta, e em sua operação geralmente envolve a seleção do valor máximo, max pooling, ou a média, average pooling, em uma região.

- **Camada Totalmente Conectada (Fully Connected Layer)**: após as camadas convolucionais e de pooling, a rede pode ter uma ou mais camadas totalmente conectadas, que realizam a classificação final com base nas características aprendidas.

Figura 7 – Rede Neural Convolucional sobre imagem



Modificado de Athanasios Voulodimos, 2018

2.3.10 **Saídas dos modelos de detecção de objetos**

Os termos "bounding box,keypoints"e "masks"referem-se a diferentes aspectos da representação e identificação de objetos em uma imagem que fazem parte das saídas e resultados dos modelos de detecção. Modelos mais complexos, e, consequentemente, mais exigentes computacionalmente, produzem mais dessas saídas, e é necessária a escolha de uma rede que proporcione as informações pertinentes à aplicação.

- **Bounding Box (Caixa Delimitadora)**: caixa retangular que envolve ou delimita a área onde o modelo infere que o objeto está localizado em uma imagem, usado para indicar a localização aproximada de um objeto especificando as coordenadas, geralmente localizadas nos cantos, da caixa delimitadora.

- **Keypoints (Pontos-chave)**: pontos específicos em um objeto ou região de interesse que têm um significado distintivo escolhidos por serem identificáveis e usados para descrever as características de uma área e localizar pontos anatomicamente significativos em objetos, por exemplo, cantos de um rosto. São úteis para estimar a pose ou orientação de um objeto.

- **Máscaras (Masks)**: imagens binárias que indicam a presença ou ausência de um objeto em cada pixel, usadas para segmentar precisamente o que pertence a um objeto específico, útil para quando é necessário distinguir a localização dentro da caixa delimitadora.

Figura 8 – Máscaras e caixas delimitadoras.

Fonte: (ANSARI, 2020)

Figura 9 – Pontos chave na imagem.

Fonte: (ANSARI, 2020)

2.4 Rastreamento de Objetos

Após a etapa da detecção de objetos, o modelo retorna as coordenadas dos pixels das extremidades do retângulo que cerca aquele elemento. Essas coordenadas são suficientes para orientar a posição no caso em que há apenas um item de interesse na imagem; entretanto, isso não se aplica ao caso em que há múltiplos exemplares da classe de interesse em uma mesma imagem. Um dos campos da visão computacional que foca sobre esta questão é o rastreamento de objetos.

Duas técnicas tradicionais para realizar o rastreamento são: o deslocamento médio (mean shift) e o fluxo óptico (optical flow).

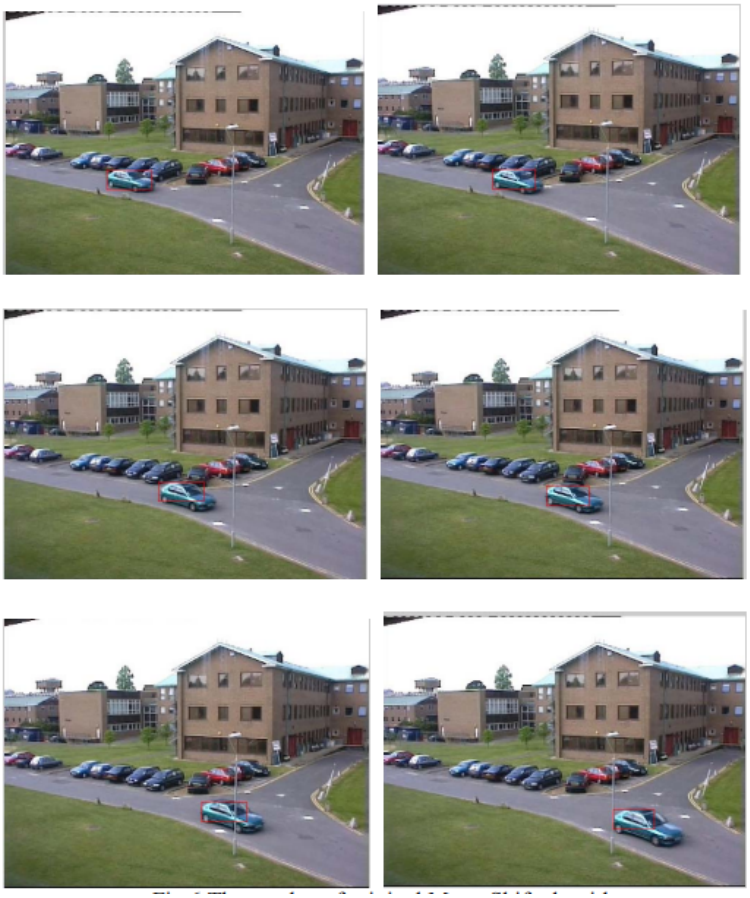
2.4.1 Deslocamento Médio

O deslocamento médio funciona a partir de detectar o objeto de interesse e extrair um padrão morfológico, suas coordenadas e seu tamanho através da análise de pixels em posições chave. Esse padrão é então procurado no próximo quadro em uma região, conhecida como vizinhança, ao redor de onde estava o centro do objeto no quadro anterior,

e a porção de pixels nessa vizinhança com a melhor correspondência recebe a atribuição da identidade do objeto. (ZHAO; WANG; HAN, 2013)

Entretanto, o método do deslocamento médio não é confiável caso o objeto saia da região da vizinhança do objeto de interesse, então, caso ele se mova muito rapidamente, será perdido. Além disso, também podemos destacar uma forte ineficiência em caso de oclusões.

Figura 10 – Resultado do algoritmo de deslocamento médio sobre carro



Fonte: Ming Zhao, 2013

2.4.2 **Fluxo Óptico**

Outro método tradicional é o fluxo óptico. Ele difere do deslocamento médio pois leva em consideração o movimento relativo dos objetos ao decorrer dos quadros anteriores. É gerado um vetor de movimento entre o quadro presente e o quadro anterior, o que possibilita o uso desses vetores para seguir e até prever a trajetória do objeto no próximo frame. (WU et al., 2018)

Contudo, apesar da boa performance vista nos métodos tradicionais, eles são computacionalmente complexos e sujeitos a ruídos, especialmente no caso do fluxo óptico, onde

detecções erradas do centro do objeto, seu tamanho e oclusões em uma pequena parte dos quadros produz erros de grandes magnitudes.

Figura 11 – Campos de fluxo de imagem previstos pelo algoritmo de fluxo óptico

Fonte: Junjie Wu, 2012

2.4.3 **Algoritmos de Predição de Trajetória**

Para contornar a complexidade computacional e a sensibilidade a ruídos e oclusões, diversos trabalhos pautados no rastreamento de objetos já foram propostos.

Essencialmente, o objetivo desses algoritmos é, por meio de uma sucessão de listas de coordenadas que são atualizadas a cada inferência, atribuir um número de identificação para cada conjunto de coordenadas a partir de comparações com as listas anteriores. Na Tabela 1 há o comparativo dos algoritmos mais conhecidos, com métricas coletadas através do Local Metrics for Multi-Object Tracking (VALMADRE et al., 2021).

2.4.4 **SORT**

O algoritmo SORT, Simple Online Real-time Tracking, foi o escolhido devido a sua simplicidade em termos de biblioteca, o que torna seu desempenho substancialmente melhor quando utilizado em sistemas onde o poder computacional é um recurso limitado e é necessária para a operação o seu uso em tempo real. (BEWLEY et al., 2016)

- **Detecção**

Tabela 1 – Comparação de algoritmos de rastreamento de objetos

Algoritmo	Vantagens	Desvantagens
Tracktor++	Boa acurácia	3 FPS, ideal para execuções que não precisam ser em tempo real
Track RCNN	Boa acurácia e com segmentação	1.6 FPS
JDE	12 FPS, performance para tempo real	Baixa resolução, 1088x608
SORT	Boa velocidade e acurácia	Sensível à oclusões e troca identidade de objetos com frequência
DeepSORT	16 FPS, boa velocidade, acurácia e lida bem com oclusões	Necessita de uma GPU para uso em tempo real

A detecção é feita através da técnica de visão computacional selecionada, e, quanto menores os erros e maior a qualidade das coordenadas, melhor serão os resultados ao final

- **Estimativas**

  Nesta etapa, é utilizado o filtro Kalman. O filtro Kalman utiliza um modelo que atribui ao objeto de interesse um modelo de movimento que se baseia em velocidade constante, e, caso haja oclusão, irá atribuir estimativas de acordo com os dados de movimento que foram previamente captados.

  Quando é possível visualizar o elemento de interesse, é fornecido ao modelo os dados de movimentação para alimentá-lo. No caso onde há obstrução parcial, é utilizado tanto o modelo quanto os dados captados pelo sensor para predizer a posição, e, caso esteja totalmente oculto do sensor, os dados do modelo são utilizados na predição.

- **Associação de elementos**

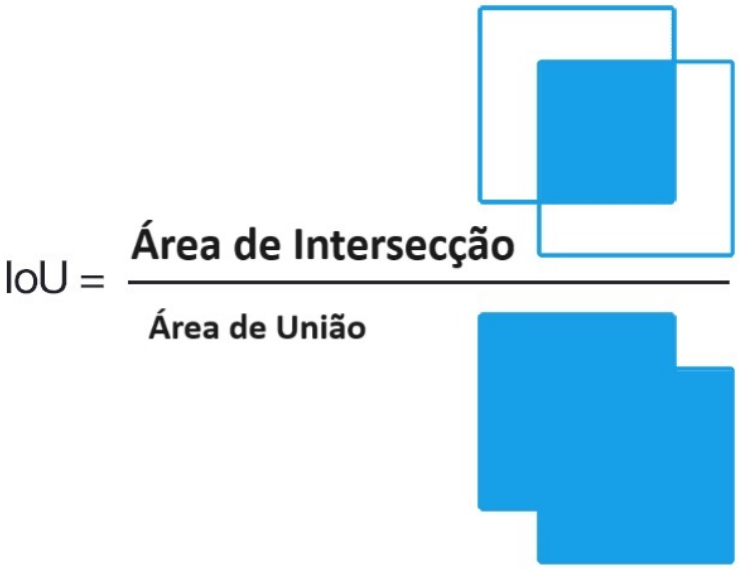
  Cada conjunto de coordenadas correspondentes às delimitações dos objetos detectados no último quadro têm, através do filtro Kalman, suas coordenadas previstas comparadas com os objetos presentes detectados no quadro atual.

  A matriz de custo de atribuição é então calculada como a distância de intersecção sobre união (IoU) entre as localizações previstas e as existentes. A atribuição é feita usando o algoritmo Húngaro, e, quanto mais sobreposição entre as caixas delimitadoras houver, maior o percentual de confiança inferido sobre aqueles dados como pertencentes ao mesmo objeto.

- **Ciclo de vida da identidade dos elementos**

  A entrada e saída de objetos implica na necessidade de criar ou eliminar identidades conforme necessário. Elementos que possuem uma sobreposição com outras caixas

Figura 12 – Intersecção sobre União



Modificado de (ROSEBROCK,)

delimitadoras menores que um valor de IOU mínimo indicam a existência de um item ainda não rastreado, e é então criado um rastreador.

O rastreador tem, inicialmente, seu modelo de velocidade ajustado para zero com um grande grau de incerteza, e passa por um período probatório onde o algoritmo tenta associá-lo a detecções previamente existentes até que acumule evidências suficientes para classificá-lo como sendo, de fato, um elemento inédito. Dessa forma, são evitados falsos positivos.

Além disso, caso um rastrador não detecte seu correspondente objeto por uma determinada quantidade de quadros, aquela identidade será eliminada, e, caso apareça novamente, será atribuído a ele uma nova identificação.

- **SORT e DeepSORT**

  DeepSORT tem 45% menos trocas de identidade (WOJKE; BEWLEY; PAULUS, 2017), entretanto, para utilizá-lo em tempo real é necessária uma GPU. Apesar da placa utilizada no projeto dispor de uma GPU, ela não possui capacidade computacional suficiente para utilizar o DeepSORT e realizar inferências do modelo de detecção de objetos em tempo real.

2.5 TensorFlow

TensorFlow é uma biblioteca de open-source para aprendizado de máquina e aprendizado profundo desenvolvida pelo Google Brain baseado no Keras, uma API de alto nível para construção e treinamento de modelos de redes neurais (ABADI et al., 2015). Teve

início em 2011 e foi aberto ao público em 2015. Projetado para simplificar e otimizar a implementação de modelos de aprendizado de máquina com arquiteturas versáteis que possibilitam a criação e treinamento de redes neurais profundas para uma variedade de aplicações, desde visão computacional até processamento de linguagem natural.

O TensorFlow manipula dados na forma de tensores, que são estruturas multidimensionais semelhantes a arrays. Apesar de sua base de operações de baixo nível, ele fornece APIs de alto nível, como Keras, para a construção rápida de modelos complexos, sendo amplamente utilizado em uma variedade de campos e aplicações, como:

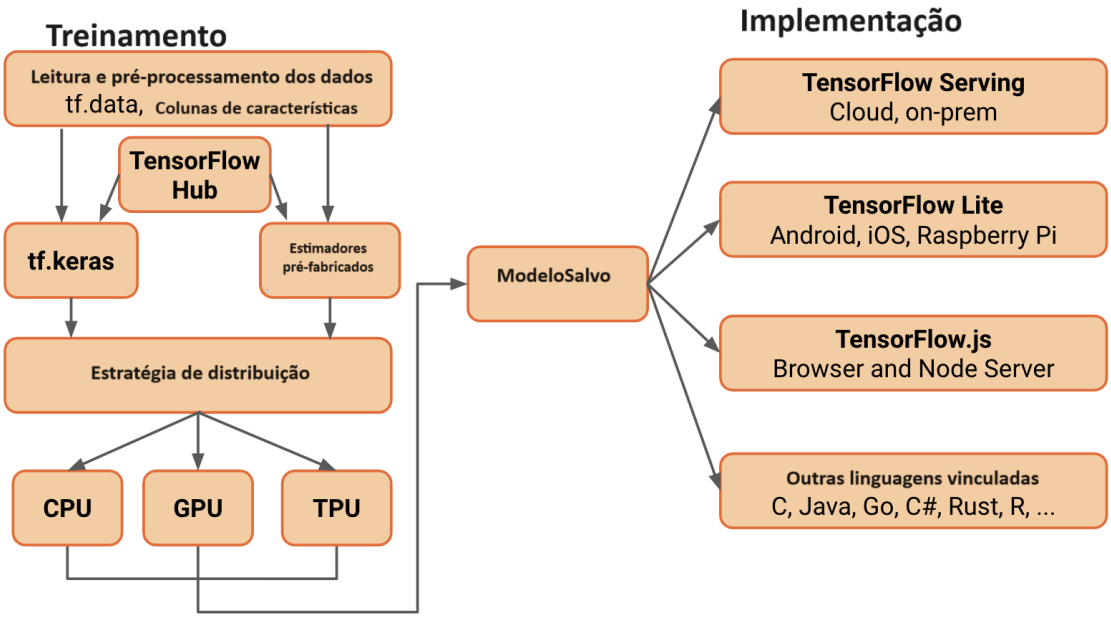
- **Aprendizado Profundo:** treinamento de redes neurais profundas em aplicações diversas.
- **Análise Preditiva:** prever tendências e otimizar processos.
- **Processamento de Linguagem Natural (NLP):** análise de texto, tradução automática e criação de chatbots.
- **Reconhecimento de Fala:** sistemas que transcrevem voz para texto para diversos usos, como a combinação dessa funcionalidade com outros algoritmos de controle.
- **Visão Computacional:** segmentação de imagem, reconhecimento e detecção de objetos, que será o foco desta tese.

O Keras foi inicialmente desenvolvido como uma biblioteca independente, mas, a partir do TensorFlow 2.0, foi incorporado como sua API oficial. Com intuito de fornecer uma interface intuitiva para a construção de redes neurais, foi projetado para ser modular, o que facilita a modificação e implementação de diferentes arquiteturas de rede. Dessa forma, o TensorFlow serve como uma aplicação de alto nível para definição, treinamento e avaliação de modelos e permite a fácil transição de protótipos para implementações em larga escala, pois suas configurações são flexíveis para diferentes requisitos de sistemas, com capacidade de operar em CPUs, GPUs e TPUs proporciona versatilidade na escolha de hardware de acordo com a aplicação e recursos disponíveis. Há também uma versão da biblioteca menos computacionalmente exigente, o TensorFlow Lite.

2.6 ROS

O ROS é uma plataforma de software open-source projetada para facilitar o desenvolvimento de robôs inicialmente desenvolvido pela Willow Garage, uma empresa de pesquisa em robótica, agora é mantido pela Open Robotics, organização sem fins lucrativos. Seu código fonte é aberto e permite que desenvolvedores contribuam, adaptem e aprimorem a plataforma, sendo ela compatível com diversos sistemas operacionais, incluindo Linux, com

Figura 13 – Diagrama conceitual do funcionamento do TensorFlow



Modificado de (TENSORFLOW. . . ,)

foco na versão Ubuntu, macOS e algumas implementações para Windows, o que aumenta a acessibilidade do ROS.

A plataforma promove o compartilhamento de códigos entre desenvolvedores através de pacotes de software de diferentes projetos e equipes, estratégia que acelera o desenvolvimento ao eliminar a necessidade de recriar soluções para problemas que aparecem de forma frequente no desenvolvimento de projetos.

Ademais, há o fornecimento de uma camada de abstração que separa o software do hardware subjacente, possibilitando a escrita de códigos para tarefas específicas sem conflitos com as especificidades de hardware. Nesse sentido, há também a facilitação da comunicação entre diferentes componentes do sistema, permitindo que módulos distintos troquem informações de maneira eficiente e sem problemas, o que será de utilidade para que o algoritmo desenvolvido nesta tese possa ser facilmente integrado ao sistema alvo, pois sua arquitetura modular facilita a adaptação a diferentes hardwares.

A plataforma possui diferentes funcionalidades para comunicação e integração de algoritmos que compõem o software do robô que serão úteis para o projeto:

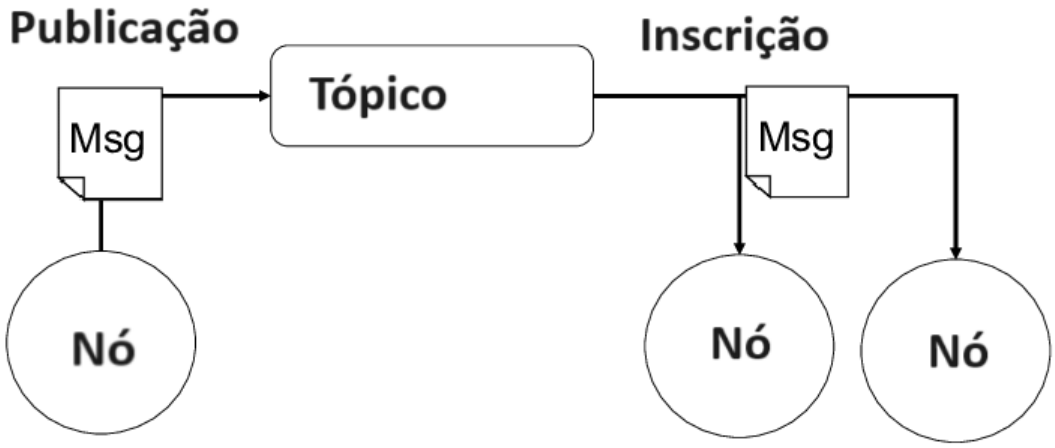
- **Nós (nodes):** processos executáveis individuais que realizam tarefas específicas diversas, desde controlar um sensor até coordenar todo o comportamento do sistema. Os nós se comunicam por meio de tópicos ou serviços, permitindo uma arquitetura distribuída.

- **Tópicos (Topics):** canais de comunicação assíncrona através dos quais os nós trocam mensagens. Nós podem publicar ou assinar no tópico para, respectivamente,

enviar ou receber as mensagens e seus dados.

- **Mensagens (Messages):** definem o formato das informações e variáveis transmitidas entre nós. Cada tópico tem um tipo de mensagem, e, consequentemente, um tipo de dado, associado que especifica como as informações devem ser organizadas e interpretadas.
- **Serviços (Services):** permitem comunicação síncrona, onde um node solicita a execução de uma função a outro nó, como solicitar dados de sensores ou enviar comandos de controle.

Figura 14 – ROS: Mensagens entre nós Publisher e Subscriber



Modificado de (YAMASHINA et al., 2015)

3 DESENVOLVIMENTO

Com a fundamentação teórica a respeito das ferramentas computacionais, bibliotecas e plataformas utilizadas, o presente capítulo tem o objetivo de expor as etapas do procedimento de desenvolvimento do algoritmo de visão, funcionalidades, limitações encontradas e aspectos da lógica por trás do código elaborado.

3.1 Hardware utilizado

3.1.1 **Máquina local**

Apesar da aplicação destino do algoritmo ser em um sistema embarcado, muitas tarefas de desenvolvimento e treino podem ser realizadas em outra máquina que dispõe de maiores recursos de processamento. Para o desenvolvimento da tese, foi usado um notebook com as seguintes especificações:

- **Nome do Sistema Operacional: Microsoft Windows 11 Home Single Language**
- **Modelo do sistema: Nitro AN515-57**
- **Processador: 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz, 2304 Mhz, 8 Núcleo(s)**
- **GPU: NVIDIA GeForce RTX 3050 Laptop GPU**

É importante levar em consideração o sistema operacional e a GPU, visto que a maioria das bibliotecas para desenvolvimento de aplicações em visão computacional e os principais usos de ROS são projetados para uso em Linux. Portanto são necessárias adaptações para uso do sistema operacional disponível, Windows 11.

3.1.2 **Raspberry Pi 4B**

A Raspberry Pi 4B, da Raspberry Pi Foundation (RASPBERRY PI FOUNDATION,), é uma placa computacional compacta com diversas aplicações que visam a portabilidade e acessibilidade, como em projetos de Internet das Coisas (IoT), servidores domésticos, aplicações industriais leves e robôs. Algumas especificações da placa utilizada no projeto:

- **Sistema operacional: Raspberry Pi OS with desktop, October 10th 2023**

  O sistema operacional oficial da Raspberry é o Raspberry Pi OS, entretanto, é possível instalar outros, visto que ele é alojado no cartão SD que deve ser inserido na placa.

- **Debian version: 12 (bookworm)**
- **Processador: quad-core ARM Cortex-A72, 64 bits**

- **RAM: 4GB**

A Raspberry Pi, através dos sistemas operacionais baseados em Linux, é compatível com Python, TensorFlow e ROS, o que permite que ela implemente projetos envolvendo redes neurais, visão computacional e outras aplicações de ML, e sua compatibilidade com ROS possibilita seu uso como o cérebro de um robô, integrando sensores, motores e algoritmos em um ecossistema coeso.

Para captar as imagens e fornecê-las ao algoritmo foi obtida uma das câmeras oficiais da Raspberry, a câmera para Raspberry Pi Rev 1.3.

Figura 15 – Raspberry Pi 4B

Fonte: Autor

3.2 Treinamento de modelo

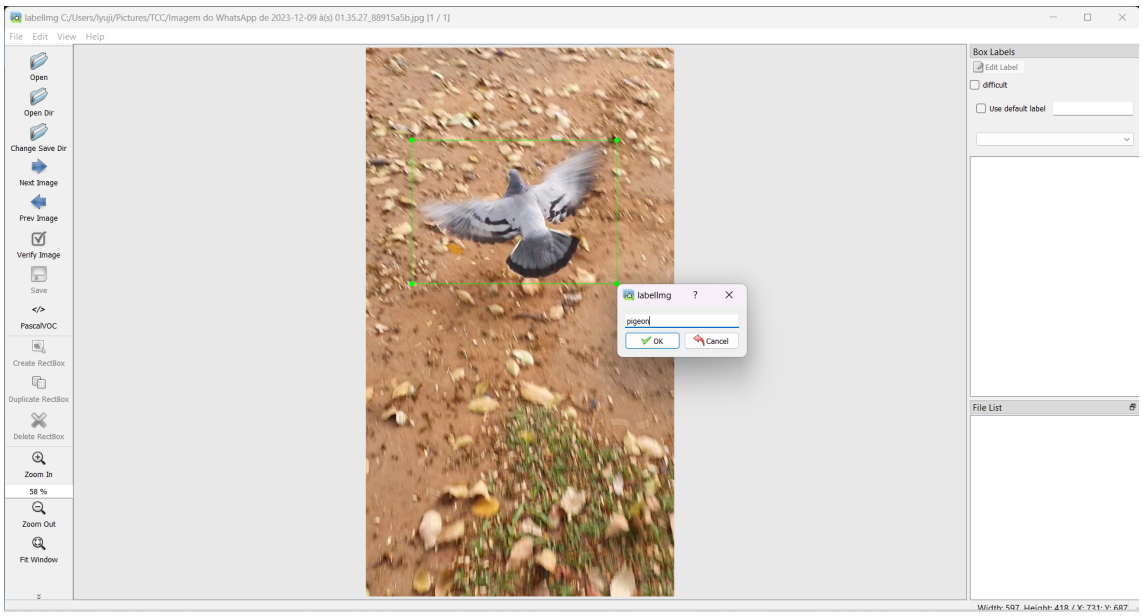
O treinamento do modelo de aprendizado profundo é um processo computacionalmente custoso e normalmente é necessário dias e, em alguns casos, até mesmo semanas, para que seja concluído e obtenha resultados satisfatórios. Entretanto, é possível utilizar modelos existentes com pesos bem definidos para que serviam como ponto de partida do treinamento de uma rede. O TensorFlow, em seu repositório oficial, disponibiliza alguns desses modelos.

O repositório contém modelos treinados para realizar a inferência de diversas classes. Ainda assim, em termos de projeto, o intuito nesta tese de passar pelo processo de treinamento é para que, caso o robô tenha o objetivo de seguir um item que não está presente nos modelos pré-treinados disponíveis publicamente, seja possível adicionar novos elementos a partir do treino de uma rede. Mais detalhes sobre treinamento de modelos podem ser encontrados no Apêndice C.

É pertinente a ressalva de que objetos que possuem características geométricas bem definidas e que pouco variam com mudanças no ângulo de captura e iluminação são expressivamente mais simples para o aprendizado. Para testar um cenário de maior complexidade,

o objetivo do treinamento da rede será a detecção de pombos, visto que a aquisição de imagens deles em ambiente urbano é simples e eles tendem a variar em cor e forma de acordo com diferentes direções de observação.

Figura 16 – Exemplo de imagem utilizada para realizar o treinamento da rede



Fonte: Autor

3.2.1 **Dataset**

Para que a rede seja treinada são necessárias não apenas as imagens, mas também a indicação dos objetos de interesse presentes, suas coordenadas e categorias dos elementos. Além disso, é necessária a divisão entre imagens para treinamento e para validação, pois se todos os dados fossem usados no treinamento ocorreria o vazamento de informações e o modelo não seria capaz de verificar sua capacidade de realizar inferências, e, por consequência, não seria devidamente otimizado. Uma prática recomendada é coletar uma quantidade acima de 200 imagens e separar em torno de 70% das imagens para treino e 30% para validação.

Antes do treino, é necessário definir o ground truth, parte da imagem que contém o objeto de interesse, e salvar os dados que contém as informações das múltiplas imagens. Isso é feito através da biblioteca **labelimg**.

- **Treinamento** O conjunto de treino é usado para treinar o modelo, ajustando seus pesos e parâmetros iterativamente. É importante não deixar o treinamento acontecer por mais tempo do que o necessário, pois, nesse caso, há o overfitting, que é quando a rede aprende a identificar as imagens utilizadas ao invés de aprender a generalizar.

- **Validação** As imagens de validação fornecem uma avaliação imparcial do desempenho do modelo com diferentes configurações e são usadas como uma estimativa do desempenho futuro durante a execução dele. Após cada iteração de treinamento o desempenho no conjunto de validação é verificado e o modelo é ajustado.

3.2.2 **Uso de modelos pré-treinados**

Treinar um modelo partindo de um já previamente treinado é um processo chamado de "transferência de aprendizado"e é uma forma de economizar recursos computacionais e, consequentemente, tempo. Modelos pré-treinados, especialmente em conjuntos de dados massivos, já aprenderam muitas características úteis e, quando treinado previamente sobre grandes conjuntos de dados, como o EfficientDet treinado sobre o MS COCO utilizado (LIN et al., 2014), tendem a aprender características gerais e úteis.

Além disso, partir dos pesos já estabelecidos faz com que o modelo possa ser ajustado incrementalmente a fim de cumprir tarefas específicas e adaptar o modelo para a nova tarefa sem perder completamente o conhecimento prévio. Não só isso, há também o efeito de regularização, especialmente quando os dados de treino são limitados, ajuda a evitar o overfitting e melhora a capacidade de generalização do modelo.

3.2.3 **TensorBoard**

O TensorBoard é uma ferramenta de visualização de métricas do TensorFlow, uma das suas principais utilidades é monitorar métricas importantes durante o treinamento de um modelo, como a função de perda. Nela, também é possível verificar quais imagens estão sendo analisadas e sendo utilizadas para treinar o modelo no momento.

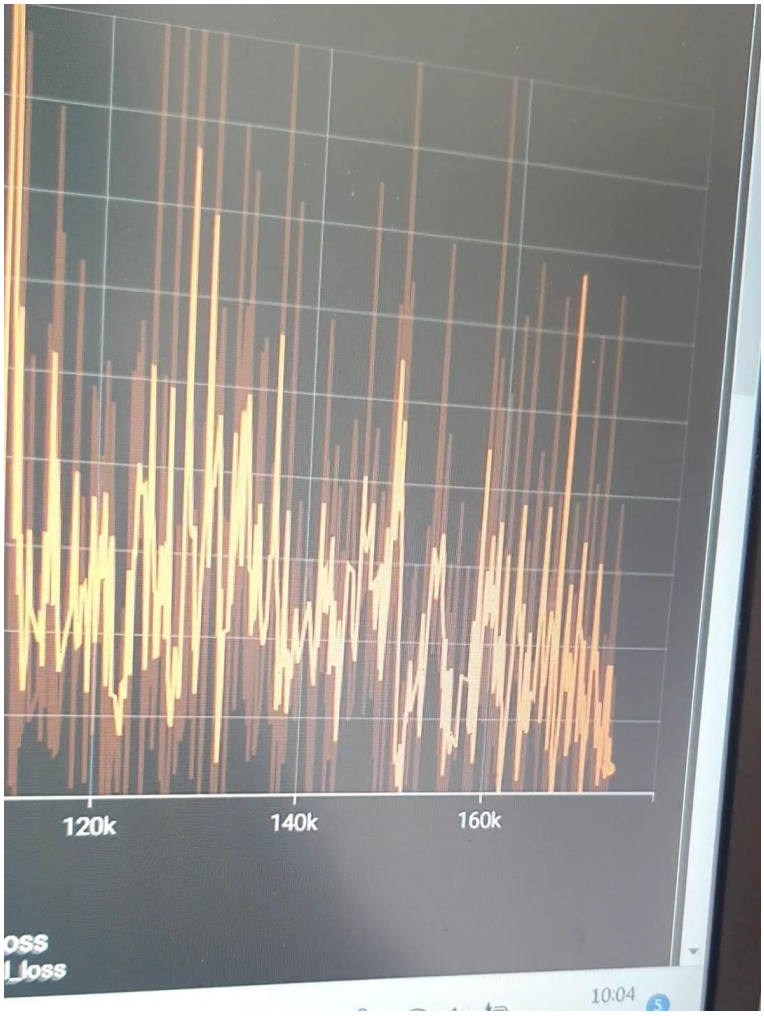
Observar o gráfico da função de perda é uma boa ideia para que se tenha uma boa ideia de quando é o momento adequado para se parar o treino e evitar o overfitting. Após alguns dias, o gráfico tende a convergir para um valor médio e não variar expressivamente. No caso, o treinamento da rede ocorreu durante cinco dias.

Após realizar o treinamento da rede e exportar o modelo é possível executar inferências através dele. No capítulo 4 há o resultado da rede de teste treinada para inferência de pombos urbanos. Entretanto, a fim de aumentar a versatilidade e variedade de objetos para detecção, será utilizado um modelo previamente treinado da biblioteca do TensorFlow que realiza inferências sobre o conjunto de dados MS COCO.

3.3 Modelos e diferenças

Entre diversos aspectos, os modelos de detecção de objetos diferem em aspectos como a arquitetura, ou seja, número de camadas e de perceptrons dentro de cada uma delas, método de geração de regiões de interesse propostas, função de ativação, função de perda,

Figura 17 – Tensorboard - Gráfico da função de perda



Fonte: Autor

eficiência, precisão e velocidade. Na Tabela X, encontram-se alguns modelos e parâmetros comparativos. (KRIZHEVSKY; SUTSKEVER; HINTON, 2012)

Devido à capacidade computacional da placa de computação que será utilizada, o código será desenvolvido sobre um dos modelos de menor exigência de processamento: o SSD MobileNet v2 320x320. Ainda assim, o algoritmo será desenvolvido de forma a permitir facilmente a troca de modelo, caso se faça necessária maior confiabilidade das previsões.

3.4 Algoritmo de Visão

Com o intuito de elaborar um código com a maior versatilidade de usos possíveis e que se adapte a diferentes necessidades e interesses, o algoritmo desenvolvido foi feito de forma a facilitar a implementação de diferentes modelos, configurar parâmetros para operação e modos de funcionamento. Além disso, a placa de computação Raspberry Pi 4B possui recursos computacionais relativamente escassos e, caso outros códigos necessários

Tabela 2 – Comparação de performance de modelos de detecção de objetos

Algoritmo Detector de Objeto	Treinado sobre o Dataset:	mAP	Velocidade do Teste (Segundos por Imagem)	Quadros por Segundo (FPS)	Aplicável em vídeos em tempo real
R-CNN	COCO 2007	66,0%	32,84	0,03	Não
Fast R-CNN	COCO 2007 and 2012	66,9%	1,72	0,60	Não
Faster R-CNN (VGG-16)	COCO 2007 and 2012	73,2%	0,11	9,1	Não
Faster R-CNN (RestNet-101)	COCO 2007 and 2012	83,8%	2,24	0,4	Não
SSD300	COCO 2007 and 2012	74,3%	0,02	46	Sim
SSD512	COCO 2007 and 2012	76,8%	0,05	19	Sim
YOLO	COCO 2007 and 2012	73,4%	0,02	46	Sim
YOLOv2	COCO 2007 and 2012	78,6%	0,03	40	Sim
YOLOv3 608x608	COCO 2007 and 2012	76,0%	0,029	34	Sim
YOLOv3 416x416	COCO 2007 and 2012	75,9%	0,051	19	Sim

Fonte: (ANSARI, 2020)

para o funcionamento do robô necessitem de mais processamento, é possível ajustar o quanto o algoritmo de visão está consumindo através da troca do modelo de inferência ou mesmo adicionando um atraso entre cada execução.

3.4.1 **Ambiente de Desenvolvimento**

Uma das características do TensorFlow e suas aplicações é que suas funções são particularmente sensíveis às versões de outras bibliotecas utilizadas, como, por exemplo, Numpy, SciPy, OpenCV, entre outros. O uso de um recurso de algum desses pacotes em versão mais ou menos atualizada em relação ao TensorFlow facilmente causa conflitos de versões, e é necessária a alteração da versão, seja para mais recente ou mais antigo. No Apêndice D se encontram as versões instaladas que utilizaram todos os recursos empregados no código desenvolvido sem conflitos de versões.

O Conda, desenvolvido pela Anaconda Inc. (ANACONDA. . . , 2020), foi utilizado para gerenciar o ambiente virtual em que os pacotes estão instalados. Um dos benefícios do uso do gerenciador de ambientes e pacotes são os serviços de compatibilidade entre versões e o uso de diferentes distribuições do Python em cada ambiente, pois o TensorFlow

e bibliotecas adjacentes necessitam de variantes específicas que estão disponíveis em apenas algumas versões do interpretador.

3.4.2 **Detecção de Objeto**

O código foi configurado para utilizar, na máquina com o sistema operacional Windows 11, a webcam. A biblioteca do OpenCV permite o uso de câmeras USB através da função empregada, mas não possui suporte para dispositivos fora dessa categoria, e, portanto, são necessárias adaptações para o uso de outros tipos de sensores de imagens.

Foi reservada uma linha para seleção do modelo, que pode ser facilmente substituído por outros do repositório oficial do TensorFlow ou por modelos customizados. O procedimento de download e uso de diferentes modelos se encontram nos Apêndices B e C, respectivamente. Foi escolhido inicialmente o detector de objetos SSD MobileNet v2 320x320.

```
# Caminho ate o modelo
detection_model = load_model('ssd_mobilenet_v2_320x320
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣coco17_tpu-8\\saved_model')
```

A inferência dos objetos é realizada sobre cada quadro do vídeo em tempo real e trata cada captura como uma imagem isolada. O conjunto de imagens sobre o qual o modelo foi treinado, MS COCO, possui 81 classes, e durante a inferência todas elas são indicadas através das caixas delimitadoras.

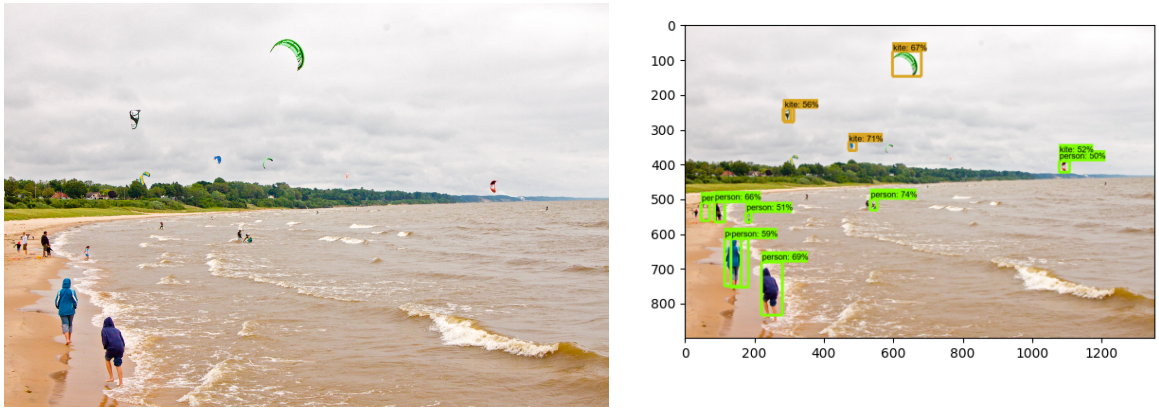
É interessante ressaltar que o modelo gera diversas proposições de objetos com diferentes probabilidades associadas a cada um deles. Caso a probabilidade associada seja baixa, o objeto não é exibido, pois entende-se que o padrão encontrado não é assertivo o suficiente para ser considerado, de fato, um elemento da classe.

Ao final de cada inferência, temos uma saída contendo todos os objetos propostos, probabilidades e coordenadas dos quatro cantos das caixas delimitadoras em uma variável do tipo dicionário chamada output_dict. Entretanto, como descrito no Capítulo 2, esse dicionário não nos fornece uma forma de diferenciar e acompanhar dois ou mais objetos da mesma classe, e isso deve ser feito através de um algoritmo de rastreamento.

```
# Run inference
output_dict = model(input_tensor)
```

3.4.3 **Rastreamento de Objeto**

O algoritmo SORT possui características que o tornam apropriado para aplicações que demandam velocidade, simplicidade, detecção de múltiplos objetos e capacidade de execução em tempo real, mesmo em ambientes computacionais limitados, visto que sua



(a) Imagem original do repositório do TensorFlow (b) Inferência do modelo SSD MobileNet v2

Figura 18 – Inferência do modelo sobre imagem

abordagem baseada em equações e bibliotecas de operações matemáticas base permitem uma resposta rápida a mudanças nas condições do ambiente, projetado para aplicações em que é necessário o rastreamento de objetos em tempo real, fornecendo resultados com baixas latências sem sobrecarregar a capacidade do hardware disponível.

Algumas bibliotecas que compõe o SORT incluem:

- numpy
- scikit-image==0.17.2
- filterpy==1.4.5
- lap==0.4.0

Para utilizar o SORT é necessário fornecer ao objeto criado para realizar o rastreamento a lista de coordenadas das caixas delimitadoras encontradas na imagem, que se encontram após a inferência no output_dict. O rastreador irá retornar as mesmas coordenadas associadas a um número ID, e, através dele, será possível fazer o acompanhamento dos objetos, visto que cada um está associado a um objeto distinto.

Entretanto, em caso de oclusões ou desaparecimento e reaparecimento de um objeto, um novo ID será associado a ele.

```
# Instancia de SORT
mot_tracker = Sort()

# (Codigo que separa as coordenadas de interesse)

# Envia a lista ao algoritmo de tracking
track_bbs_ids = mot_tracker.update(detections)
```

3.4.3.1 **Critérios**

Um importante aspecto do objetivo do robô de seguir um objeto se beneficia de forma expressiva em termos de versatilidade através do uso de redes neurais. Através dos objetos existentes no MS COCO, é possível implementar o critério com base nas classes presentes. Por exemplo, aqui faremos com que o algoritmo observe os elementos da classe "person".

```
# Envia a lista de objetos sob o algoritmo de tracking
# para o algoritmo que seleciona um alvo com o modo de
# operacao 2
target = target_tracking(tracked_objects, 'person', 2)
```

Entretanto, caso haja mais de um elemento desta classe, apenas a categoria não será o suficiente para determinar qual elemento presente será o alvo, e em decorrência do comportamento de troca e novos IDs o robô não conseguirá determinar um número específico de forma autônoma.

Para contornar esse problema, no código foram configurados três modos de operação que podem ser trocados que levam em consideração os IDs da classe alvo:

- **Modo 0** O Modo 0 tornará como alvo o primeiro objeto da classe de interesse que for detectado, ou seja, o com o menor ID presente.

```
if mode == 0:
    target_id = min(target_labeled[:,0])
    for x, (id, x1, y1, x2, y2, score)
                    in enumerate(target_labeled):
        if id == target_id:
            return(id, x1, y1, x2, y2, score)
```

- **Modo 1** O Modo 1 irá definir como alvo o último objeto da classe a entrar na visão, ou seja, o com o maior ID. Vale a ressalva de que, caso haja uma troca de ID de um objeto já dentro da visão, ele trocará para esse objeto existente previamente mas que teve seu número de identificação renovado.

```
elif mode == 1:
    target_id = max(target_labeled[:,0])
    for x, (id, x1, y1, x2, y2, score) in
                    enumerate(target_labeled):
        if id == target_id:
            return(id, x1, y1, x2, y2, score)
```

```
        return 0
```

- **Modo 2** O Modo 2 tornará o objeto com a maior probabilidade associada de ser da classe de interesse como alvo.

```
    elif mode == 2:
        max_labeled_score = max(target_labeled[:,5])
        for x, (id, x1, y1, x2, y2, score)
                                        in enumerate(target_labeled):
            target_id = id if float(score) ==
                                    float(max_labeled_score) else None
            return(id, x1, y1, x2, y2, score)
```

3.4.3.2 **Coordenadas extraídas**

Após a definição do objeto alvo, a saída da função serão as coordenadas e probabilidade da inferência, e, a partir dessas coordenadas separadas das demais, o robô terá um meio de se localizar em relação ao alvo.

A forma proposta para realizar a tarefa de seguir o objeto de acordo com os critérios definidos é, a partir das coordenadas alvo e do alinhamento da frente do robô com a câmera, realizar a subtração das coordenadas de centro da caixa delimitadora do elemento de interesse das coordenadas de centro da imagem. Dessa forma, será possível saber o quanto será necessário ajustar a trajetória para que consiga chegar até o objetivo.

[**Centro Horizontal do Objeto**] = [Coordenada X menor + Coordenada X maior] / 2

[**Centro Vertical do Objeto**] = [Coordenada Y menor + Coordenada Y maior] / 2

[**Ajuste Horizontal da Trajetoria**] = ([Dimensão Horizontal da Imagem] / 2) - [Centro Horizontal do Objeto]

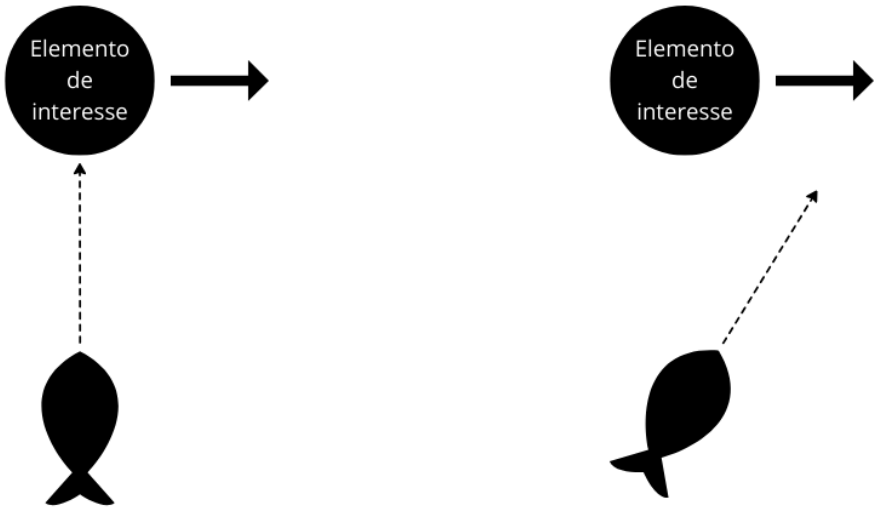
[**Ajuste Vertical da Trajetoria**] = ([Dimensão Vertical da Imagem] / 2) - [Centro Vertical do Objeto]

3.4.4 **Predição de trajetória**

Uma funcionalidade otimizadora do movimento do dispositivo consiste em não levar em consideração apenas a posição do objeto, como também sua velocidade e, então, adaptar o rumo para mirar na posição futura prevista. Uma das alternativas para cumprir esse propósito é um dos algoritmos que compõem o SORT, o filtro Kalman, entretanto, são necessárias adaptações para utilizá-lo sobre um alvo específico de forma a resultar na previsão em um intervalo desejado.

Figura 19 – Ajuste de trajetória através das coordenadas de centro e do objeto

Fonte: Autor



(a) Algoritmo que leva em consideração a posição atual

(b) Algoritmo que utiliza as coordenadas previstas

Figura 20 – Uso do filtro Kalman para otimização do ajuste de trajetória e perseguição do objeto alvo em movimento.
Fonte: Autor

3.4.4.1 **Filtro Kalman**

De forma semelhante ao algoritmo do SORT, para utilizar o filtro Kalman e realizar predições de trajetória também é necessário fornecer uma lista de coordenadas. Entretanto, neste caso os números fornecidos devem pertencer apenas ao objeto de interesse determinado anteriormente.

Cada conjunto de posições informadas ao filtro devem estar temporalmente espaçadas de acordo com o quão distante do quadro presente seja a próxima previsão. Em outras palavras, caso nosso interesse seja saber a estimativa de posição do objeto daqui a 5 segundos, as coordenadas da lista devem ser coletadas e enviadas seguindo esse intervalo de tempo.

3.4.4.2 **Cálculo de FPS**

A biblioteca do OpenCV não possui, de forma nativa, uma forma de verificar os quadros por segundo de um vídeo, o que torna então necessária a implementação dessa função a partir da análise da velocidade de execução do algoritmo.

Isso é feito através da criação de marcos temporais no início e fim da execução, e, com a informação de quanto tempo um laço demora para ser concluído, é possível calcular quantos quadros por segundo o vídeo está sendo executado.

```
# Calculo do FPS, trecho executado
# a cada inicio do loop do algoritmo
# Variavel para calcular o FPS
FPS_frame_count = 0
# Quantidade de frames a serem contabilizados no calculo
FPS_frame_amount = 5
FPS = 0

if FPS_frame_count == 0:
    FPS_start = time.time()
    FPS_frame_count = 0
    FPS_frame_count += 1
elif FPS_frame_count < FPS_frame_amount:
    FPS_frame_count += 1
else:
    FPS_end = time.time()
    FPS = FPS_frame_amount // (FPS_end - FPS_start)
    FPS_frame_count = 0
```

A partir do FPS do vídeo, temos uma forma de associar o número de quadros, ou seja, de vezes que o laço de repetição foi executado, com o intervalo de tempo decorrido. Dessa forma, o algoritmo saberá em que momentos fornecer as coordenadas ao filtro Kalman para que a previsão seja feita no espaçamento temporal desejado.

```
# Variavel que controla o inicio do Filtro Kalman
Kalman_start = False
# Intervalo de tempo entre os registros e as previsoes
Kalman_time = 0.5
# Variavel que conta os frames dentro desse intervalo de tempo
Kalman_frame_count = 0
# Quantidade de frames correspondente ao intervalo
# de tempo entre as previsoes
# do Filtro Kalman
Kalman_frame_amount = int(FPS*Kalman_time)

if Kalman_frame_count >= Kalman_frame_amount:
    if Kalman_start is False:
        Kalman_start = True
        Kalman_Filter = KalmanBoxTracker([target_register[0,0],
                        target_register[0,1],
                        target_register[0,2],
                        target_register[0,3]])
        target_register = np.empty((0,4))
        Kalman_frame_count = 0
    else:
        Kalman_Filter.update([target_register[0,0],
                              target_register[0,1],
                              target_register[0,2],
                              target_register[0,3]])
        print(Kalman_Filter.predict())
        target_register = np.empty((0,4))
        Kalman_frame_count = 0
else:
    Kalman_frame_count += 1
```

3.4.4.3 **Considerações**

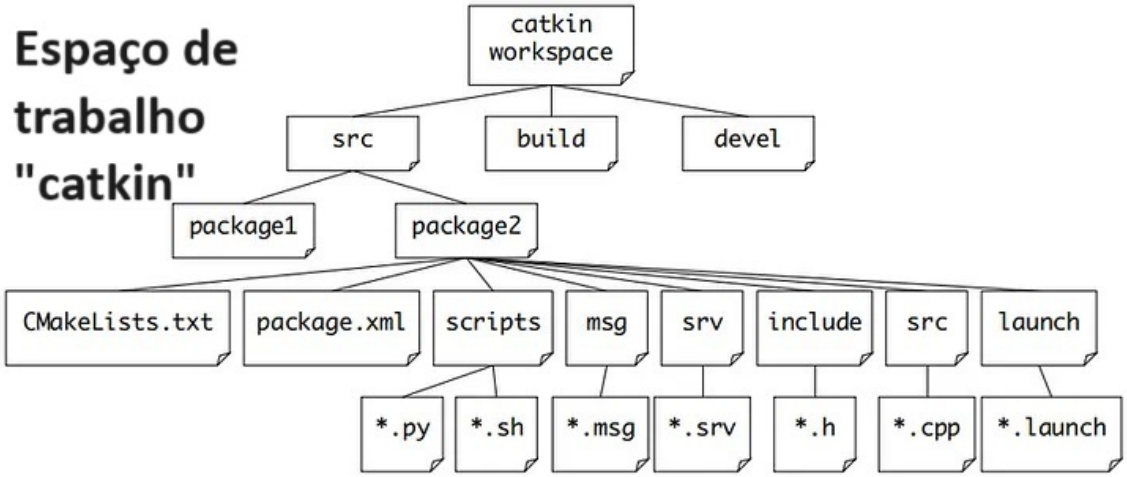
Este método para prever e se adequar ao movimento aparente do objeto deve levar em consideração o movimento relativo do objeto em relação à câmera, pois, caso o objeto esteja parado e o robô ajuste sua trajetória para que ele se encontre no centro, é possível que acabe sendo percebido um movimento relativo mesmo que o elemento esteja em repouso.

3.5 Implementação do ROS

A plataforma de desenvolvimento do ROS possui diversas funcionalidades, como simulação, modularização de funcionalidades e integração de algoritmos. Neste projeto, a funcionalidade de maior interesse é a comunicação e troca de informações entre os diferentes códigos.

Um dos aspectos que torna os projetos de ROS altamente intercambiáveis é que há uma organização de pastas e arquivos padronizados. Há diversos benefícios em utilizar o ambiente ROS ao invés de colocar todas as funcionalidades dentro de um mesmo arquivo, como, por exemplo, maior facilidade na hora de editar funções e mesmo integrar novas ao sistema.

Figura 21 – Organização de um workspace (espaço de trabalho) ROS

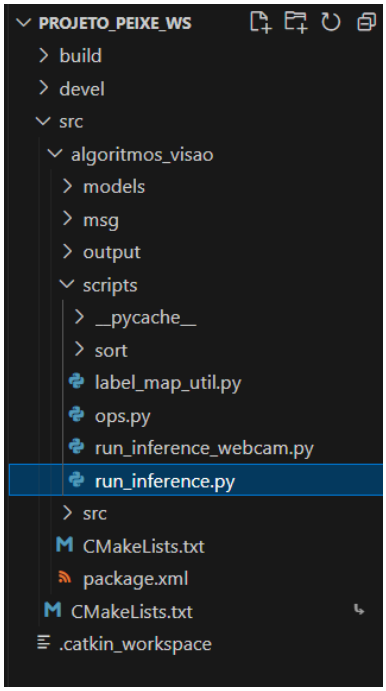


Modificado de (ANWAR, 2021)

3.5.1 **Publisher Node**

Para que o algoritmo de visão atue em conjunto com outros códigos ele deverá disponibilizar os dados de interesse, as coordenadas, para que outros executáveis sejam capazes de visualizá-los. Isso é possível ser feito implementando a funcionalidade de publicação em um tópico. O código feito, dentro de um ambiente ROS, é chamado de node, e se torna um publisher node que envia mensagens, estruturas de dados organizados de forma específica, a um tópico.

Figura 22 – Pastas do projeto



Fonte: Autor

Outros códigos são capazes de se inscrever e ler mensagens publicadas no tópico, sendo então chamados de subscriber node.

```
def publish_object_data(target_class, x, y, x_sort, y_sort):
    # Cria objeto publisher
    #(Nome do topico, tipo da mensagem, quantas
    mensagens da para adicionar na fila)
    pub_coord = rospy.Publisher('obj_coordinates',
                                Obj_coordinates, queue_size=10)
    # Inicia o node
    #(Nome do node, anonymous - cria um nome caso
    # haja mais de um node publicando)
    rospy.init_node('coord_publisher', anonymous=True)
    # Unidade em Hz
    rate = rospy.Rate(1)
    # Aviso que o node foi iniciado
    rospy.loginfo("Publisher iniciado
    #e publicando coordenadas do alvo")
    while not rospy.is_shutdown():
        msg_obj_coord = Obj_coordinates()
        # Insira as informacoes
```

```
        # criadas no Obj_coordinates.msg
        msg_obj_coord.target_class = target_class
        msg_obj_coord.x = x
        msg_obj_coord.y = y
        msg_obj_coord.x_predicted = x_sort
        msg_obj_coord.y_predicted = y_sort
        pub_coord.publish(msg_obj_coord)
        rate.sleep()
```

3.5.2 **Message**

A mensagem, message, contém a informação de quais tipos de dado e sequência que devem ser enviados ao tópico. Para as coordenadas, é interessante que sejam no formato float, para caso se deseje trabalhar com coordenadas normalizadas, ou int, no caso das coordenadas baseadas nas dimensões da imagem da câmera em pixels.

```
string target_class
float32 x
float32 y
float32 x_predicted
float32 y_predicted
```

3.5.3 **Topic**

O tópico é onde as mensagens com a formatação específica são publicados. No projeto da tese, é interessante que o tópico contenha as coordenadas do objeto alvo.

3.5.4 **Subscriber**

O node que se torna subscriber lê informações publicadas no tópico. Uma possível abordagem é fazer com que o algoritmo responsável pela dinâmica do robô seja inscrito no tópico em que as coordenadas são publicadas para realizar o controle dos movimentos.

4 RESULTADOS

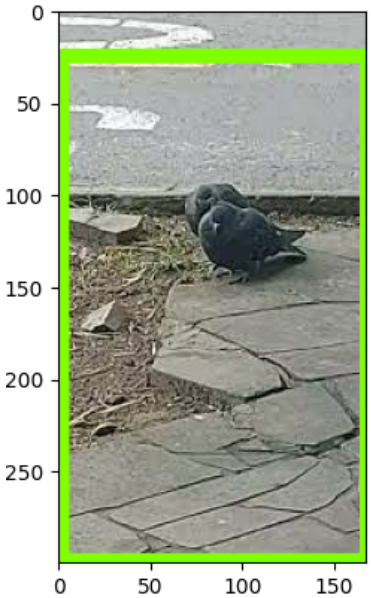
4.1 Treinamento do modelo

Foi feito um primeiro processo de treinamento do modelo, inicialmente, ao longo de nove horas, no entanto, os resultados obtidos não foram satisfatórios, como pode ser observado na Figura 23.

Diante disso, um segundo teste foi conduzido e o modelo foi treinado durante cinco dias de treino, resultando em um desempenho mais eficiente que pode ser verificado na Figura 24 contendo múltiplos pombos. É importante notar que devido ao treinamento realizado especificamente com pombos cinzas o modelo não foi capaz de detectar pombos brancos.

Esse aspecto evidencia a necessidade de treinar redes personalizadas para objetos de interesse específicos. Assim, se o objetivo é ter um modelo que siga objetos não inclusos nos modelos pré-treinados, é possível treinar novas redes neurais com esses objetos e ajustá-las às necessidades específicas. Nesse contexto, o treinamento de um modelo personalizado se mostrou uma solução satisfatória para aumentar a versatilidade do algoritmo desenvolvido.

Figura 23 – Resultado após nove horas de treino

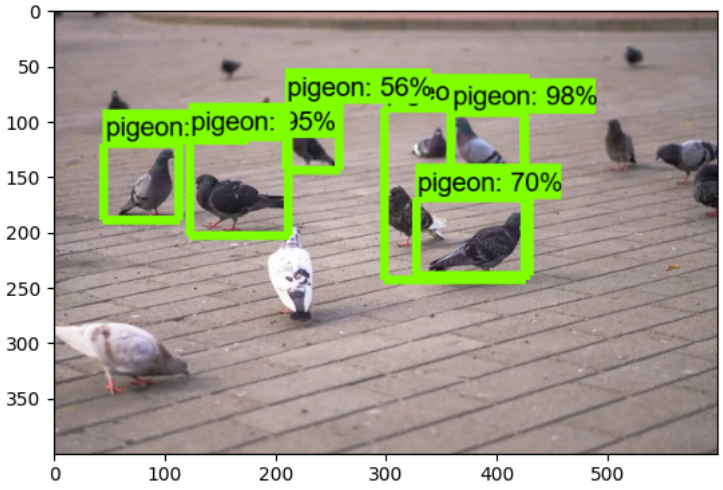


Fonte: Autor

4.2 Detecção e Rastreamento de Objetos

A rede pré-treinada utilizada, SSD MobileNet v2 320x320, empregada durante o desenvolvimento, apresentava oscilações significativas nas dimensões das caixas delimitadoras. Essa instabilidade é associada à velocidade do modelo que é priorizada so-

Figura 24 – Resultado após cinco dias de treino



Fonte: Autor

bre a precisão e acurácia. Contudo, ao testar o código com um modelo mais robusto, faster_rcnn_resnet50_v1_640x640_coco17_tpu-8, observou-se uma melhoria considerável na consistência das caixas delimitadoras, que passaram a ter uma maior estabilidade e delimitações mais precisas.

Ainda assim, a integração do algoritmo SORT ao modelo proporcionou resultados satisfatórios no rastreamento de objetos. Especificamente, o modo 2 do algoritmo demonstrou ser apropriado para uso em cenários com apenas um objeto da classe, minimizando confusões temporárias com elementos do ambiente, geralmente associados a probabilidades de correspondência baixas, garantindo que o objeto real permaneça como foco do rastreamento.

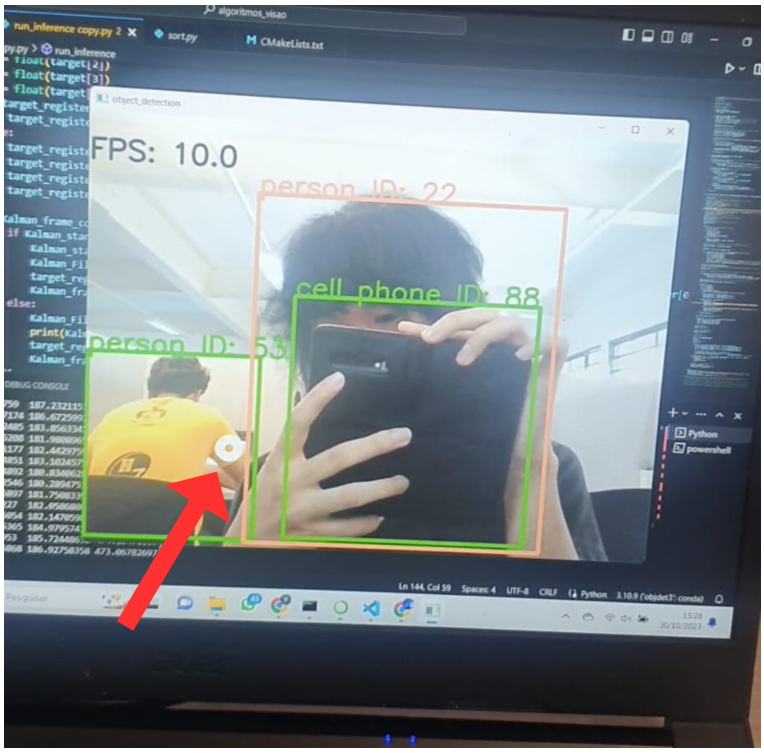
Os modos 0 e 1 também forneceram bons resultados e se mostraram adequados para quando há mais de um elemento da classe de interesse em cena, sendo a priorização do último elemento a aparecer mais adequada devido à natureza do SORT de frequentemente trocar IDs. Além disso, nesse cenário, eles são mais adequados do que o modo 2, visto que as probabilidades associadas tendem a variar consideravelmente ao longo das inferências devido a variações no ângulo de observação e iluminação.

O filtro de Kalman aplicado para a predição de trajetória demonstrou sucesso na tarefa de acompanhar o alvo durante os testes realizados na webcam. No entanto, é importante salientar que ainda não foi avaliado o desempenho das predições e o comportamento do movimento diante de deslocamentos relativos da câmera, que ocorrerão quando ela estiver acoplada ao dispositivo robótico. Essa condição pode levar à percepção de uma velocidade do objeto que não condiz com a realidade, sendo um aspecto crucial a ser investigado para uma compreensão completa do desempenho do filtro em cenários dinâmicos.

É possível elaborar algumas propostas de solução de antemão, como, por exemplo, alternar para as coordenadas previstas pelo filtro Kalman apenas em momentos específicos,

como apenas durante o deslocamento para frente do robô, ou seja, quando ele estiver em linha reta com o alvo alinhado à câmera.

Figura 25 – Imagem da inferência com múltiplos objetos



A seta vermelha aponta para a coordenada gerada pelo filtro Kalman. Note que o alvo selecionado é, conforme configurado, o com o maior ID, que corresponde ao último a entrar na cena.
Fonte: Autor

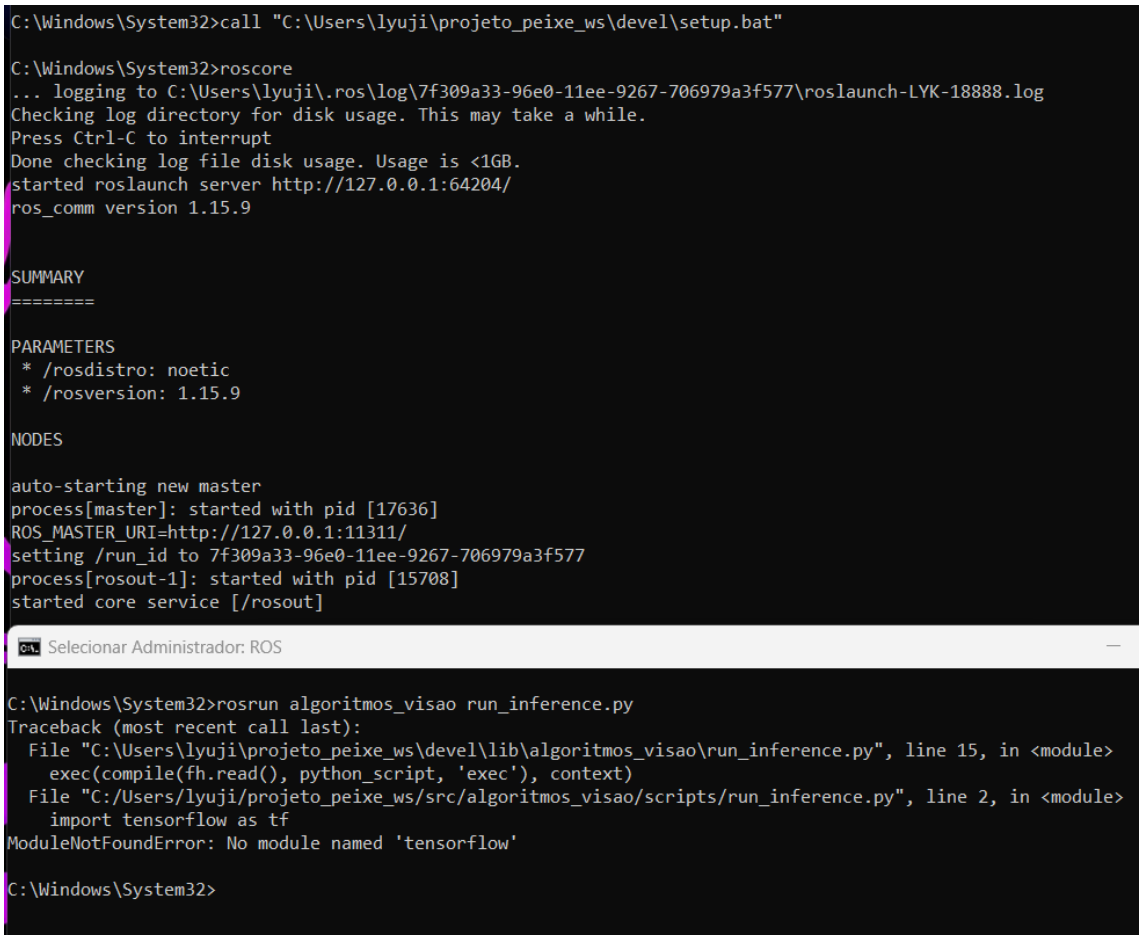
4.3 ROS

A utilização do ROS no ambiente Windows apresenta algumas limitações uma vez que o ROS é predominantemente desenvolvido para ambientes baseados em Linux. Isso implica que ajustes para garantir compatibilidade são esperados e, apesar de a máquina Windows ter conseguido iniciar o ambiente ROS, construir as dependências do projeto e executar com sucesso os algoritmos de publicação e inscrição, enfrentou falhas ao executar o código de visão como node pois foi incapaz de localizar o módulo do TensorFlow. Esse problema pode estar associado à execução do ROS no Windows ou a um erro na seleção do interpretador, aspectos que demandam investigação adicional para resolução.

4.4 Uso da Raspberry

Para abordar a hipótese de que os problemas estavam relacionados ao ambiente Windows foi realizada a configuração da Raspberry Pi com o sistema operacional Raspberry

Figura 26 – Módulo TensorFlow não disponível ao executar o ROS na máquina local Windows 11



Fonte: Autor

Pi OS baseado em Linux. Foi instalada a versão do Conda para Raspberry, visto que o Raspberry Pi OS possui uma versão nativa do Python (3.11) e sua modificação junto a outras bibliotecas e funcionalidades do sistema não é viável.

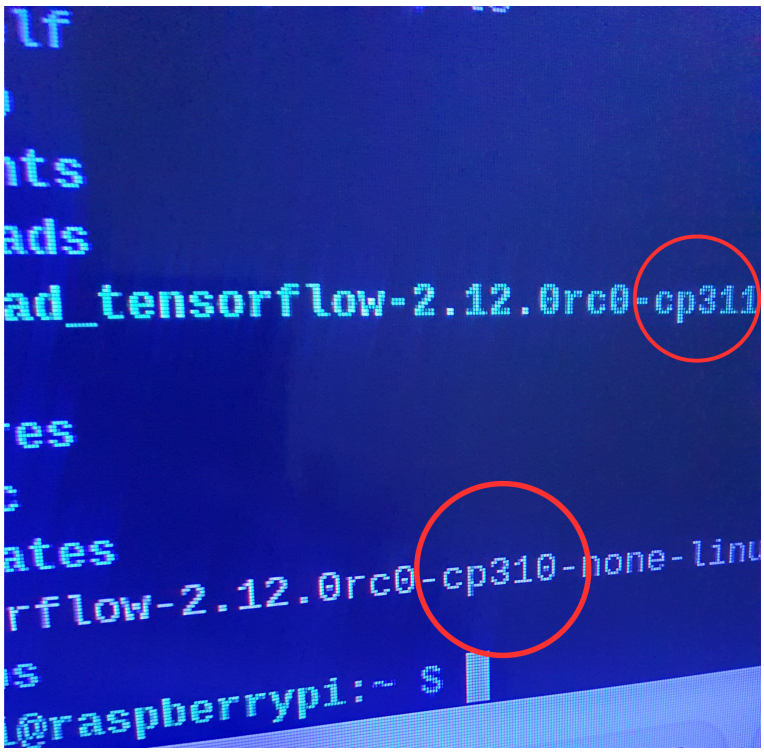
Após a criação do ambiente virtual a próxima etapa envolveu testar o algoritmo fora do ambiente ROS para identificar e corrigir possíveis problemas de versões. Houve a necessidade de adaptar o código, uma vez que a câmera oficial não é uma câmera USB, impossibilitando o uso do comando OpenCV utilizado na máquina Windows.

A solução adotada envolveu a utilização da biblioteca picamera2, que, por sua vez, faz uso da **libcamera**. No entanto, devido a um bug já constatado pela comunidade desenvolvedora (KASPERROR, 2022), a libcamera não pode ser utilizada a não ser no Python nativo.

Diante dessa limitação, a opção foi reescrever o código fora do ambiente virtual e baixar as dependências de forma nativa, ou seja, diretamente no sistema. Entretanto, na instalação, o procedimento se deparou com um bug da distribuição de TensorFlow para aquela versão específica de Python (3.11), onde, ao requisitar ao servidor, era fornecida a versão errada e

não foi possível instalar a versão correta para o sistema.

Figura 27 – Erro da requisição do servidor



Os círculos vermelhos apontam para a versão que foi requisitada ao servidor e a versão recebida após a requisição.
Fonte: Autor

5 CONCLUSÃO

Neste trabalho foi desenvolvido um algoritmo de visão que apresenta as funcionalidades necessárias para navegação, incluindo a capacidade de selecionar um alvo de uma classe específica seguindo critérios configuráveis. Além disso, realizamos testes bem-sucedidos para explorar a viabilidade de implementar redes neurais personalizadas para adição de novos objetos, destacando a flexibilidade e adaptabilidade do código.

A versatilidade do algoritmo é evidenciada pela possibilidade de configurar a classe alvo, o modo de seleção em casos de múltiplos objetos da mesma categoria, a capacidade de seleção de diferentes modelo de inferência e a inclusão de novos objetos, aspectos cruciais para a aplicabilidade prática do algoritmo em diferentes contextos e cenários.

No entanto, é importante salientar que, durante a implementação, foram encontradas dificuldades que requerem solução para garantir o bom funcionamento do sistema no ambiente para o qual foi projetado. Apesar da lógica e desenvolvimento bem sucedido da programação, são necessárias correções no que diz respeito às plataformas de implementação do dispositivo.

Atualmente, é possível utilizar o algoritmo fora do ambiente ROS, integrando a funcionalidade de controle de movimento e outros algoritmos em um único arquivo. Embora essa prática permita uma execução mais simplificada, ela dificulta a integração de novas funções e a edição das existentes.

Portanto, próximos passos incluem a resolução dos conflitos de dependência, teste de viabilidade de diferentes modelos de câmera para contornar o problema de incompatibilidade de bibliotecas e integração do código na plataforma ROS.

REFERÊNCIAS

ABADI, M. et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. Software available from tensorflow.org. Disponível em: https://www.tensorflow.org/.

ANACONDA Software Distribution. Anaconda Inc., 2020. Disponível em: https://docs.anaconda.com/.

ANSARI, S. *Building Computer Vision Applications Using Artificial Neural Networks*: With step-by-step examples in opencv and tensorflow with python. [S.l.]: Apress, 2020.

ANWAR, A. Create and build your first ros package. 2021. Disponível em: https://medium.com/swlh/7-simple-steps-to-create-and-build-our-first-ros-package-7e3080d36faa. Acesso em: 28 nov. 2023.

BEWLEY, A. et al. Simple online and realtime tracking. In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2016. Disponível em: http://dx.doi.org/10.1109/ICIP.2016.7533003.

CONVOLUTIONAL Neural Networks for Visual Recognition. Disponível em: https://cs231n.github.io/convolutional-networks/. Acesso em: 04 dez. 2023.

GONZALEZ, R. C. *Digital Image Processing*: Third edition. [S.l.]: Pearson Prentice Hall, 2008.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org.

KASPERROR. *[BUG] Cannot be used with non-system python*. [S.l.]: GitHub, 2022. https://github.com/raspberrypi/picamera2/issues/446. Acesso em: 09 dez. 2023.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012. v. 25. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

LIN, T. et al. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. Disponível em: http://arxiv.org/abs/1405.0312.

RASPBERRY PI FOUNDATION. *Raspberry Pi 4 Model B*. [S.l.]. Disponível em: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf?_gl=1*7gnzm1*_ga*MTQ1NzU1NjU4OS4xNzAyMDkzOTEy*_ga_22FD70LWDS*MTcwMjA5MzkxMy4xLjEuMTcwMjA5NDAwMC4wLjAuMA.. Acesso em: 09 dez. 2023.

ROSEBROCK, A. *Intersection over Union (IoU) for object detection*. https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. Acesso em: 22 nov. 2023.

TENSORFLOW: What's coming in TensorFlow 2.0. https://blog.tensorflow.org/2019/01/whats-coming-in-tensorflow-2-0.html. Acesso em: 02 dez. 2023.

TRAN, D. *xmltocsv*. [S.l.]: GitHub, 2017. https://github.com/datitran/raccoon_dataset/blob/master/xml_to_csv.py.

TRAN, D. *generatetfrecord*. [S.l.]: GitHub, 2018. https://github.com/datitran/raccoon_dataset/blob/master/generate_tfrecord.py.

VALMADRE, J. et al. *Local Metrics for Multi-Object Tracking*. 2021.

VOULODIMOS, A. Deep learning for computer vision: A brief review. 2018. Disponível em: https://www.hindawi.com/journals/cin/2018/7068349/.

WOJKE, N.; BEWLEY, A.; PAULUS, D. *Simple Online and Realtime Tracking with a Deep Association Metric*. 2017.

WU, J. et al. Robust variational optical flow algorithm based on rolling guided filtering. In: *2018 11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*. [S.l.: s.n.], 2018. p. 1–6.

YAMASHINA, K. et al. Proposal of ros-compliant fpga component for low-power robotic systems. 08 2015.

ZHAO, M.; WANG, L.; HAN, J. An adaptive tracking window based on mean-shift target tracking algorithm. In: *2013 Chinese Automation Congress*. [S.l.: s.n.], 2013. p. 348–352.

APÊNDICE A - CÓDIGO DE INFERÊNCIA

Para utilizar o código, é necessário configurar de acordo com os caminhos das pastas e arquivos dentro da função **if __name__ == '__main__'**. Os códigos podem ser encontrados no repositório do autor. Para utilizar as funções importadas no algoritmo, baixe o repositório oficial do TensorFlow.

```
import numpy as np
import tensorflow as tf
import cv2
import sys
import random
import time
sys.path.append('\\projeto_peixev4\\sort')
from object_detection.utils import ops as utils_ops
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_util
from sort import *

# patch tf1 into `utils.ops`
utils_ops.tf = tf.compat.v1
# Patch the location of gfile
tf.gfile = tf.io.gfile
def load_model(model_path):
    model = tf.saved_model.load(model_path)
    return model

def inference_for_frame(model, image):
    image = np.asarray(image)
    # The input needs to be a tensor,
    # convert it using `tf.convert_to_tensor`.
    input_tensor = tf.convert_to_tensor(image)
    # The model expects a batch of images,
    # so add an axis with `tf.newaxis`.
    input_tensor = input_tensor[tf.newaxis,...]

    # Run inference
    output_dict = model(input_tensor)

    # All outputs are batches tensors.
```

```
    # Convert to numpy arrays, and take index [0] to
    # remove the batch dimension.
    # We're only interested in the first num_detections.
    num_detections = int(output_dict.pop('num_detections'))
    output_dict = {key: value[0, :num_detections].numpy()
                   for key, value in output_dict.items()}
    output_dict['num_detections'] = num_detections

    # detection_classes should be ints.
    output_dict['detection_classes'] =
        output_dict['detection_classes'].astype(np.int64)

    # Handle models with masks:
    if 'detection_masks' in output_dict:
        # Reframe the the bbox mask to the image size.
        detection_masks_reframed =
            utils_ops.reframe_box_masks_to_image_masks(
            output_dict['detection_masks'],
            output_dict['detection_boxes'],
            image.shape[0], image.shape[1])
        detection_masks_reframed =
            tf.cast(detection_masks_reframed > 0.5, tf.uint8)
        output_dict['detection_masks_reframed'] =
            detection_masks_reframed.numpy()
    return output_dict

def target_tracking(tracked_objects_list, target, mode=0):
    '''tracked_object_list: array de shape (n,7),
    contendo as informacoes no formato
    (x1, y1, x2, y2, score, label, id)
    target: string com o label do objeto alvo
    mode: 0 -> segue o primeiro objeto da classe alvo que aparecer
    mode: 1 -> segue o ultimo objeto da classe alvo que aparecer
    mode: 2 -> segue o objeto da classe alvo com o maior score
    em cena'''
    target_labeled = np.empty((0,6))

    # Checagem de se a lista dos objetos da cena com categoria
    # alvo tem pelo menos um objeto
```

```
    if tracked_objects_list.shape[0] != 0:
    # Loop armazena os objetos da cena com a categoria alvo
        for x, (x1, y1, x2, y2, score, label, id)
            in enumerate(tracked_objects_list):
            if label == target:
                target_labeled = np.append(target_labeled,
                    [[id, x1, y1, x2, y2, score]], axis=0)

    # Checagem de se existe ao menos um objeto na lista de
    # objetos da categoria alvo
    if target_labeled.shape[0] != 0:
    # Loop gera o output de acordo com o modo (mode)
        if mode == 0:
            target_id = min(target_labeled[:,0])
            for x, (id, x1, y1, x2, y2, score)
                in enumerate(target_labeled):
                    if id == target_id:
                        return(id, x1, y1, x2, y2, score)
        elif mode == 1:
            target_id = max(target_labeled[:,0])
            for x, (id, x1, y1, x2, y2, score)
                in enumerate(target_labeled):
                if id == target_id:
                    return(id, x1, y1, x2, y2, score)
            return 0
        elif mode == 2:
            max_labeled_score = max(target_labeled[:,5])
            for x, (id, x1, y1, x2, y2, score)
                in enumerate(target_labeled):
                    target_id = id if float(score) ==
                        float(max_labeled_score) else None
                    return(id, x1, y1, x2, y2, score)
    else:
        return None

def run_inference(model, category_index, cap):
    # Limiar para que o objeto seja classificado como valido
    threshold = 0.5
    # Lista que contera as coordenadas dos objetos detectados
```

```
    # e suas pontuacoes
    # no formato [[x1,y1,x2,y2,score],[x1,y1,x2,y2,score],...]
    detections = np.empty((0, 5))
    # Lista contendo as coordenadas, os scores e as categorias
    detections_plus_labels = np.empty((0, 6))
    # Lista que contera as coordenadas, o score,
    # a categoria e o ID
    tracked_objects = np.empty((0, 7))

    # Instancia de SORT
    mot_tracker = Sort()

    # Cores para as bounding boxes
    colors = [(random.randint(0, 255), random.randint(0, 255),
        random.randint(0, 255)) for j in range(10)]

    # Armazenamento do historico coordenadas de centro
    target_register = np.empty((0,4))

    # Variavel que controla o inicio do Filtro Kalman
    Kalman_start = False
    # Intervalo de tempo entre os registros e as previsoes
    Kalman_time = 0.5
    # Variavel que conta os frames dentro desse intervalo de tempo
    Kalman_frame_count = 0

    # Variavel para calcular o FPS
    FPS_frame_count = 0
    # Quantidade de frames a serem contabilizados no calculo
    FPS_frame_amount = 5
    FPS = 0

    while True:
        # ret retorna falso caso nao haja nada no frame
        ret, frame = cap.read()
        # Calculo do FPS
        if FPS_frame_count == 0:
            FPS_start = time.time()
            FPS_frame_count = 0
```

```
        FPS_frame_count += 1
    elif FPS_frame_count < FPS_frame_amount:
        FPS_frame_count += 1
    else:
        FPS_end = time.time()
        FPS = FPS_frame_amount // (FPS_end - FPS_start)
        FPS_frame_count = 0
        print('FPS: ', FPS)

    # Quantidade de frames correspondente ao
    intervalo de tempo
    # entre as previsoes do Filtro Kalman
    Kalman_frame_amount = int(FPS*Kalman_time)

    # Actual detection
    output_dict = inference_for_frame(model, frame)

    for x, (y_min, x_min, y_max, x_max)
        in enumerate(output_dict['detection_boxes']):
            if output_dict['detection_scores'][x] > threshold:
                x1 = int(x_min*frame.shape[1])
                y1 = int(y_min*frame.shape[0])
                x2 = int(x_max*frame.shape[1])
                y2 = int(y_max*frame.shape[0])
                # O array abaixo armazena os objetos em cena
                detections = np.append(detections,
                    [x1, y1, x2, y2,
                    output_dict['detection_scores'][x]]]
                    , axis=0)
                # O array abaixo armazena os objetos em cena
                # e suas categorias para comparar
                # posteriormente com os IDs
                detections_plus_labels =
                    np.append(detections_plus_labels,
                    [[x1, y1, x2, y2,
                    output_dict['detection_scores'][x],
                    category_index[output_dict
                    ['detection_classes'][x]]
                    ['name']]], axis=0)
```

```
# Envia a lista ao algoritmo de tracking
track_bbs_ids = mot_tracker.update(detections)
# Diferenca maxima entre a distancia entre as coordenadas
# das bounding boxes, visto que o output do SORT
# difere em algumas unidades
delta_bbox = 10
# Compara as coordenadas da lista de deteccoes com as
# coordenadas na lista dos IDs para fazer as associacoes

for track, (tx1, ty1, tx2, ty2, id)
    in enumerate(track_bbs_ids):
        for detection, (dx1, dy1, dx2, dy2, score, label)
            in enumerate(detections_plus_labels):
                if abs(int(tx1) - int(dx1)) <=
                delta_bbox and abs(int(ty1) - int(dy1))
                <= delta_bbox and abs(int(tx2) - int(dx2))
                <= delta_bbox and abs(int(ty2) - int(dy2))
                <= delta_bbox:
                    tracked_objects =
                        np.append(tracked_objects,
                        [[tx1, ty1, tx2, ty2, score,
                        label, int(id)]], axis=0)

# Envia a lista de objetos sob o algoritmo de tracking
# para o algoritmo que seleciona um alvo
target = target_tracking(tracked_objects, 'person', 2)

# Armazenamento
if target is not None:
    x1 = float(target[1])
    y1 = float(target[2])
    x2 = float(target[3])
    y2 = float(target[4])
    if target_register.shape[0] < 1:
        target_register = np.append(target_register,
            [[x1, y1, x2, y2]], axis=0)
    else:
        target_register[0,0] =
```

```
                (target_register[0,1] + x1) / 2
            target_register[0,1] =
                (target_register[0,1] + y1) / 2
            target_register[0,2] =
                (target_register[0,2] + x2) / 2
            target_register[0,3] =
                (target_register[0,3] + y2) / 2

        if Kalman_frame_count >= Kalman_frame_amount:
            if Kalman_start is False:
                Kalman_start = True
                Kalman_Filter = KalmanBoxTracker(
                    [target_register[0,0],
                    target_register[0,1],
                    target_register[0,2],
                    target_register[0,3]])
                target_register = np.empty((0,4))
                Kalman_frame_count = 0
            else:
                Kalman_Filter.update([target_register[0,0],
                    target_register[0,1],
                    target_register[0,2],
                    target_register[0,3]])
                print(Kalman_Filter.predict())
                target_register = np.empty((0,4))
                Kalman_frame_count = 0
        else:
            Kalman_frame_count += 1

    #print(target_center_register)
    # Coordenadas geradas pelo Filtro de Kalman
    xk1, yk1, xk2, yk2 = 0, 0, 0, 0

    if Kalman_start is True and not
        np.isnan(Kalman_Filter.predict()[0,0]):
            xk1 = int(Kalman_Filter.predict()[0,0])
            yk1 = int(Kalman_Filter.predict()[0,1])
            xk2 = int(Kalman_Filter.predict()[0,2])
            yk2 = int(Kalman_Filter.predict()[0,3])
```

```
# Centro da previsao de Kalman
xkcenter = (xk1 + xk2) // 2
ykcenter = (yk1 + yk2) // 2

# Visualizacao
# Texto sobre os objetos
# Fonte
fonte = cv2.FONT_HERSHEY_SIMPLEX
# Tamanho
fontScale = 1
# Line thickness of 2 px
thickness = 2

for object, (x1, y1, x2, y2, score, label, id)
    in enumerate(tracked_objects):
        x1 = 0 if tracked_objects.shape[0] == 0
            else int(float(tracked_objects[object,0]))
        y1 = 0 if tracked_objects.shape[0] == 0
            else int(float(tracked_objects[object,1]))
        x2 = 0 if tracked_objects.shape[0] == 0
            else int(float(tracked_objects[object,2]))
        y2 = 0 if tracked_objects.shape[0] == 0
            else int(float(tracked_objects[object,3]))

    # Centro do objeto
    xcenter = (x1 + x2) // 2
    ycenter = (y1 + y2) // 2

    #cv2.circle(frame, (xc1, yc1), 20, (0,0,255), 10, -1)
    #cv2.circle(frame, (xc2, yc2), 20, (0,0,255), 10, -1)
    cv2.circle(frame, (xkcenter, ykcenter), 10,
        (255,255,255), 10, -1)
    cv2.rectangle(frame, (x1, y1), (x2, y2),
        (colors[int(float(id)) % len(colors)]), 3)
    cv2.putText(frame, (tracked_objects[object,5] +
        ' ID: ' + tracked_objects[object,6]), (x1, y1),
        fonte, fontScale, (colors[int(float(id))
            % len(colors)]), thickness, cv2.LINE_AA)
```

```
            cv2.putText(frame, ('FPS: ' + str(FPS)), (0, 50), fonte,
                fontScale, (0,0,0), thickness, cv2.LINE_AA)
            cv2.imshow('object_detection',
                cv2.resize(frame, (800, 600)))
            if cv2.waitKey(25) & 0xFF == ord('q'):
                cap.release()
                cv2.destroyAllWindows()
                break

            # print(frame.shape) -> (480, 640, 3)

            '''vis_util.visualize_boxes_and_labels_on_image_array(
                frame,
                output_dict['detection_boxes'],
                output_dict['detection_classes'],
                output_dict['detection_scores'],
                category_index,
                instance_masks=output_dict.get(
                    'detection_masks_reframed', None),
                use_normalized_coordinates=True,
                line_thickness=8)'''

            # Esvazia as listas para o proximo frame
            detections = np.empty((0, 5))
            detections_plus_labels = np.empty((0, 6))
            tracked_objects = np.empty((0, 7))

    if __name__ == '__main__':
        # Caminho ate o modelo
        detection_model = load_model(
            'ssd_mobilenet_v2_320x320_coco17_tpu-8\\saved_model')
        # Caminho ate o label map
        category_index =
        label_map_util.create_category_index_from_labelmap('models\\
    ________research\\object_detection\\data\\mscoco_label_map.pbtxt',
            use_display_name=True)
        cap = cv2.VideoCapture(1)
        run_inference(detection_model, category_index, cap)
```

APÊNDICE B - DOWNLOAD DE MODELOS

Para fazer download de um dos modelos do TensorFlow, acesse o repositório oficial, selecione um dos modelos, insira-o no código abaixo e execute.

```
import wget
model_link = "(http://download.tensorflow.org/models
␣␣␣␣␣␣␣␣␣␣␣␣/object_detection/tf2/20200711/link.tar.gz)"
wget.download(model_link)
import tarfile
tar = tarfile.open('arquivo.tar.gz')
tar.extractall('.')
tar.close()
```

Após isso, será baixado o modelo no diretório que contém o código acima, e será necessário fazer a referência da pasta **saved_model** no algoritmo de visão contido no Apêndice A.

APÊNDICE C - TREINAMENTO DE MODELO

Neste apêndice estão descritas as etapas referentes ao treinamento de um modelo para detectar objetos customizados, o que será útil no caso de categorias específicas não encontradas nos datasets disponíveis publicamente. Para utilizar as funções de treinamento, baixe orepositório oficial do TensorFlow.

Coletar imagens que contenham os objetos de interesse e separá-las em uma pasta **train**, para o treino do modelo, e **test**, para o teste. Coletar acima de 200 imagens e inserir 70% delas na pasta para treino e 30% delas na para teste.

Executar no terminal labelimg.exe e abrir as imagens para categorizar os objetos presentes nela através do Create RectBox, como na Figura 16.

Converter Anotações para CSV: Utilize o script abaixo para converter os arquivos de anotação XML para o formato CSV. Este script espera uma pasta "images"com subpastas "train"e "test". Baseado no código (TRAN, 2017)

```
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET

def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     member[0].text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
                     int(member[4][3].text)
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class',
        'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df
```

```
def main():
    for folder in ['train', 'test']:
        image_path = os.path.join(os.getcwd(),
            ('images/' + folder))
        xml_df = xml_to_csv(image_path)
        xml_df.to_csv(('images/'+folder+'_labels.csv'),
            index=None)
    print('Successfully converted xml to csv.')

main()
```

Gerar Arquivos TFRecord: Outro script gera arquivos TFRecord, um formato que o TensorFlow usa para treinamento. Modifique de acordo com os labels nomeados na etapa de categorização e execute este script tanto para os dados de treinamento quanto para os de teste. Baseado no código (TRAN, 2018)

```
from __future__ import division
from __future__ import print_function
from __future__ import absolute_import

import os
import io
import pandas as pd

from tensorflow.python.framework.versions import VERSION
if VERSION >= "2.0.0a0":
    import tensorflow.compat.v1 as tf
else:
    import tensorflow as tf

from PIL import Image
from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict

flags = tf.app.flags
flags.DEFINE_string('csv_input', '', 'Path to the CSV input')
flags.DEFINE_string('output_path', '', 'Path to output TFRecord')
flags.DEFINE_string('image_dir', '', 'Path to images')
FLAGS = flags.FLAGS
```

```
'''
****************************************
Make sure to edit this method to match
the labels you made with labelImg!
****************************************
'''
def class_text_to_int(row_label):
    if row_label == 'example_label_1':
        return 1
    elif row_label == 'example_label_2':
        return 2
    elif row_label == 'example_label_3':
        return 3
    else:
        return None

def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for
        filename, x in zip(gb.groups.keys(), gb.groups)]

def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path,
        '{}'.format(group.filename)), 'rb') as fid:
            encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size
    filename = group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
    classes = []
    for index, row in group.object.iterrows():
```

```
            xmins.append(row['xmin'] / width)
            xmaxs.append(row['xmax'] / width)
            ymins.append(row['ymin'] / height)
            ymaxs.append(row['ymax'] / height)
            classes_text.append(row['class'].encode('utf8'))
            classes.append(class_text_to_int(row['class']))
        tf_example = tf.train.Example(
            features=tf.train.Features(feature={
            'image/height':
                dataset_util.int64_feature(height),
            'image/width':
                dataset_util.int64_feature(width),
            'image/filename':
                dataset_util.bytes_feature(filename),
            'image/source_id':
                dataset_util.bytes_feature(filename),
            'image/encoded':
                dataset_util.bytes_feature(encoded_jpg),
            'image/format':
                dataset_util.bytes_feature(image_format),
            'image/object/bbox/xmin':
                dataset_util.float_list_feature(xmins),
            'image/object/bbox/xmax':
                dataset_util.float_list_feature(xmaxs),
            'image/object/bbox/ymin':
                dataset_util.float_list_feature(ymins),
            'image/object/bbox/ymax':
                dataset_util.float_list_feature(ymaxs),
            'image/object/class/text':
                dataset_util.bytes_list_feature(classes_text),
            'image/object/class/label':
                dataset_util.int64_list_feature(classes),
        }))
        return tf_example

    def main(_):
        writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
        path = os.path.join(FLAGS.image_dir)
        examples = pd.read_csv(FLAGS.csv_input)
```

```
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())

    writer.close()
    output_path = os.path.join(os.getcwd(), FLAGS.output_path)
    print('Successfully␣created␣the␣TFRecords:
␣␣␣␣␣␣␣␣{}'.format(output_path))


if __name__ == '__main__':
    tf.app.run()

# commands:
# python generate_tfrecord.py
    --csv_input=images/test_labels.csv
    --image_dir=images/test --output_path=test.record
# python generate_tfrecord.py
    --csv_input=images/train_labels.csv
    --image_dir=images/train --output_path=train.record
main()
```

Baixar um Modelo Pré-Treinado: Após a escolha de um modelo pré-treinado do TensorFlow Model Zoo. Escolha um modelo de acordo com suas necessidades e desempenho desejado. Baixar e descompactar conforme discorrido no Apêndice B.
Criar um Mapa de Rótulos: Crie um arquivo de mapa de rótulos (normalmente chamado labelmap.pbtxt). Cada classe deve ter um bloco no formato item id: ..., name: ... . Os IDs devem coincidir com os IDs usados nos scripts anteriores.

```
item{
id: 1
name: 'example_label_1'
}
item{
id: 2
name: 'example_label_2'
}
item{
```

```
  id: 3
  name: 'example_label_3'
}
```

Configurar o Arquivo de Configuração .config do Modelo: Copie o arquivo de configuração do modelo (normalmente encontrado em models/research/object_detection/configs/tf2/) para a pasta principal do projeto. Abra o arquivo de configuração em um editor de texto. Altere os seguintes parâmetros:

- **num_classes**: Número de classes no seu conjunto de dados.
- **fine_tune_checkpoint**: Caminho para o arquivo de checkpoint do modelo pré-treinado baixadp.
- **batch_size**: Defina um valor apropriado com base na capacidade do seu hardware. Caso use CPU, mantenha um valor baixo, como 2.
- **Configurar Caminhos para Dados**: Atualize os caminhos para os dados de treinamento e teste no arquivo de configuração. Certifique-se de que os caminhos estejam corretos, apontando para os arquivos TFRecord que você gerou.
  No **label_map_path** dentro do **train_input_reader**, insira o **.../labelmap.pbtxt** criado.
  No **fine_tune_checkpoint**, aponte para o .../checkpoint/ckpt-0 dentro do modelo que foi baixado na etapa anterior. Também altere o **fine_tune_checkpoint_type** de **"classification"** para **"detection"**.
  No **input_path** dentro do **train_input_reader**, insira o **.../train.record** criado.
  No **input_path** dentro do **eval_input_reader**, insira o **.../test.record** criado.

Treinamento do Modelo: Para iniciar o treinamento, utilize a função do TensorFlow. Use o script de treinamento fornecido no repositório oficial do TensorFlow no GitHub. Execute, no terminal, o comando:

```
python model_main_tf2.py --caminho_do.config
    --model_dir=training --alsologtostderr
```

O arquivo model_main_tf2.py se encontra no .../models/research/object_detection.
TensorBoard: É possível monitorar a progressão do modelo executando no terminal o comando

```
tensorboard --logdir 'training/train'
```

para abrir o TensorBoard
Exportar Modelo Treinado: Uma vez concluído o treinamento, exporte o modelo treinado para uso posterior através do comando no terminal

```
python exporter_main_v2.py --trained_checkpoint_dir=training
    --pipeline_config_path=arquivo.config
    --output_directory inference_graph
```

O script para exportação deve ser ajustado com os caminhos corretos.

APÊNDICE D - VERSÕES DE BIBLIOTECAS COMPATÍVEIS

Ao solucionar problemas de compatibilidade que ocorreram ao longo do desenvolvimento da tese, o ambiente virtual foi capaz de executar a inferência e treinamento a partir das versões listadas neste apêndice. Note que nem todas as bibliotecas foram utilizadas, visto que o Conda insere automaticamente na criação de espaços virtuais.

absl-py 1.4.0; apache-beam 2.46.0; array-record 0.4.0; astunparse 1.6.3; avro-python3 1.10.2; bleach 6.0.0; cachetools 5.3.1; certifi 2023.5.7; charset-normalizer 3.1.0; click 8.1.3; cloudpickle 2.2.1; colorama 0.4.6; contextlib2 21.6.0; contourpy 1.1.0; crcmod 1.7; cycler 0.11.0; Cython 0.29.35; dill 0.3.1.1; dm-tree 0.1.8; docopt 0.6.2; etils 1.3.0; fastavro 1.7.4; fasteners 0.18; filterpy 1.4.5; flatbuffers 1.12; fonttools 4.40.0; gast 0.4.0; gin-config 0.5.0; google-api-core 2.11.1; google-api-python-client 2.91.0; google-auth 2.21.0; google-auth-httplib2 0.1.0; google-auth-oauthlib 0.4.6; google-pasta 0.2.0; googleapis-common-protos 1.59.1; grpcio 1.56.0; h5py 3.9.0; hdfs 2.7.0; httplib2 0.21.0; idna 3.4; imageio 2.31.3; immutabledict 2.2.5; importlib-resources 5.12.0; joblib 1.3.1; kaggle 1.5.15; keras 2.9.0; Keras-Preprocessing 1.1.2; kiwisolver 1.4.4; labelImg 1.8.6; labeling 0.1.13; lap 0.4.0; lazy_loader 0.3; libclang 16.0.0; lvis 0.5.3; lxml 4.9.2; Markdown 3.4.3; MarkupSafe 2.1.3; mask-rcnn-tf2 1.0; matplotlib 3.7.1; mkl-fft 1.3.6; mkl-random 1.2.2; mkl-service 2.4.0; networkx 3.1; numpy 1.24.4; oauth2client 4.1.3; oauthlib 3.2.2; object-detection 0.1; objsize 0.6.1; opencv-contrib-python 4.8.0.74; opencv-python 4.8.0.74; opencv-python-headless 4.8.0.74; opt-einsum 3.3.0; orjson 3.9.1; packaging 23.1; pandas 2.0.3; Pillow 9.5.0; pip 23.1.2; portalocker 2.7.0; promise 2.3; proto-plus 1.22.3; protobuf 3.19.6; psutil 5.9.5; py-cpuinfo 9.0.0; pyarrow 9.0.0; pyasn1 0.5.0; pyasn1-modules 0.3.0; pycocotools 2.0; pydot 1.4.2; pymongo 3.13.0; pyparsing 2.4.7; PyQt5 5.15.9; PyQt5-Qt5 5.15.2; PyQt5-sip 12.12.2; python-dateutil 2.8.2; python-slugify 8.0.1; pytz 2023.3; PyWavelets 1.4.1; pywin32 306; PyYAML 5.4.1; regex 2023.6.3; requests 2.31.0; requests-oauthlib 1.3.1; rsa 4.9; sacrebleu 2.2.0; scikit-image 0.21.0; scikit-learn 1.3.0; scipy 1.11.1; sentencepiece 0.1.99; seqeval 1.2.2; setuptools 67.8.0; six 1.16.0; tabulate 0.9.0; tensorboard 2.9.1; tensorboard-data-server 0.6.1; tensorboard-plugin-wit 1.8.1; tensorflow 2.9.1; tensorflow-addons 0.20.0; tensorflow-datasets 4.9.0; tensorflow-estimator 2.9.0; tensorflow-hub 0.13.0; tensorflow-io 0.31.0; tensorflow-io-gcs-filesystem 0.31.0; tensorflow-metadata 1.13.0; tensorflow-model-optimization 0.7.5; tensorflow-text 2.10.0; termcolor 2.3.0; text-unidecode 1.3; tf-models-official 2.10.1; tf-slim 1.1.0; threadpoolctl 3.1.0; tifffile 2023.7.18; toml 0.10.2; tqdm 4.65.0; typeguard 2.13.3; typing_extensions 4.7.1; tzdata 2023.3; uritemplate 4.1.1; urllib3 1.26.16; webencodings 0.5.1; Werkzeug 2.3.6; wget 3.2; wheel 0.38.4; wrapt 1.15.0; zipp 3.15.0; zstandard 0.21.0

ANEXO A - LICENÇA DO LABELIMG

Copyright (c) <2015-Present> Tzutalin
Copyright (C) 2013 MIT, Computer Science and Artificial Intelligence Laboratory. Bryan Russell, Antonio Torralba, William T. Freeman
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

ANEXO B - LICENÇA DO NUMPY

Copyright (C) 2008-2023 Stefan van der Walt <stefan@mentat.za.net>, Pauli Virtanen <pav@iki.fi>

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ANEXO C - LICENÇA DO OPENCV

Apache License Version 2.0, January 2004 http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License"shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor"shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity"shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control"means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50outstanding shares, or (iii) beneficial ownership of such entity.

"You"(or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source"form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object"form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work"shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works"shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution"shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted"means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."
"Contributor"shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.
2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE"text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS"BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

ANEXO D - LICENÇA DO ROS

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. * Neither the name of copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ANEXO E - LICENÇA DO SORT

GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License"refers to version 3 of the GNU General Public License.

"Copyright"also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program"refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees"and "recipients"may be individuals or organizations.

To "modify"a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version"of the earlier work or a work "based on"the earlier work.

A "covered work"means either the unmodified Program or a work based on the Program.

To "propagate"a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey"a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices"to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code"for a work means the preferred form of the work for making modifications to it. "Object code"means any non-source form of a work.

A "Standard Interface"means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries"of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source"for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate"if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product"is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used"refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information"for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions"are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions"within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction"is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor"is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims"are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control"includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license"is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant"such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying"means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory"if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version"applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS"WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

ANEXO F - LICENÇA DO TENSORFLOW

Apache License Version 2.0, January 2004 http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License"shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor"shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity"shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control"means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50outstanding shares, or (iii) beneficial ownership of such entity.

"You"(or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source"form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object"form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work"shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works"shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution"shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted"means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."
"Contributor"shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.
2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE"text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS"BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS