COLÉGIO DANTE ALIGHIERI
Programa de Pré-Iniciação Científica “Cientista Aprendiz”

LEONARDO PASCHOAL BARTOCCINI

LARA MEGDA SCHUSTERSCHITZ

SafeSkies: Desenvolvimento de um protótipo para detecção e rastreamento de balões em áreas de risco utilizando inteligência artificial

São Paulo

2025

LEONARDO PASCHOAL BARTOCCINI
LARA MEGDA SCHUSTERSCHITZ

SafeSkies: Desenvolvimento de um protótipo para detecção e rastreamento de balões em áreas de risco utilizando inteligência artificial

Relatório de pesquisa apresentado à 24ª Feira Brasileira de Ciências e Engenharia (FEBRACE) para avaliação do projeto.

Categoria: Ciências Exatas e da Terra
Subcategoria: Ciência da Computação

Orientador: Prof. Me. Rodrigo Assirati Dias

Coorientador: Prof. Me. Wayner de Souza Klën

São Paulo

2025

Agradecimentos

Primeiramente, expressamos nossa profunda gratidão ao Colégio Dante Alighieri pelo inestimável apoio que nos foi concedido ao longo do desenvolvimento deste projeto. Agradecemos especialmente à equipe do programa de Iniciação Científica Júnior "Cientista Aprendiz", que forneceu os recursos e a orientação necessários para a materialização deste protótipo.

Em seguida, agradecemos ao nosso coorientador, Prof. Wayner de Souza Klën, por todo o apoio que nos forneceu neste processo. Suas ideias brilhantes e seu conhecimento imensurável em Física foram essenciais para que superássemos desafios que muitas vezes nos pareciam insuperáveis.

Além disso, gostaríamos de agradecer imensamente ao nosso orientador, Prof. Rodrigo Assirati Dias, que esteve conosco desde o início desta jornada. Sua dedicação e apoio constantes fizeram toda a diferença ao longo deste percurso.

Agradecemos também ao Sr. Lucas Meyer, do AI For Good Lab, pelo seu auxílio técnico na área de visão computacional, e à Revista InCiência, pela oportunidade de publicação e divulgação do projeto.

Por fim, deixamos nossa sincera gratidão aos nossos pais e familiares, que sempre nos incentivaram e nos deram forças para superar os altos e baixos deste longo processo.

Sem o apoio e a confiança de todos, este projeto jamais teria se tornado realidade.

Resumo

BARTOCCINI, Leonardo Paschoal; SCHUSTERSCHITZ, Lara Megda. **SafeSkies**: Desenvolvimento de um protótipo para detecção e rastreamento de balões em áreas de risco utilizando inteligência artificial. 2025. 62 f. Relatório de Pesquisa – Programa de Pré-Iniciação Científica, Colégio Dante Alighieri, São Paulo, 2025.

A soltura de balões é uma tradição cultural brasileira cujas origens remontam ao século XVI. Apesar de ilegal, essa prática persiste até os dias atuais, representando um risco socioeconômico e ambiental para o país. Por utilizarem fogo para locomoção, os balões figuram entre os principais causadores de incêndios em áreas de preservação ambiental e também representam uma ameaça ao espaço aéreo nacional, pois não são detectados por radares e podem colidir com aeronaves. Diante disso, este projeto tem como objetivo detectar, rastrear e prever a trajetória de balões em áreas de risco, como aeroportos e unidades de conservação, por meio da instalação de câmeras *in loco* associadas ao modelo de visão computacional YOLOv10. O sistema foi treinado com milhares de imagens de balões, pássaros e aviões, permitindo sua identificação em imagens e vídeos com 94% de precisão. Em seguida, aplicou-se o método da triangulação para deduzir uma equação capaz de determinar a posição de um balão a partir da distância entre duas câmeras e de seus ângulos horizontais e verticais. Após essa etapa, foi desenvolvido um protótipo composto por duas câmeras, um sistema de pan-tilt e um Arduino Uno. Esse sistema de rastreamento utiliza um controlador PID, que calcula o melhor ajuste horizontal e vertical para cada câmera, realizado por servomotores controlados pelo Arduino, de modo a manter o balão centralizado e garantir medições precisas. Também foram implementados o cálculo do expoente de Lyapunov, para estimar o horizonte de previsibilidade, e um sistema de alertas automáticos para notificar as autoridades assim que um balão for detectado. Por fim, serão integradas bases de dados meteorológicos para aprimorar a acurácia das previsões de deslocamento e realizar-se-ão testes em ambientes reais. Dessa forma, espera-se que as autoridades possam ser alertadas com antecedência sobre o possível local de queda de um balão, mitigando seus impactos antes que um incêndio se alastre.

Palavras-chaves: Balões. Incêndios Florestais. Inteligência Artificial.

Abstract

BARTOCCINI, Leonardo Paschoal; SCHUSTERSCHITZ, Lara Megda. **SafeSkies**: Development of a prototype for hot-air balloon detection and tracking in high-risk areas using artificial intelligence. 2025. 62 p. Research Report – Scientific Apprenticeship Program, Colégio Dante Alighieri, São Paulo, 2025.

The release of hot-air balloons is a Brazilian cultural tradition whose origins date back to the 16th century. Despite being illegal, this practice persists to this day, posing socioeconomic and environmental risks to the country. Because they use fire for propulsion, balloons are among the main causes of wildfires in environmental preservation areas and also represent a threat to national airspace, as they cannot be detected by radar and may collide with aircraft. In view of this, the present project aims to detect, track, and predict the trajectory of balloons in risk areas—such as airports and conservation units—through the installation of *in loco* cameras integrated with the YOLOv10 computer vision model. The system was trained with thousands of images of balloons, birds, and airplanes, enabling identification in images and videos with 94% accuracy. Next, the triangulation method was applied to derive an equation capable of determining a balloon's position based on the distance between two cameras and their horizontal and vertical angles. Following this stage, a prototype was developed consisting of two cameras, a pan-tilt system, and an Arduino Uno. This tracking system employs a PID controller that calculates the optimal horizontal and vertical adjustments for each camera, performed by servomotors controlled by the Arduino, in order to keep the balloon centered and ensure precise measurements. The calculation of the Lyapunov exponent was also implemented to estimate the predictability horizon, along with an automatic alert system to notify authorities as soon as a balloon is detected. Finally, meteorological databases will be integrated to improve the accuracy of displacement predictions, and real-world tests will be conducted. Thus, it is expected that authorities will be alerted in advance to the possible landing site of a balloon, enabling the mitigation of its potential impacts and the prevention of fires in both urban and natural areas.

Keywords: Hot-air Balloons. Forest Fires. Artificial Intelligence.

Lista de figuras

Lista de abreviaturas e siglas

IA	Inteligência Artificial
ML	*Machine Learning* (Aprendizado de Máquina)
RNA	Rede Neural Artificial
RGB	*Red, Green, Blue* (Vermelho, Verde, Azul)
CNN	Rede Neural Convolucional
IDE	Ambiente de Desenvolvimento Integrado
GPU	Unidade de Processamento Gráfico
mAP	*Mean Average Precision* (Precisão Média)
IoU	*Intersection over Union* (Interseção sobre União)
API	Interface de Programação de Aplicações
PCA	*Principal Component Analysis* (Análise de Componentes Principais)

Sumário

1 Introdução

1.1 História dos balões no Brasil

A festa junina é uma celebração que ocorre no mês de junho, cuja origem remonta às festividades pagãs europeias que festejavam a passagem da primavera para o verão. Décadas depois, essa festa foi incorporada ao calendário católico para comemorar os dias de Santo Antônio (13/06), São João (24/06) e São Pedro (29/06). Com a chegada dos portugueses na América, essa tradição foi trazida para o continente, sendo aos poucos modificada e se tornando uma das mais tradicionais celebrações da cultura brasileira (Nutrição FSP, 2023). Para celebrar tal momento de alegria e festividade, se tornou muito comum soltar balões à céu aberto.

A soltura de balões não tripulados é uma prática cultural antiga que se mantém de forma adaptada até os dias de hoje. Acredita-se que a tradição tenha sido trazida ao Brasil pelos colonizadores portugueses, que utilizavam balões para anunciar o início de festas e eventos. Esses balões serviam como uma forma de sinalizar festividades aos cidadãos (WALBERT, 2013).

Contudo, essa prática não se restringe unicamente à celebração da festa junina. Atualmente, no Brasil, há registros da soltura de milhares balões durante o ano todo, os quais são usados, dentre outros motivos, como forma de celebrar e agradecer aos santos pelos pedidos atendidos, geralmente relacionados a namoro ou casamento (TENÓRIO, 2022).

Entretanto, seja por desconhecimento ou por completo descaso, as pessoas que os soltam causam enormes danos ao patrimônio nacional. Esses balões, apesar de serem símbolos culturais, representam um enorme risco à sociedade e ao meio ambiente.

1.2 Impactos causados por balões

De acordo com o Centro de Investigação e Prevenção de Acidentes Aeronáuticos (CENIPA), a cada ano, cerca de 100 mil balões são soltos no Brasil, sendo que os estados de São Paulo e Rio de Janeiro lideram o número de avistamentos (DECEA, 2023). Apesar de muitas vezes ser vista como uma prática inofensiva, a soltura de balões está sujeita a severas consequências legais. O artigo 42 da Lei Federal 9.605/98 proíbe “fabricar, vender,

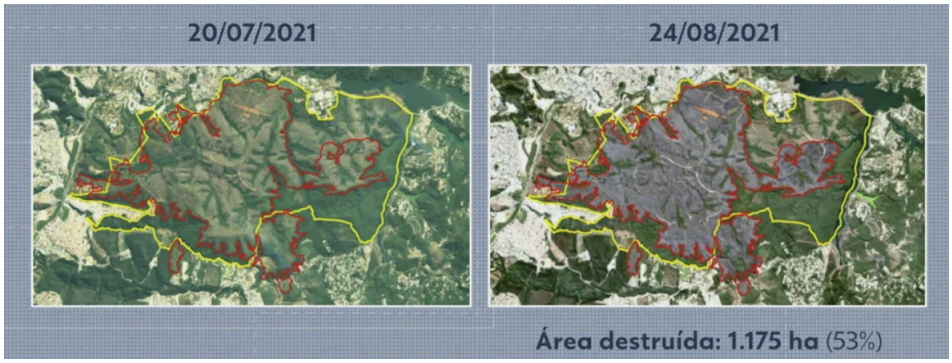
transportar ou soltar balões que possam provocar incêndios nas florestas e demais formas de vegetação, áreas urbanas ou qualquer tipo de assentamento humano". A pena varia de 1 a 3 anos de prisão, multa, ou ambas as penas cumulativamente (Presidência da República, 1998).

Os balões de ar quente utilizam fogo para elevação. Ao caírem, essa chama pode se alastrar, dando início a um incêndio que pode tomar grandes proporções. Caso o local da queda seja uma unidade de conservação ambiental, as perdas de fauna e flora são imensuráveis, ocasionando a morte de milhares de plantas e animais, alguns ameaçados de extinção (AXIMOFF; BARRETO; KURTZ, 2020). Além disso, as queimadas provocam uma elevação no percentual de dióxido de carbono na atmosfera, intensificando o efeito estufa e agravando as mudanças climáticas e o aquecimento global.

Segundo Eufrasio *et al.* (2022): "Um dos principais fatores que afeta o meio ambiente está ligado aos incêndios florestais que podem ter origem por meio das atividades como a agricultura, a pecuária, queima de resíduos de forma inadequada, a queda de balões, além das queimadas de origem criminosa." Esses incêndios florestais podem se alastrar ao ponto de praticamente destruir uma reserva ambiental inteira, como ocorreu no Parque Estadual do Juquery, que abriga o último grande remanescente de Cerrado na Região Metropolitana de São Paulo.

Em 2021, um incêndio que durou quatro dias gerou perdas de 1.175 hectares no Parque do Juquery, o que corresponde a 53% da área verde do parque. A proporção desse desastre foi grande ao ponto de moradores de diversas regiões da capital paulista terem relatado uma "chuva de fuligem" invadindo suas casas (G1 SP, 2021). A causa do incêndio foi justamente a queda de um balão em uma área de eucaliptos dentro do parque. "O balão caiu na copa de um eucalipto, que estava seco devido à geada. As fagulhas foram se espalhando e não foi possível defender a área de vegetação rasteira", afirma o bombeiro civil João Luiz da Cruz, que atuou no combate ao incêndio na ocasião. De acordo com os relatos, naquele dia, cerca de 20 balões sobrevoavam a área do parque, sendo isso algo comum na região, especialmente nos fins de semana (SP Notícias, 2024).

Figura 1 – Comparação do Parque Estadual do Juquery antes e depois do incêndio.



Fonte: Disponível em: ⟨https://dante.pro/juqueryg1⟩. Acesso em: 13 ago. 2024.

Figura 2 – Fotos do incêndio no Parque Estadual do Juquery.

Fonte: Disponível em: ⟨https://dante.pro/fotosjuquery⟩. Acesso em: 13 ago. 2024.

Outro exemplo é o Parque Nacional da Barra da Tijuca, localizado no Rio de Janeiro, Brasil, que tem cerca de 3.300 hectares, sendo reconhecido como a maior floresta urbana do mundo. Essa reserva enfrenta constantes desafios devido a incêndios e expansão urbana. De acordo com o artigo *"Fire and Restoration of the World's Largest Urban Forest in Rio de Janeiro City, Brazil"* (MATOS; SANTOS; CHEVALIER, 2002), no período de 1991 a 2000, os bombeiros florestais registraram uma média de 75 ocorrências de incêndios por ano. As causas incluem balões de ar quente (24%), incêndios intencionais (24%), queima de lixo (21%) e práticas religiosas (17%).

Logo, pode-se perceber que a Região Sudeste é a principal afetada por quedas de balões. Nessa região, há um predomínio dos biomas Mata Atlântica e Cerrado, ambos considerados *hotspots* de biodiversidade, ou seja, são áreas com uma grande variedade de espécies de fauna e flora que se encontram sob ameaça extrema (BRANDON *et al.*, 2005). Mais especificamente, segundo a ONG Conservation International (2024), para que uma área seja classificada como um *hotspot*, ela deve apresentar ao menos 1.500 plantas vasculares endêmicas e deve ter menos de 30% de sua área original preservada. Portanto, a

conservação desses biomas é fundamental para a garantia do equilíbrio ecológico e climático no Brasil e no mundo, e para que tal objetivo seja alcançado, ações devem ser tomadas para mitigar os impactos causados pelos balões nessas localidades.

Os balões não tripulados também causam problemas no setor da aviação. De janeiro a maio de 2024, foram registrados 186 avistamentos de balões, vários deles próximos a aeroportos. Os mais afetados foram o Aeroporto Santos Dumont (RJ), com 23 reportes de balões, seguido do Aeroporto Internacional de Viracopos, Campinas (SP), com 21 e, por último, o Aeroporto de Bacacheri, Curitiba (PR), com 14 (Força Aérea Brasileira, 2024). No mesmo período do ano anterior, o aeroporto de Viracopos, em Campinas, registrou média de uma queda de balão a cada oito dias na área interna do aeroporto (G1 Campinas e Região, 2023).

Além de gerar atrasos e prejuízos financeiros às companhias aéreas e ao aeroporto, essas quedas representam riscos de possíveis incêndios, sendo perigosos para a segurança dos passageiros e trabalhadores do aeroporto (MARQUES; GARCIA, 2021). Segundo a Associação Brasileira de Empresas Aéreas (Abear), uma paralisação de 9 horas no Aeroporto de Congonhas em 2022 gerou um prejuízo que superou os R$ 15 milhões, uma perda de aproximadamente R$ 28 mil por minuto (GLOBO, 2022). Em aeroportos de maior porte, uma queda de balão poderia gerar prejuízos ainda maiores.

Os aviões também correm risco de colidirem com os balões durante o voo, pois "além de possuírem características de incontrolabilidade, os balões não são detectados pelos radares das aeronaves, nem dos controladores de tráfego aéreo"(SCHMITT, 2018). Ou seja, esses balões só podem ser detectados através do avistamento por um piloto ou por algum cidadão no solo, o qual deve informar aos órgãos responsáveis, que irão avisar os outros pilotos. Entretanto, esse processo leva um tempo considerável, o que pode levar a acidentes trágicos.

Figura 3 – Avistamento de um balão próximo a uma aeronave durante o voo.

Fonte: Disponível em: ⟨https://dante.pro/imgbalao⟩. Acesso em: 29 out. 2024.

Além disso, os balões podem cair em áreas urbanas, resultando na destruição de casas, causando mortes e ferimentos, e deixando os moradores desamparados. A queda desses balões também pode derrubar postes de eletricidade, interrompendo o fornecimento de energia por longos períodos. Além disso, eles podem cair em locais sensíveis, como hospitais e escolas, ou até mesmo nas ruas, colocando em risco a integridade de edifícios e a segurança das pessoas ao redor (Ministério dos Transportes, Portos e Aviação Civil, 2018).

Recentemente, um triste episódio tomou o noticiário nacional. Na Zona Leste de São Paulo, um balão arrastou um carro, ergueu uma moto e atingiu a rede elétrica, iniciando um incêndio próximo a imóveis, dentre eles, uma creche (G1 SP, 2024). Conforme apurado pelo Fantástico, programa da Rede Globo de Televisão, o balão, o qual estava carregado com centenas de fogos de artifício, tinha 75 metros de altura (o equivalente a um prédio de 25 andares), com um custo estimado em mais de 200 mil reais. Os impactos à fiação elétrica foram tão graves ao ponto de deixar um milhão de pessoas sem eletricidade (Fantástico, 2024).

Figura 4 – Fotos do balão que atingiu a Zona Leste de São Paulo.

Fonte: Disponível em: ⟨https://dante.pro/fogosbalao⟩. Acesso em: 20 out. 2024.

Contudo, essa não é a primeira vez que algo do tipo ocorre. Um mês antes do acontecimento, em junho de 2024, foi registrada a queda de um balão em um galpão na Zona Oeste de SP, que também foi incendiado (TV Globo, 2024).

Situações semelhantes também ocorrem com frequência em outras cidades. No Rio de Janeiro, a atividade baloeira toma uma nova proporção, uma vez que está associada ao tráfico e às milícias da cidade, sendo amplamente divulgada nas redes sociais. A prática pode englobar outros crimes como estocagem irregular de material explosivo e apologia a criminosos (Revista Veja Rio, 2023). Em junho de 2024, o Comando Vermelho, principal organização criminosa da cidade, usou um balão para celebrar o aniversário de Antônio Ilário Ferreira, um dos traficantes mais procurados do país (Metrópoles, 2024).

Infelizmente, tais ocorrências têm se tornado cada vez mais comuns. Segundo a Secretaria do Meio Ambiente, Infraestrutura e Logística do Estado de São Paulo, somente nos cinco primeiros meses de 2024, foram aplicados 40 autos de infração ambiental por fabricação, venda, transporte ou ato de soltar balões, representando um aumento de 135% nas autuações contra balões. O tenente Edson Alves de Lima, da PM Ambiental, lembra também que "no início de maio deste ano, um balão caiu sobre a reserva biológica do Tamboré, em Santana do Parnaíba, e incendiou o local. Além da vegetação perdida, animais foram mortos"(Governo de São Paulo, 2024).

Logo, por mais que soltar balões pareça uma prática divertida e inofensiva, na verdade se trata do contrário. Os balões não tripulados (conhecidos popularmente como balões de festa junina) são de grande risco para todas as áreas da sociedade, tanto no âmbito econômico como relacionado à segurança da população, já que uma vez soltos são

incontroláveis e podem cair em qualquer lugar, sejam áreas urbanas ou rurais, gerando danos imensuráveis para os habitantes e proprietários do local (MENINO, 2019).

1.3 Inteligência artificial

A inteligência artificial, também conhecida como IA, é um ramo da ciência da computação que se concentra em desenvolver softwares capazes de realizar tarefas que normalmente só poderiam ser realizadas por seres humanos. Essas atividades geralmente envolvem raciocínio, tomada de decisões, reconhecimento de padrões, compreensão da linguagem, entre outras habilidades (SILVA, 2013).

A IA se trata, portanto, de uma tentativa de simular habilidades humanas artificialmente, de modo a automatizar processos, tornando-os mais rápidos e evitando erros. Logo, ela permite que os seres humanos se aproveitem da alta velocidade de processamento das máquinas para realizar diversas tarefas do dia a dia, permitindo que se concentrem nas atividades que exigem mais criatividade (IBM, 2024).

As inteligências artificiais utilizam algumas técnicas para desenvolver a habilidade de executar essas tarefas. Dentre elas, pode-se citar o aprendizado de máquina (*machine learning*), as redes neurais artificiais e a visão computacional, os quais serão descritos em detalhes a seguir.

1.3.1 Machine learning

O *machine learning*, em português aprendizado de máquina, é uma subárea da inteligência artificial que se concentra no desenvolvimento de algoritmos e modelos que permitem que os computadores aprendam a partir de dados e melhorem seu desempenho em tarefas específicas ao longo do tempo, sem serem explicitamente programados. Isto permite que eles realizem previsões de maneira muito precisa (MONARD; BARANAUSKAS, 2003).

A principal ideia do *machine learning* (também chamado de ML) é permitir que os computadores aprendam e melhorem seu desempenho em tarefas ao serem expostos a uma grande quantidade de dados, de modo que o algoritmo ajuste seus parâmetros para encontrar padrões e fornecer uma saída mais adequada. Esses treinamentos podem

ser feitos tanto de maneira supervisionada, com entradas e saídas desejadas, quanto não supervisionada (NAQA; MURPHY, 2015).

1.3.2 Deep learning

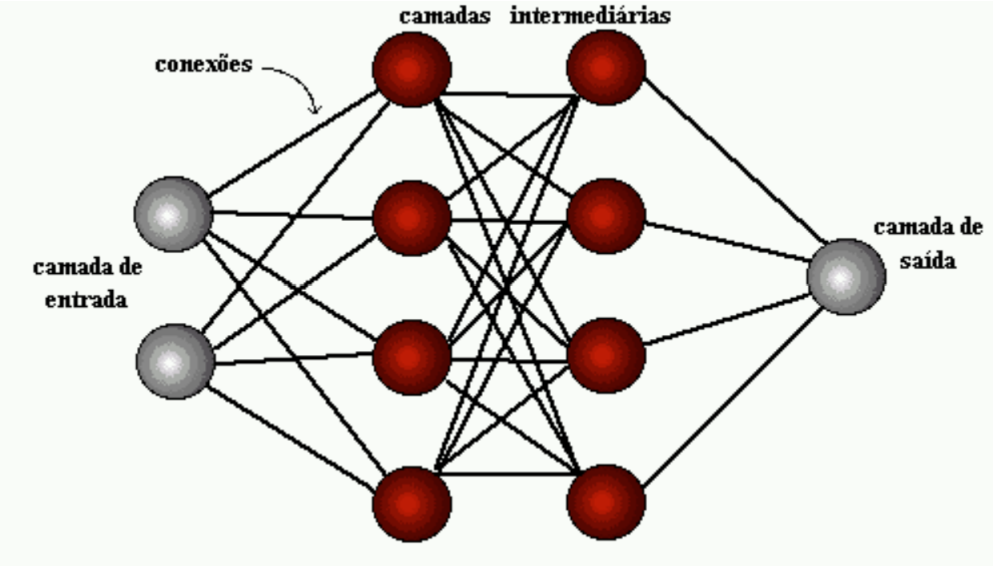
Conforme descrito anteriormente, o *machine learning* é a base do funcionamento das inteligências artificiais. Entretanto, quando nos deparamos com tarefas mais complexas, é preciso utilizar uma outra técnica, conhecida como *deep learning*, cujo funcionamento é baseado no uso de redes neurais artificiais (RNAs) (Google Cloud, 2024).

As RNAs são baseadas no sistema nervoso humano, sendo formadas por uma grande quantidade de neurônios artificiais (também chamados de nós), unidades computacionais interconectadas que recebem entradas, realizam cálculos e geram saídas, funcionando como os neurônios do sistema nervoso (ZOU; HAN; SO, 2009).

Entretanto, um único neurônio não é capaz de executar uma tarefa sozinho. Para que uma pessoa consiga falar, ouvir, caminhar ou realizar qualquer atividade, é necessário que uma grande quantidade de neurônios trabalhe em conjunto, transmitindo informações entre si. Logo, eles se agrupam em nervos e gânglios (sistema nervoso periférico) e no encéfalo (sistema nervoso central) (Universidade Federal de Alfenas, 2022).

O mesmo ocorre com os nós, que não são capazes de realizar tarefas complexas sozinhos. Para o funcionamento das RNAs, é preciso que eles se agrupem em camadas. Cada neurônio artificial está conectado a todos os neurônios da camada anterior e da seguinte, com cada ligação tendo um peso diferente. Conforme a rede neural é treinada, o peso dessas conexões é ajustado para produzir saídas cada vez mais adequadas (RAUBER, 2005).

Figura 5 – Representação do funcionamento de uma rede neural artificial.



Fonte: Disponível em: ⟨https://dante.pro/rna⟩. Acesso em: 10 out. 2023.

1.3.3 Fundamentos matemáticos das redes neurais

Nesta seção, será realizada uma abordagem superficial dos princípios matemáticos responsáveis pelo funcionamento de uma rede neural artificial.

Primeiramente, é preciso compreender que um neurônio artificial nada mais é do que uma função, que irá receber alguns parâmetros e gerar um número como resultado, de modo que, quanto maior esse número, maior a ativação do neurônio.

Antes de apresentar os cálculos, seguem abaixo algumas variáveis e seus respectivos significados:

- $a_i^{(m)}$: i-ésimo neurônio da m-ésima camada.
- $w_{k,j}$: peso da conexão entre o j-ésimo elemento da camada anterior e o k-ésimo elemento da camada em questão.
- b_n: *bias* do n-ésimo neurônio da camada em questão. Representa a partir de que valor a ativação do neurônio é relevante.
- σ: função sigmoide, que converte um dado número em um valor entre 0 e 1. Atualmente, é mais comum usar a função ReLU, mas iremos usar a sigmoide para simplificar os cálculos.
- y_n: valor esperado para o n-ésimo neurônio da camada de saída.

Os valores dos neurônios da camada de entrada já são conhecidos, pois dependem exclusivamente dos dados que estão sendo usados naquele treinamento (cada neurônio da camada 0 pode conter o valor de um determinado pixel de uma imagem, por exemplo). Desse modo, pode-se dizer que o valor de um neurônio da segunda camada é dado pela seguinte expressão:

$$a_0^{(1)} = \sigma(w_{0,0} \cdot a_0^{(0)} + w_{0,1} \cdot a_1^{(0)} + ... + w_{0,n} \cdot a_n^{(0)} + b_0) \tag{1}$$

Ou seja, para encontrar o valor de um nó, é preciso somar os produtos de cada neurônio da camada anterior pelo seu respectivo peso, adicionar o *bias* daquele neurônio e passar o resultado dessa expressão pela função sigmoide. Logo, se as variáveis forem representadas na forma de vetores, a expressão pode ser reescrita da seguinte forma:

$$a^{(1)} = \sigma(W \cdot a^{(0)} + b) \tag{2}$$

Na Equação 2 (acima), W é uma matriz com todos os pesos entre os neurônios das camadas 0 e 1, $a^{(0)}$ é o vetor que contém todos os neurônios da camada 0 e b é o vetor que contém os *biases* de cada neurônio da camada 1. Se esse mesmo raciocínio for aplicado para as camadas seguintes, pode-se chegar ao valor de todos os nós da rede neural (WANG, 2003).

Sabendo quais foram as saídas geradas pela rede neural, deve-se calcular o custo daquele determinado treinamento. O custo indica a precisão do modelo, ou seja, quanto maior o custo, menor a precisão obtida naquele treinamento. O custo é dado pela seguinte expressão:

$$C = (a_0^{(x)} - y_0)^2 + (a_1^{(x)} - y_1)^2 + ... + (a_n^{(x)} - y_n)^2 \tag{3}$$

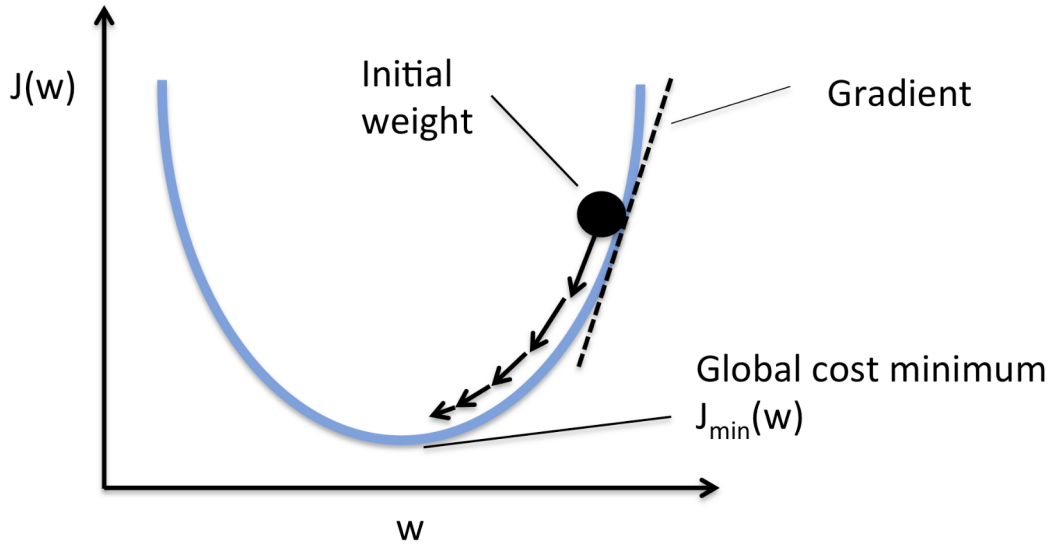
Na Equação 3 (acima), x é a última camada da rede neural. Ou seja, o custo daquele treinamento é a soma dos quadrados das diferenças entre a saída esperada e a saída gerada pelo modelo para cada neurônio da última camada. O custo da rede neural é, portanto, a média dos custos obtidos em diversos treinamento.

Com isso, conclui-se que o objetivo de treinar uma rede neural é reduzir ao máximo o seu custo, aumentando a sua precisão; em outras palavras, dada a função $C(w_{k,j})$ (custo

em função de um determinado peso), deve-se determinar qual o valor que aquele peso deve adquirir para que haja a maior redução possível no custo (LECUN *et al.*, 2002).

Para fazer isso, é utilizado um método conhecido como descida do gradiente. O procedimento consiste em dois passos. Primeiramente, deve-se encontrar o gradiente da função do custo dado o peso atual da conexão que está sendo avaliada, ou seja, é preciso descobrir qual a direção que deveria ser tomada (como o peso deveria ser alterado) para provocar o maior incremento no custo. Em seguida, deve-se deslocar o peso na direção oposta à do gradiente (gradiente negativo) de maneira proporcional à inclinação da função. Se esse processo for repetido diversas vezes, eventualmente chegar-se-á ao mínimo local da função (AGHDAM; HERAVI, 2017).

Figura 6 – Representação gráfica da descida do gradiente.



Fonte: Disponível em: ⟨https://dante.pro/descidagradiente⟩. Acesso em: 20 out. 2024.

O processo utilizado para calcular o gradiente de cada peso e *bias* da rede neural é conhecido como retropropagação. O método consiste em avaliar como cada parâmetro impacta a função do custo por meio do cálculo de derivadas utilizando a regra da cadeia. Em outras palavras, a retropropagação propaga o erro para trás, ou seja, a partir dos erros na camada de saída, cada peso conectado à camada anterior pode ser ajustado e, com isso, pode-se avaliar como os valores dos neurônios da camada anterior deveriam ser modificados. Como é possível modificar apenas os valores dos pesos, e não dos neurônios, repete-se o mesmo processo com as demais camadas, ajustando todos pesos até alcançar a camada de entrada (LECUN; BENGIO; HINTON, 2015).

Portanto, ao simularem o sistema nervoso humano, as redes neurais artificiais se utilizam das conexões entre neurônios (ou nós) para adquirirem a habilidade de realizar tarefas complexas, a qual vai sendo aprimorada com o treinamento através do *deep learning* e de técnicas como a descida do gradiente e a retropropagação. Dentre os processos que utilizam RNAs em seu funcionamento, pode-se citar a regressão, classificação, processamento de linguagem natural e visão computacional (ISLAM; CHEN; JIN, 2019).

1.3.4 Aplicações

As IAs possuem uma infinidade de aplicações, que permeiam toda a sociedade atual. Elas podem ser usadas no reconhecimento de imagens, voz e escrita manual, no processamento de linguagem natural para *chatbots* e tradutores online, classificação e categorização usadas na detecção de fraudes e spam, recomendações de produtos e conteúdos, entre outros.

O *machine learning* também tem ganhado espaço na indústria. Ele pode atuar na prevenção de falhas (manutenção preditiva) e no controle automático de qualidade dos produtos. Na medicina, auxilia na detecção precoce de doenças e na descoberta de novos medicamentos, salvando muitas vidas. Além disso, está revolucionando o ramo automobilístico com veículos autônomos, que podem reduzir o número de acidentes de trânsito em até 90% (BERTONCELLO; WEE, 2015).

1.4 Representação digital de imagens e vídeos

Pixel, abreviação de *picture element* (elemento de imagem, em inglês), é um pequeno quadrado que assume uma determinada cor dependendo do que estiver representando. Essas cores são geradas através do sistema RGB, no qual uma combinação de vermelho, verde e azul é utilizada para representar cada pixel. A quantidade de cada uma dessas cores é que irá determinar a cor que será obtida (CMR College of Engineering & Technology, 2024).

Cada pixel é representado digitalmente por 3 bytes, sendo que cada byte representa a quantidade de uma cor (vermelho, verde ou azul). Cada byte é composto por 8 bits, que é a menor unidade de informação na computação. Cada bit pode assumir 2 valores: 0 ou 1.

Portanto, com 8 bits pode-se representar valores de 0 a 255, totalizando $2^8 = 256$ valores diferentes.

Ou seja, um pixel nada mais é do que um conjunto de 3 bytes, cada um representando a quantidade de vermelho, verde ou azul que varia de 0 a 255. Cada combinação de uma determinada quantidade de cada uma dessas cores gera uma cor diferente.

Ao colocarmos uma grande quantidade de pixels e os organizarmos em uma matriz, obtemos uma imagem. Quanto maior a quantidade de pixels que formam a imagem, maior a sua resolução. Já os vídeos são um conjunto de quadros (ou *frames*, em inglês), que são reproduzidos rapidamente para criar a ilusão de movimento. Cada quadro é formado por uma imagem estática que, conforme explicado anteriormente, é formada por pixels (Faculdade de Computação da Universidade Federal de Uberlândia, 2014).

Logo, em um computador, toda imagem na verdade se resume a uma combinação de pixels de uma determinada cor, que ao serem combinados formam uma figura, enquanto os vídeos nada mais são do que uma grande quantidade de imagens mostradas em sequência.

1.5 Visão computacional e detecção de objetos

Os termos "visão computacional"e "detecção de objetos"são frequentemente usados como sinônimos, mas é importante entender que o primeiro abrange muito mais do que a simples análise de imagens, uma vez que a visão engloba uma série de análises complexas.

A visão computacional abrange desde o reconhecimento de objetos e caracteres até a interpretação de texto e a avaliação de emoções, possuindo um potencial verdadeiramente sobre-humano. Ela tem a capacidade de classificar imagens, detectar e rastrear objetos, além de poder realizar a segmentação de imagens, tarefa que envolve a divisão de uma imagem em diferentes regiões com base na análise dos pixels detectados (Amazon Web Services, 2023).

Quanto à detecção de objetos, esta é uma subárea crucial da visão computacional que é responsável pela identificação e classificação de objetos presentes em imagens (Amazon Web Services, 2024). Ou seja, a visão computacional é uma disciplina multifacetada dedicada a reconstruir e interpretar cenários tridimensionais a partir de imagens bidimensionais.

1.5.1 Modo de funcionamento

Os sistemas de visão computacional usam a inteligência artificial para imitar as capacidades do cérebro humano que são responsáveis pelo reconhecimento e classificação de objetos. Os algoritmos de *machine learning* reconhecem dados visuais inserindo grandes quantidades de informações, identificam padrões comuns nessas imagens ou vídeos e aplicam esse conhecimento para identificar imagens desconhecidas com precisão.

Na Figura 7 (abaixo), vemos uma utilização prática de um modelo de detecção de objetos, conhecido como YOLO. Pode-se observar a criação de uma caixa delimitadora (*bounding box*) que serve como ponto de orientação para o reconhecimento do objeto.

Figura 7 – Detecção e classificação de objetos usando o YOLO.

Fonte: Disponível em: ⟨https://dante.pro/yolo⟩. Acesso em: 15 out. 2023.

A primeira etapa de um sistema de visão computacional envolve análise quantitativa. Isso é alcançado por meio de câmeras e sensores que capturam imagens, sejam elas bidimensionais ou tridimensionais. Toda imagem é bidimensional, pois o meio usado para reproduzi-la, como tela ou papel, possui apenas duas dimensões. Portanto, chamamos de imagem tridimensional aquela que, exibida em um meio de duas dimensões, cria a ilusão de possuir três. A máquina analisa os pixels para identificar a intensidade da luz, cores e outros aspectos, como a profundidade (BRANDIZZI, 2020).

Imagens reais podem conter uma gama infinita de cores e tons. Logo, no processamento de imagens computacionais, é necessário limitar os níveis de cores ou tons atribuídos a cada pixel da imagem, método conhecido como gradação tonal. Essa conversão de uma

imagem colorida para tons de cinza é um processo comum chamado *grayscale*. Isso é feito para simplificar a imagem e torná-la mais fácil de processar, economizando recursos computacionais e reduzindo a sua complexidade (ČADÍK, 2008).

Nas Figuras 8 e 9) (abaixo) pode-se ver como a quantidade de *pixels* e tons utilizados para representar uma imagem afeta a sua qualidade, o que pode dificultar o seu processamento.

Figura 8 – Efeito da limitação de pixels em imagens.

Fonte: Disponível em: ⟨https://dante.pro/pixels⟩. Acesso em: 16 out. 2023.

Figura 9 – Efeito da limitação de tons em imagens.

Fonte: Disponível em: ⟨https://dante.pro/tons⟩. Acesso em: 16 out. 2023.

Após essa fase, ocorre o pré-processamento, processo no qual o sistema faz ajustes na imagem para assegurar a qualidade da análise. Isso inclui a redução de ruídos, ajuste de contraste e outras melhorias. Em seguida, se dá a etapa de extração de características, na qual ocorre o reconhecimento das características matemáticas da imagem, como texturas, formatos e movimentos. Essa etapa possibilita, por exemplo, que as máquinas identifiquem bordas e cantos. Posteriormente, se inicia a fase de detecção e segmentação, na qual a relevância de cada região da imagem é priorizada para o processamento final. Por fim, ocorre a classificação do objeto detectado (QUEIROZ; GOMES, 2006).

1.5.2 Redes neurais convolucionais

As redes neurais convolucionais (CNNs) são uma aplicação das redes neurais artificiais no processamento de dados na forma de grade, em especial imagens. As CNNs tem a capacidade de aprender rapidamente a identificar padrões complexos, como diferenciar e delimitar objetos. Por conta disso, uma de suas principais aplicações é na detecção de objetos e classificação de imagens (O'SHEA; NASH, 2015).

O funcionamento das redes neurais convolucionais, assim como de qualquer outra rede neural, é dividido em camadas. Primeiramente, temos as camadas de convolução, onde há filtros que produzem mapas de características através das imagens de entrada, detectando bordas, cantos, texturas, etc. Em seguida, temos as camadas de *pooling*, que servem para reduzir a dimensão dos mapas de características para tornar a rede mais eficiente. Por último, há as camadas totalmente conectadas, semelhantes às utilizadas em RNAs comuns, que servem para realizar a classificação da imagem (AGHDAM; HERAVI, 2017).

Uma das vantagens das CNNs é que elas podem ser treinadas previamente com um grande conjunto de dados, através de um banco de imagens, e posteriormente ajustadas para realizar tarefas específicas, utilizando conjuntos de dados menores. Essa técnica é conhecida como *transfer learning* (aprendizagem por transferência, na tradução do inglês), sendo muito comum no treinamento de modelos de visão computacional (SHAHA; PAWAR, 2018).

1.5.3 Aplicação na localização de balões

Em muitos cenários, a visão computacional é usada para fins de segurança e prevenção de acidentes, uma vez que a tecnologia promete a capacidade de enxergar com clareza no escuro, através de obstáculos, a grandes distâncias e de processar quantidades massivas de dados em tempo real. Um desses cenários críticos envolve a detecção e monitoramento de balões, que podem causar incêndios em áreas florestais ou urbanas.

Para isso, é possível aplicar técnicas de *machine learning*, que envolve alimentar a rede com um grande conjunto de imagens que contenham balões, bem como imagens sem balões. A rede é treinada para aprender a diferenciar os balões de outros objetos

e identificar sua presença nas imagens. Isso é feito por meio da criação de uma caixa delimitadora (*bounding box*, em inglês) em torno do balão detectado, o que permite sua localização precisa.

Detectar balões pode ser uma tarefa difícil, pois os balões podem ter formas, cores e tamanhos variados. Portanto, a rede neural precisa ser treinada com um conjunto diversificado de dados para garantir uma detecção precisa. Além disso, a detecção de balões deve ser realizada em tempo real, exigindo um desempenho rápido da rede neural.

2 Questão-problema

Como detectar e prever a trajetória e o local de queda de balões em regiões de risco, como unidades de conservação ambiental e aeroportos?

3 Hipótese

Hipotetizamos que, através da criação de um sistema de medição *in loco* baseado em câmeras e visão computacional, é possível detectar e rastrear o movimento de balões em áreas de risco, mitigando danos potenciais.

4 Metodologia

Neste capítulo, serão descritos os passos metodológicos para o desenvolvimento do projeto e verificação da hipótese, apresentando todas as decisões tomadas para a sua realização.

4.1 Levantamento bibliográfico

Inicialmente, foi realizado um levantamento bibliográfico para construção de repertório a respeito da importância dos balões na sociedade e os riscos por eles gerados, assim como conceitos e técnicas de aprendizado de máquina, redes neurais e quaisquer outras áreas que venham a ser relevantes durante a elaboração do protótipo. Esta etapa se estenderá até o final do projeto, de modo a ampliar cada vez mais os conhecimentos sobre as áreas de pesquisa.

4.2 Ferramentas e tecnologias

Nesta seção, serão descritas as principais linguagens, bibliotecas, hardwares e algoritmos que foram utilizados no projeto.

O Python é uma linguagem de programação de alto nível, sendo reconhecida internacionalmente por sua sintaxe limpa e pela sua incrível versatilidade. Ele é utilizado para inúmeros fins, como análise e visualização de dados, desenvolvimento, prototipação e automação para a web, criação de modelos de inteligência artificial e *machine learning*. Além disso, por ser uma linguagem interpretada, ela é extremamente simples de se "debugar"(encontrar e resolver erros no código), o que torna o processo de programação muito mais rápido. Logo, é ideal tanto para programadores iniciantes como para avançados, o que faz do Python a linguagem mais amplamente utilizada em todo o planeta atualmente (YUILL; HALPIN, 2006).

Dessa forma, com base nos fatores apresentados e devido ao fato de ser a linguagem mais recomendada para desenvolvimento de softwares de visão computacional, o Python foi escolhido como a linguagem de programação na qual a maior parte do projeto será desenvolvida.

Já o YOLO — acrônimo de *You Only Look Once* — é um modelo de detecção de objetos em tempo real que revolucionou a área da visão computacional. Através do uso de uma série de técnicas, as quais foram detalhadas na introdução deste relatório de pesquisa, o YOLO consegue analisar uma imagem, detectar objetos presentes nela e classificá-los dentro de uma série de categorias. Atualmente, o YOLO é o modelo de visão computacional mais utilizado no mundo por conta de sua alta velocidade de processamento e excelente precisão na classificação de objetos (LIANG, 2024).

Desse modo, para realizar a detecção dos balões, juntamente à linguagem de programação Python, será utilizada a biblioteca Ultralytics, a qual contém o modelo YOLOv10, a mais atual versão do YOLO, que promete entregar elevada acurácia e baixo tempo de inferência.

Contudo, para que o YOLOv10 possa ser aplicado no *tracking* de objetos em vídeos, é preciso utilizar também a Open Computer Vision Library. A OpenCV é uma biblioteca reconhecida internacionalmente por oferecer diversas ferramentas para execução otimizada de modelos de visão computacional em tempo real, apresentando mais de 12 milhões de downloads mensais (OpenCV, 2025). Em nosso projeto, será utilizada a biblioteca CV2, a mais recente versão da OpenCV, para realizar inferências em vídeos utilizando o modelo YOLO treinado.

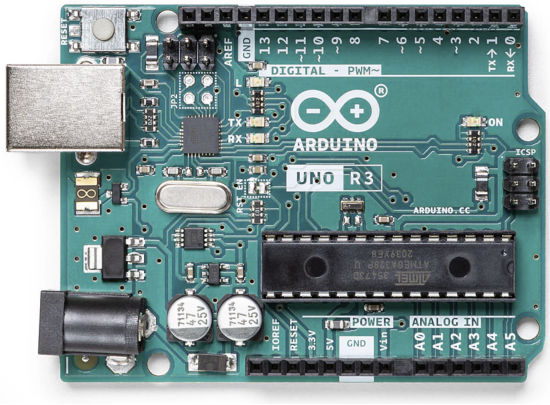
Após o treinamento do modelo, outras bibliotecas do Python serão usadas. Dentre elas, destaca-se a NumPy, que permite a realização de operações matemáticas complexas com um código simples e alta velocidade de processamento. Isso se deve ao fato de que o núcleo da NumPy é escrito em linguagem C, tornando os cálculos mais otimizados. Essa biblioteca será utilizada em nosso projeto principalmente no cálculo de parâmetros relevantes.

Adicionalmente, serão utilizadas as bibliotecas Nolds para o cálculo do expoente de Lyapunov usando o algoritmo de Rosenstein; Scikit-learn para a determinação do eixo que melhor descreve a trajetória do balão (*First Principal Component*); e Requests para o envio automatizado de alertas usando a API do Telegram.

Em seguida, para o desenvolvimento do sistema de movimentação, será necessário utilizar uma placa microcontroladora para controlar os servo-motores. Para o projeto, foi escolhido o Arduino Uno Rev3 (Figura 10 abaixo), o qual é ideal para utilização em protótipos devido à sua versatilidade, baixo custo e excelente documentação, a qual é essencial na criação do código (Arduino, 2025). Vale destacar também que todos os

programas que serão criados para o Arduino são escritos em linguagem C++ usando a Arduino IDE, e os servo-motores serão movimentados usando a biblioteca Servo.

Figura 10 – Foto de uma placa microcontroladora Arduino Uno Rev3.



Fonte: Disponível em: ⟨https://dante.pro/fotoarduino⟩. Acesso em: 20 mai. 2025.

Além disso, a comunicação entre os programas escritos em Python e aqueles escritos usando o Arduino IDE será realizada por meio da porta serial, que conecta o computador ao Arduino. Para isso, será utilizada a biblioteca Serial, a qual enviará o comando ao Arduino Uno. Já o cálculo da correção a ser realizada pelos servo-motores será realizada utilizando um controlador PID por meio da biblioteca Simple-pid devido à sua praticidade de implementação no código aliada a uma robustez significativa. Ambas as bibliotecas estão disponíveis na linguagem Python.

Por fim, destaca-se que todos os códigos e bancos de imagens que serão utilizados no desenvolvimento do protótipo, assim como os resultados obtidos nos testes, serão disponibilizados no repositório do projeto, acessível pelo seguinte link: ⟨https://github.com/laraeleo/SafeSkies⟩.

4.3 Treinamento do modelo

A biblioteca que engloba o YOLOv10 já vem com a capacidade de reconhecer uma série de objetos, como carros, pessoas, pássaros, aviões etc. Entretanto, os balões não são um deles. Logo, para que o modelo consiga reconhecer balões, foi necessário obter milhares de imagens e, em seguida, treiná-lo. Esse processo está descrito a seguir.

4.3.1 Obtenção de imagens

Antes de treinar o modelo, foram coletadas 2526 imagens, sendo 1407 de balões, 458 de pássaros e 334 de aviões, com o objetivo de ensinar à inteligência artificial a identificar e distinguir esses três objetos. Vale destacar que, apesar de o modelo pré-treinado do YOLOv10 ser capaz de identificar pássaros e aviões, é prática comum em processos de *fine-tuning* incluir algumas imagens dessas classes no treinamento para que o modelo não ajuste seus parâmetros unicamente com base nos resultados da detecção de balões.

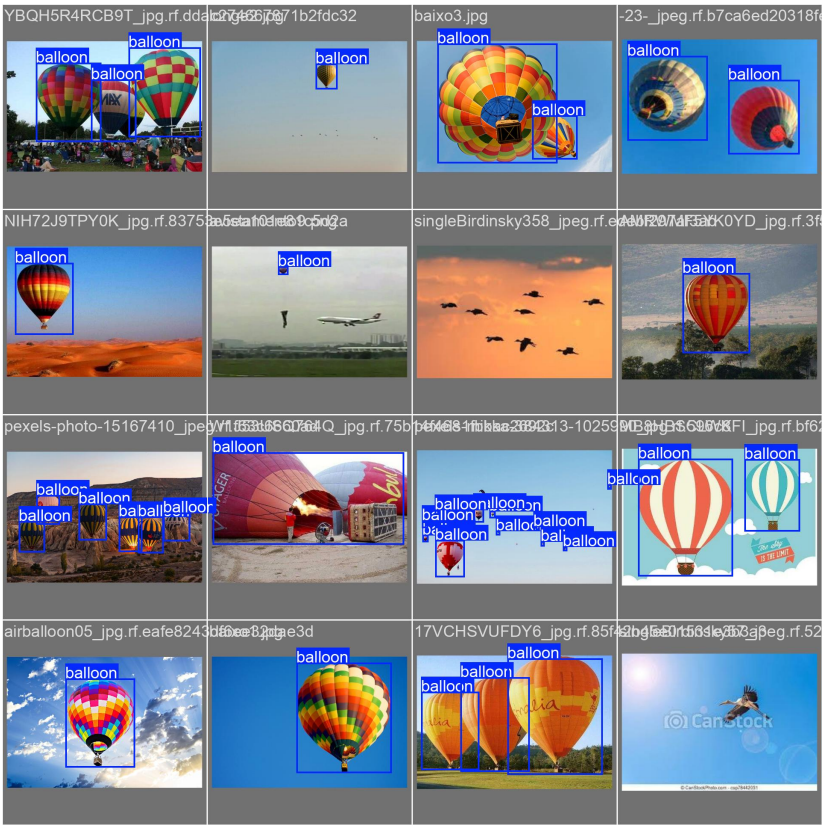
Para obter os dados necessários, foram extraídas fotos dos bancos de imagens listados abaixo:

- *Open Images Dataset V7*: banco de imagens gratuito com 136 fotos de balões.
- *Images.cv - Hot air balloon*: banco de imagens gratuito com 703 fotos de balões.
- *Pexels*: banco de imagens gratuito com 92 fotos de balões.
- *Flickr*: banco de imagens gratuito com 103 fotos de balões.
- *Roboflow Universe*: banco de imagens gratuito com 293 fotos de balões e 334 fotos de aviões.
- *Kaggle*: banco de imagens gratuito com 458 fotos de pássaros.
- Outros: foram utilizadas 80 imagens de balões extraídas de outras fontes gratuitas da internet.

Vale destacar que o *Roboflow Universe* é um repositório que contém inúmeros bancos de imagens. Para este projeto, foram usados os seguintes bancos: *balloon*, *assignment_airballoon*, *KT_aivle*, *hot-air-balloon*, *HOT AIR BALLOON DETECTION* e *planes-zmdv1*.

Para realizar o treinamento, a validação e os testes do modelo, é preciso que as imagens estejam anotadas, ou seja, é preciso delimitar os locais nos quais os objetos podem ser encontrados nas fotos utilizando caixas delimitadoras (*bounding boxes*), conforme mostrado na Figura 11 (abaixo). Com isso, a IA consegue comparar suas previsões com os dados reais (*ground truth*) e, dessa forma, melhorar seu desempenho gradualmente utilizando os algoritmos apresentados na Introdução. Os bancos de imagens *Open Images Dataset V7*, *Roboflow Universe* e *Kaggle* disponibilizam as imagens juntamente com as suas respectivas anotações, agilizando o processo de treinamento. Contudo, os demais bancos de imagens tiveram que ser anotados manualmente.

Figura 11 – Exemplos de caixas delimitadoras em imagens.



Fonte: Imagem de autoria própria.

Após anotar todas as imagens, foi realizada a divisão do banco. Como eram necessárias imagens tanto para treinar o modelo como para testá-lo posteriormente, foi utilizada uma estratégia de divisão de bancos de imagens muito comum, conhecida como 80-20, na qual 80% das imagens são usadas para treinamento e 20% para validação e testes (sendo 10% para cada). Ou seja, dentre as 2199 imagens disponíveis, 1759 foram usadas para treinamento, 220 para validação e 220 para testes.

Por fim, foi executado um procedimento conhecido como *data augmentation* (aumento de dados), no qual as imagens de treinamento sofrem pequenas modificações com o objetivo de aumentar o tamanho do banco, expondo o modelo a novas fotos a fim de melhorar seu desempenho em situações variadas. Para tal, foi utilizado o site *Roboflow*, o que permitiu que o banco de imagens de treinamento fosse triplicado através de modificações na saturação e orientação das imagens. Ou seja, para cada imagem, o sistema criou outras duas imagens idênticas e as rotacionou em um ângulo qualquer entre 0° e 15°, além de ter aumentado ou diminuindo a saturação da imagem em até 34%.

Dessa forma, foi obtido um novo banco de imagens para treinamento com 5633 imagens, de modo que o banco completo (treinamento, validação e testes) totalizou 6073

imagens. Vale destacar que nem todas as imagens de treinamento puderam ser triplicadas devido ao posicionamento das *bounding boxes*. Por fim, durante o treinamento (que será descrito a seguir), o próprio YOLO realizou um novo processo de *data augmentation* por meio da composição de imagens (junção de diversas fotos em uma única imagem para expandir o banco existente).

4.3.2 Detecção de balões

Para treinar e testar o modelo, foi utilizada a IDE do Google Colaboratory, um serviço que permite escrever e executar programas na linguagem Python diretamente do navegador, armazenando o código na nuvem. O Google Colab é amplamente usado na área do *machine learning*, pois é voltado para a execução de Jupyter Notebooks, arquivos nos quais o código é dividido em células, de modo que cada parte do programa possa ser executada separadamente, o que facilita na identificação de bugs. Além disso, o ambiente permite que o treinamento seja realizado utilizando GPUs de última geração (como NVIDIA A100 e L4) a um baixo custo, o que reduz o tempo necessário para treinar a rede neural (Google, 2024).

Ademais, é importante destacar que o YOLO oferece redes neurais de diferentes tamanhos, as quais apresentam diferentes aplicações. O YOLOv10n (Nano), por exemplo, é extremamente veloz para ser treinado e para realizar inferências (previsões) devido ao seu tamanho reduzido, enquanto o YOLOv10x (*Extra Large* ou Extra Grande) apresenta maior precisão, porém é mais lento devido à maior quantidade de camadas em sua rede. Logo, a rede neural escolhida para o protótipo foi o YOLOv10s (*Small* ou Pequeno), uma vez que apresenta uma precisão suficientemente elevada sem comprometer de maneira significativa a sua velocidadede de inferência, possibilitando a execução da detecção em tempo real no futuro.

Por fim, vale destacar que o treinamento foi realizado com um limite de 300 *epochs*, ou seja, 300 iterações sobre o conjunto de imagens. Este valor foi escolhido por ser o recomendado pela Ultralytics, empresa criadora do YOLO (Ultralytics, 2024a). Ademais, foi definido um *early stopping* de 10% do número máximo de iterações (no caso, 30 *epochs*), algo comum no aprendizado de máquina. Em outras palavras, se após 30 iterações sobre as imagens os resultados apresentados pela rede neural não mostrarem nenhuma melhora,

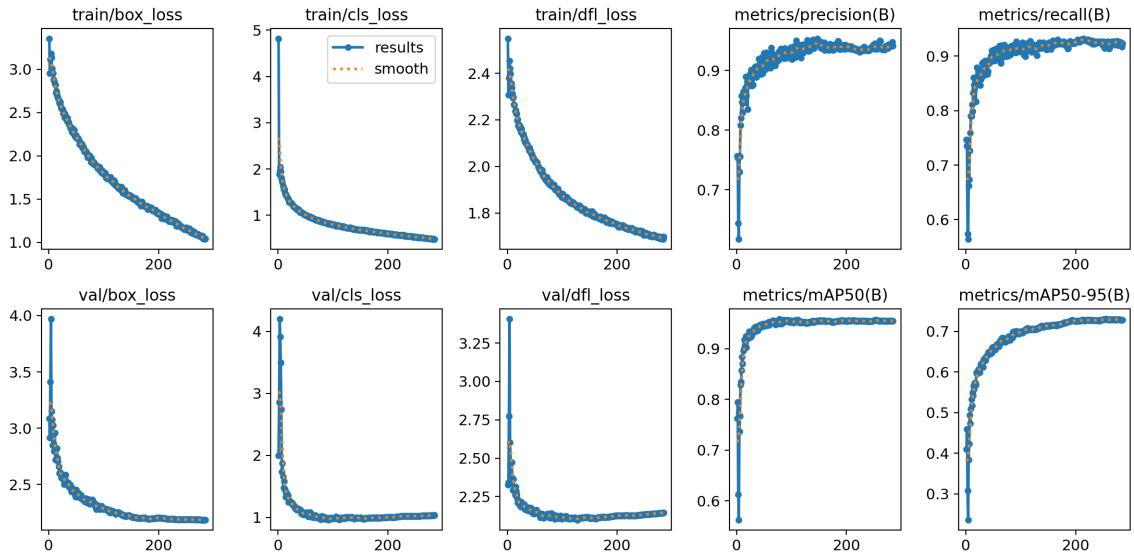
o treinamento será interrompido. Essa decisão foi tomada com o objetivo de evitar a ocorrência de fenômenos indesejados, como *overtraining* e *overfitting*.

O termo *overtraining* se refere ao processo de treinamento excessivo de um modelo de IA, sem um controle adequado, gastando energia e tempo desnecessariamente e podendo levar ao *overfitting*. O *overfitting* ocorre quando uma rede neural se ajusta excessivamente aos dados de treinamento, aprendendo não apenas os padrões gerais, mas também os ruídos e detalhes irrelevantes específicos do conjunto de treinamento. Ou seja, o modelo acaba por memorizar o banco de imagens, o que reduz a sua capacidade de generalização e diminui a sua precisão ao realizar previsões em imagens novas (YING, 2019).

Para evitar a ocorrência desses fenômenos, outras medidas também foram tomadas. Dentre elas, podemos citar a utilização da função de *data augmentation* do próprio YOLO. Através dela, o modelo realiza transformações nas imagens, como rotação, mudança de brilho e contraste, recortes, adição de ruído, entre outros. Dessa forma, além do aumento que o banco de imagens já havia sofrido, um segundo aumento pôde ser aplicado com a finalidade de aumentar melhorar mais a capacidade de generalização do modelo.

Assim que o treinamento foi finalizado, o YOLOv10 gerou alguns gráficos (Figura 12) que auxiliam a compreender como se deu o aprendizado da rede neural. Esses gráficos, também chamados de curvas de aprendizagem, serão analisados a seguir.

Figura 12 – Gráficos do treinamento de detecção de balões em imagens.



Fonte: Imagem de autoria própria. Criada utilizando o YOLOv10.

Inicialmente, analisemos os seis gráficos à esquerda da imagem. Eles representam a variação da perda (quantidade de erros) em função do treinamento, sendo estes os três os

tipos de perda avaliados: *Box Loss*, *Cls Loss*, *DFL Loss*. O primeiro (*Box Loss*) se refere aos erros na localização das caixas delimitadoras previstas em relação às caixas reais. Já o segundo (*Cls Loss*) mede o erro na classificação dos objetos detectados, comparando as previsões com as classes reais (não se aplica ao contexto deste projeto, pois há uma única classe de objetos sendo detectada). Por fim, o *DFL Loss* (*Distribution Focal Loss*) foca em melhorar a precisão da localização das caixas delimitadoras refinando as distribuições previstas para as coordenadas das caixas.

Além disso, mantendo o foco nos seis gráficos à esquerda, observa-se que os três gráficos superiores são relacionados às imagens de treinamento, enquanto os três inferiores estão associadas ao conjunto de imagens de validação, mas todos representam a quantidade de erros cometidos pelo modelo. Dessa forma, o fato de estes seis gráficos apresentarem curvas decrescentes é considerado algo positivo, pois isso aponta que houve uma queda na quantidade de erros conforme o avanço do treinamento.

Agora, voltemos nossa atenção aos quatro gráficos à direita. Estes representam a variação de diferentes métricas, como precisão, *recall*, mAP50 (sinônimo de mAP@0.5) e mAP50-95, ao longo do treinamento. Pode-se notar que todas as curvas são crescentes, o que novamente reforça o sucesso do treinamento, uma vez que demonstra que o modelo apresentou melhora conforme o aumento no número iterações pelas imagens.

Ademais, percebe-se que todos os gráficos atingem um platô rapidamente. Isso indica que o modelo apresentou uma convergência rápida, ou seja, alcançou um patamar de estabilização após poucas iterações sobre o banco de imagens. Isso enfatiza a importância de ter sido estabelecido um *early stopping* no treinamento, já que ele foi interrompido automaticamente na 285ª iteração. Caso não ocorresse essa parada antecipada, o modelo poderia passar por um processo de *overfitting*, tornando-se mais impreciso.

Após o treinamento ser finalizado, o modelo passou a ser capaz de reconhecer balões, pássaros e aviões de diferentes tamanhos e a diversas distâncias. Para que o treinamento pudesse ser considerado bem sucedido, o modelo foi testado utilizando 220 imagens inéditas, ou seja, que a IA nunca havia tido contato. Com isso, foi possível avaliar o desempenho do YOLOv10s ao realizar previsões em situações novas.

4.3.3 Análise de desempenho

Para averiguar o desempenho do modelo, utilizou-se uma Matriz de Confusão, tabela que permite uma melhor visualização do desempenho de um algoritmo de classificação como o YOLO.

Figura 13 – Representação de uma matriz de confusão.

		Valor Predito	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Fonte: Disponível em: ⟨https://dante.pro/matriz⟩. Acesso em: 13 mai. 2024.

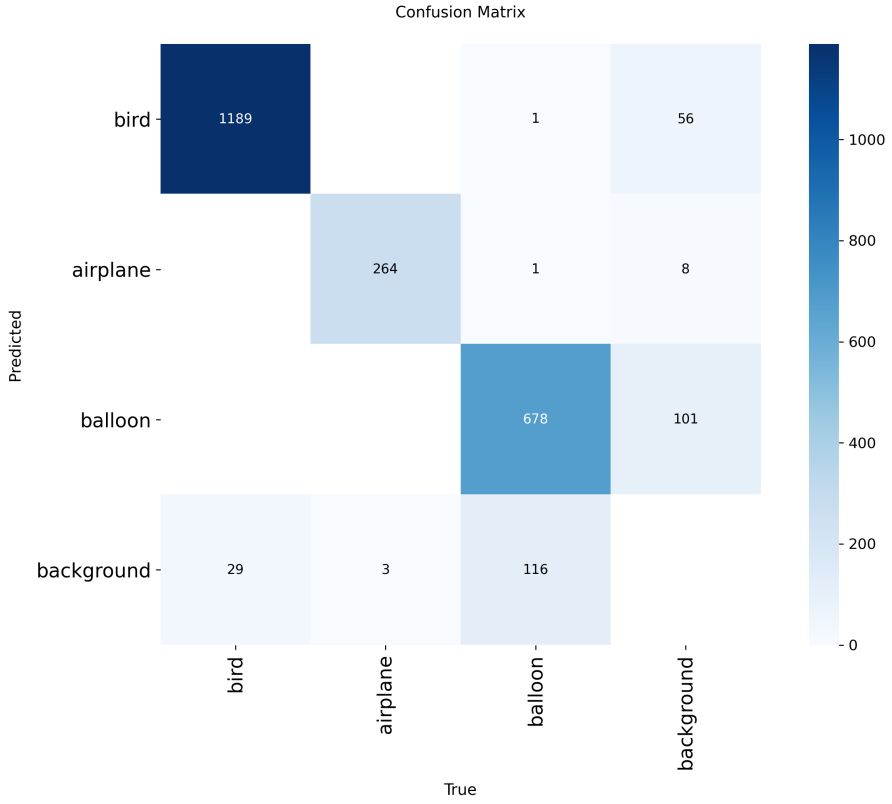
A matriz divide os resultados obtidos em quatro possibilidades:

- Verdadeiro Positivo: tanto a condição prevista como a real são verdadeiras.
- Falso Positivo: a condição prevista foi positiva, porém a condição real é negativa.
- Falso Negativo: a condição prevista foi negativa, porém a condição real é positiva.
- Verdadeiro Negativo: tanto a condição prevista como a real são negativas.

Logo, tanto os verdadeiros positivos quanto os verdadeiros negativos são considerados acertos do modelo, enquanto os falsos positivos e falsos negativos são considerados erros.

Na Figura 14 (abaixo), encontra-se a matriz de confusão criada pelo YOLO com os resultados dos testes no modelo.

Figura 14 – Matriz de confusão dos testes de detecção de balões em imagens.



Fonte: Imagem de autoria própria. Criada utilizando o YOLOv10.

A imagem é dividida em 16 quadrantes, cada um representando uma das possibilidades listadas acima. No eixo vertical consta aquilo que foi previsto, enquanto no horizontal consta a realidade. Assim, percebe-se que os quadrantes da diagonal principal da matriz representam os acertos do modelo; por outro lado, os demais quadrantes representam os erros. Por fim, deve-se notar que, quanto mais escuro o quadrante, maior a quantidade de casos que ele engloba. Portanto, ao analisar a matriz, tem-se que a quantidade de cada caso pode ser expressa na tabela a seguir:

Tabela 1 – Quantidade de casos para cada possibilidade.

Possibilidades	**Casos**
Verdadeiros Positivos	2131
Falsos Positivos	167
Falsos Negativos	150

Fonte – Elaborado pelos autores.

Pode-se observar que a quantidade de verdadeiros negativos é desconsiderada. Isso se deve ao fato de que, em um sistema de visão computacional, seria infrutífero avaliar

quantos foram os locais nos quais não havia balões na imagem e, de fato, eles não foram detectados. Logo, este valor não possui nenhuma aplicação prática na detecção de objetos, além de que, por ser extremamente elevado, poderia dificultar a visualização dos dados que de fato são relevantes.

Para avaliar o desempenho do modelo, foram analisadas as principais métricas que compõem a matriz. Segundo Ultralytics (2024b), são elas: Precisão, *Recall* e *F1 Score*.

- Precisão: proporção de detecções corretas em relação ao total de detecções feitas pelo modelo. Avalia a confiabilidade das detecções. Quanto maior a precisão, menor a quantidade de falsos positivos.

$$\text{Precisão} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Positivos}} \tag{4}$$

- *Recall* (ou Revocação): proporção de objetos reais detectados corretamente em relação ao total de objetos reais. Mede a capacidade do modelo de encontrar todos os objetos. Quanto maior o *recall*, menor a quantidade de falsos negativos.

$$\text{Recall} = \frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Negativos}} \tag{5}$$

- *F1 Score*: média harmônica entre precisão e *recall*, equilibrando as duas métricas em um único valor. Analisa o desempenho geral, considerando tanto os falsos positivos quanto os falsos negativos.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{6}$$

- mAP@0.5: média das precisões ao longo de diferentes níveis de *recall*, considerando uma sobreposição mínima de 50% (IoU ≥ 0.5) entre detecções e objetos reais. É usada para avaliar o desempenho geral do modelo na detecção de objetos, pois trata-se da média entre as precisões médias de cada classe. Se considerarmos N como o número de classes de objetos avaliadas e AP_i como a precisão média (*average precision*) da i-ésima classe, o mAP@0.5 pode ser expresso pela seguinte fórmula:

$$\text{mAP@0.5} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i \tag{7}$$

Através desta análise, foram obtidos os seguintes resultados:

Tabela 2 – Resumo das métricas de treinamento.

Métrica	**Resultado**
Precisão	94,24%
Recall	92,37%
F1 Score	93,29%
mAP@0.5	95,55%

Fonte – Elaborado pelos autores.

Vale destacar que esses valores são fornecidos pelo YOLO, logo podem não coincidir exatamente com os valores calculados com base na Tabela 1. Isso se deve ao fato de que os cálculos são feitos com base na média ponderada entre as métricas de cada classe, enquanto a tabela apresenta uma generalização dos casos.

Por fim, segue abaixo um resumo das principais métricas de treinamento para cada uma das classes avaliadas:

Tabela 3 – Métricas de treinamento por classe.

Classe	**Precisão**	**Recall**	**F1 Score**
Balões	88,16%	82,35%	85,16%
Aviões	97,84%	98,50%	98,17%
Pássaros	96,70%	96,26%	96,48%

Fonte – Elaborado pelos autores.

Pode-se concluir, portanto, que há uma baixa quantidade de erros, tanto falsos positivos (precisão elevada) quanto falsos negativos (*recall* elevado). Além disso, os indicadores de desempenho geral (*F1 Score* e mAP@0.5) corroboram esta ideia, uma vez que se encontram próximos dos 95%. Vale ressaltar também que muitas das imagens utilizadas para testes possuíam baixa resolução, justamente para avaliar a capacidade de inferência do modelo em situações adversas. Desse modo, acredita-se que estes parâmetros serão ainda mais elevados com a utilização de uma câmera de boa qualidade.

Logo, todos os fatores citados anteriormente levam a crer que o treinamento do modelo para detecção de balões em imagens tenha sido bem sucedido, permitindo que esta parte da metodologia seja dada como concluída.

Vale lembrar que todos os programas descritos nesta seção, assim como os resultados dos testes realizados, estão disponíveis no repositório do projeto, na pasta *balloon_detection*, e podem ser acessados pelo link: ⟨https://github.com/laraeleo/SafeSkies/tree/main/model_training⟩.

4.4 Inferências em vídeos

Assim que o modelo passou a ser capaz de reconhecer balões em fotos, foi desenvolvido um programa para realizar o reconhecimento de balões em vídeos. Conforme explicado na introdução, os vídeos nada mais são do que uma sequência de imagens passadas em alta velocidade. Logo, utilizando o modelo que havia sido treinado, foi criado um código que realizasse o *tracking* do objeto, ou seja, que conseguisse identificar o mesmo objeto em várias imagens seguidas, rastreando-o ao longo do vídeo.

Inicialmente, foi criado um programa em linguagem Python com o objetivo de realizar o processamento dos vídeos de maneira assíncrona. O código utiliza a biblioteca OpenCV e é baseado no programa "*predict_video*", desenvolvido por Felipe Tambasco sob o usuário "computervisioneng"e acessado através do repositório "*train-yolov8-custom-dataset-step-by-step-guide*"no GitHub. O código está disponível sob a licença *GNU Affero General Public License*, versão 3 (AGPL-3.0) (TAMBASCO, 2023).

Em seguida, foram realizados alguns testes com vídeos de balões, os quais apontaram que o modelo era capaz de identificar os balões em vídeos, apresentando poucos erros neste processo. O tempo de inferência também aparentava ser viável, uma vez que os vídeos testados foram processados pelo modelo em um tempo menor do que o próprio tempo de execução dos vídeos, o que corroborou a escolha pela utilização do modelo YOLOv10s.

Posteriormente, foi desenvolvido um novo programa em Python para realizar o *tracking* de objetos, mas agora em tempo real. No código, a biblioteca CV2 é utilizada para obter os *frames* do vídeo que são registrados em tempo real e, em seguida, utiliza o modelo treinado para realizar as inferências nesses *frames*. Dessa forma, o código passou a ser capaz de fornecer as coordenadas das *bounding boxes* criadas em torno dos objetos detectados, as quais serão utilizadas posteriormente no sistema de movimentação.

Vale ressaltar que, a partir do momento em que o código passou a ser executado em tempo real, não foi mais possível utilizar a IDE do Google Colab, uma vez que ela não suporta a realização de inferências em tempo real. Com isso, os programas passaram a ser criados e executados utilizando o Visual Studio Code, uma IDE que permite executar Jupyter Notebooks localmente. Dessa forma, apesar de perder o acesso às GPUs Premium do Google Colab, garante-se que o programa possa ser executado mesmo sem acesso à internet, o que será importante para fases futuras do projeto.

Os programas que foram descritos nesta seção também estão disponíveis no repositório do projeto, na pasta *video_inference*, e podem ser acessados pelo link: ⟨https://github.com/laraeleo/SafeSkies/tree/main/video_inference⟩.

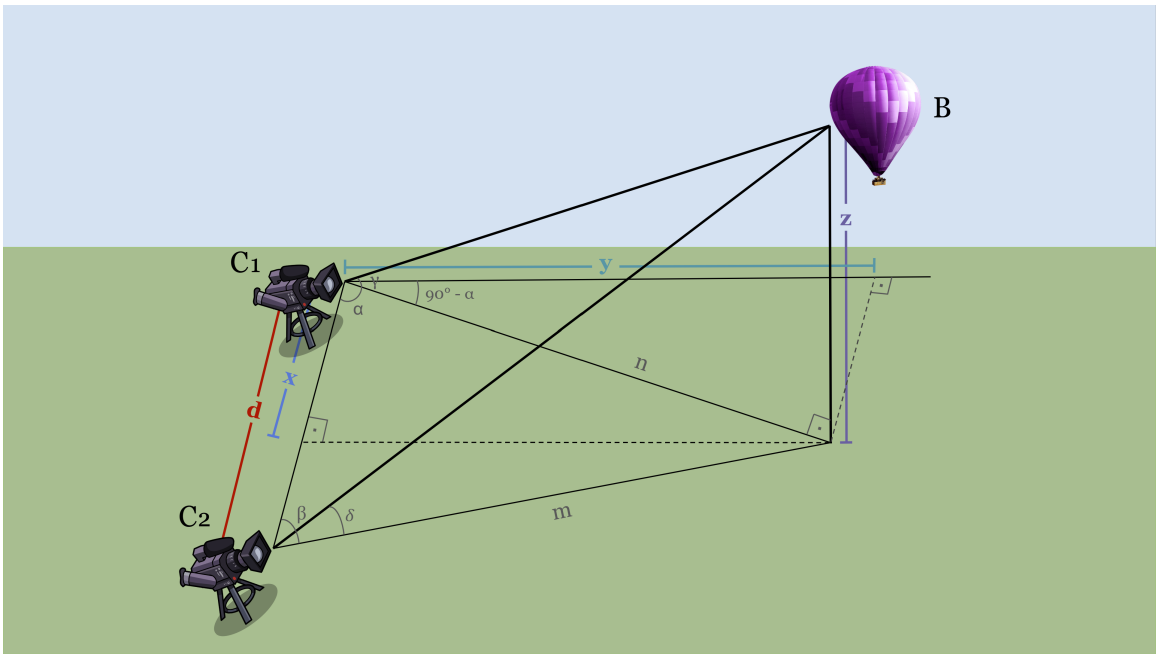
4.5 Cálculo de parâmetros relevantes

Após finalizado o treinamento do modelo de detecção de objetos utilizando YOLOv10, foi iniciado o processo de cálculo de parâmetros relevantes.

Primeiramente, foram determinadas as equações que descrevem a posição de um balão. Isso foi feito com o objetivo de que, no futuro, seja possível verificar a direção, sentido e velocidade de sua movimentação. Essas informações serão fundamentais para que se possa calcular a trajetória do balão e prever onde irá ocorrer sua queda, de modo a avisar às autoridades responsáveis para que possíveis acidentes e incêndios sejam evitados.

O método utilizado para determinar as coordenadas de um objeto no espaço a partir de alguns ângulos e as distâncias entre eles é chamado de triangulação. Esse método, o qual é descrito a seguir, foi usado para determinar as fórmulas que descrevem a posição de um balão no espaço em um dado instante. Contudo, tais cálculos podem ser de difícil entendimento. Logo, para melhorar a melhorar a compreensão dos cálculos desta seção, foi elaborado o esquema representativo presente na Figura 15 (abaixo).

Figura 15 – Esquema representativo do método da triangulação.



Fonte: Imagem de autoria própria.

Para que esses cálculos possam ser realizados, é preciso que sejam conhecidas as seguintes variáveis:

- d: distância entre as câmeras C_1 e C_2.
- α: ângulo $\angle PC_1C_2$, ou seja, trata-se do ângulo horizontal formado entre a câmera C_1 e a projeção do balão no plano (x,y).
- β: ângulo $\angle PC_2C_1$, ou seja, trata-se do ângulo horizontal formado entre a câmera C_2 e a projeção do balão no plano (x,y).
- γ: ângulo $\angle PC_1B$, ou seja, trata-se do ângulo vertical formado entre a câmera C_1 e o balão.
- δ: ângulo $\angle PC_2B$, ou seja, trata-se do ângulo vertical formado entre a câmera C_2 e o balão.

Além disso, definamos as seguintes variáveis relevantes:

- n: distância entre a câmera C_1 e a projeção do balão no plano (x,y).
- m: distância entre a câmera C_2 e a projeção do balão no plano (x,y).

É importante notar que, conhecendo o valor de n, é possível determinar as coordenadas (x, y, z) do balão utilizando razões trigonométricas, uma vez que:

$$B_n = (n \cdot \cos(\alpha), n \cdot \sin(\alpha), n \cdot \tan(\gamma)) \tag{8}$$

Pode-se determinar o valor de n por meio da utilização da Lei dos Senos da seguinte forma:

$$\begin{aligned} \frac{\sin(\beta)}{n} &= \frac{\sin(180^\circ - \alpha - \beta)}{d} \\ \frac{\sin(\beta)}{n} &= \frac{\sin(\alpha + \beta)}{d} \\ n &= d \cdot \frac{\sin(\beta)}{\sin(\alpha + \beta)} \end{aligned} \tag{9}$$

Com isso, basta que os valores das variáveis d, α, β e γ sejam conhecidos para que todas as distâncias possam ser determinadas.

Contudo, vale destacar que o conhecimento do valor da variável δ poderá auxiliar a minimizar os erros experimentais, pois também é possível descobrir as coordenadas do balão utilizando m, uma vez que:

$$B_m = (d - m \cdot \cos(\beta), m \cdot \sin(\beta), m \cdot \tan(\delta)) \tag{10}$$

Aplicando o mesmo procedimento utilizado para n para determinar o valor de m, temos:

$$\frac{\sin(\alpha)}{m} = \frac{\sin(180° - \alpha - \beta)}{d}$$
$$\frac{\sin(\alpha)}{m} = \frac{\sin(\alpha + \beta)}{d}$$
$$m = d \cdot \frac{\sin(\alpha)}{\sin(\alpha + \beta)} \tag{11}$$

Portanto, o valor de B que será utilizado no projeto é dado pela média aritmética entre os valores obtidos por meio de n e m. Ou seja,

$$B = \left(\frac{n \cdot \cos(\alpha) + d - m \cdot \cos(\beta)}{2}, \frac{n \cdot \sin(\alpha) + m \cdot \sin(\beta)}{2}, \frac{n \cdot \tan(\gamma) + m \cdot \tan(\delta)}{2} \right) \tag{12}$$

Em seguida, é possível também determinar a distância das câmeras ao balão através da distância euclidiana. No caso da câmera C_1, temos:

$$C_1B = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Como foi definido que C_1 é a origem do plano, então suas coordenadas são (0,0,0) e, portanto, $x_1 = 0$, $y_1 = 0$ e $z_1 = 0$. Além disso, como x_2, y_2 e z_2 são coordenadas do balão, tem-se que:

$$C_1B_n = \sqrt{(n \cdot \cos\alpha)^2 + (n \cdot \sin\alpha)^2 + (n \cdot \sin\gamma)^2}$$
$$C_1B_n = \sqrt{n^2 \cdot \cos^2\alpha + n^2 \cdot \sin^2\alpha + n^2 \cdot \sin^2\gamma}$$
$$C_1B_n = n \cdot \sqrt{\sin^2\alpha + \cos^2\alpha + \sin^2\gamma}$$

Aplicando a Propriedade Fundamental da Trigonometria:

$$C_1B_n = n \cdot \sqrt{1 + \sin^2\gamma} \tag{13}$$

Utilizando o mesmo raciocínio com as coordenadas obtidas por meio de m, temos:

$$C_1B_m = \sqrt{(d - m \cdot \cos\beta)^2 + (m \cdot \sin\beta)^2 + (m \cdot \sin\delta)^2}$$
$$C_1B_m = \sqrt{d^2 - 2 \cdot d \cdot m \cdot \cos\beta + m^2 \cdot \cos^2\beta + m^2 \cdot \sin^2\beta + m^2 \cdot \sin^2\delta}$$
$$C_1B_m = \sqrt{m^2 \cdot \left(\frac{d^2}{m^2} - \frac{2 \cdot d \cdot \cos\beta}{m} + \cos^2\beta + \sin^2\beta + \sin^2\delta \right)}$$
$$C_1B_m = m \cdot \sqrt{\frac{d}{m} \cdot \left(\frac{d}{m} - 2 \cdot \cos\beta \right) + \cos^2\beta + \sin^2\beta + \sin^2\delta}$$

Dessa forma,

$$C_1B_m = m \cdot \sqrt{1 + \sin^2\delta + \frac{d}{m} \cdot \left(\frac{d}{m} - 2 \cdot \cos\beta\right)} \tag{14}$$

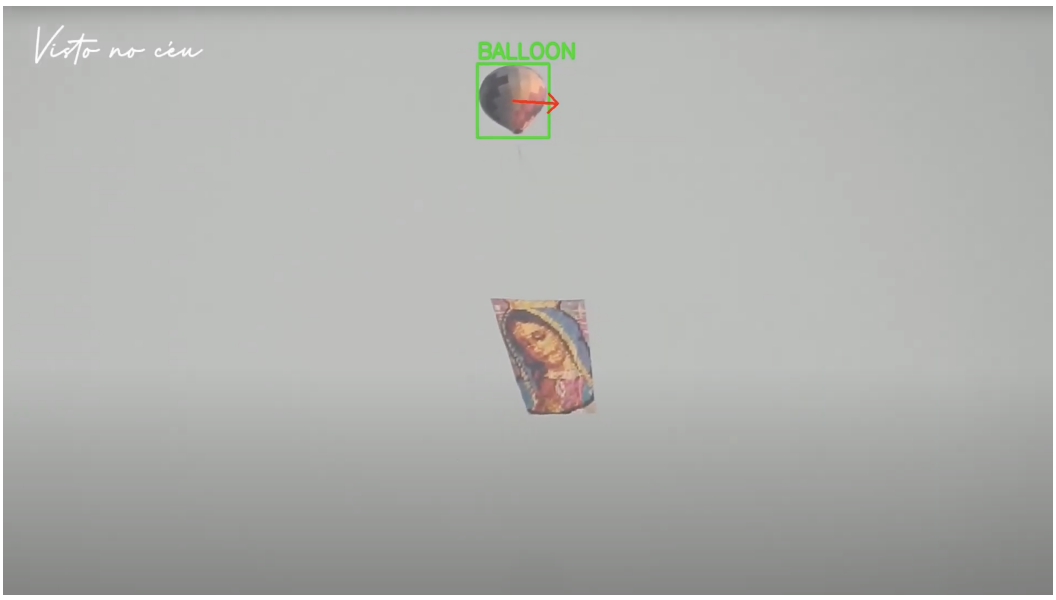
Portanto, utilizando a média aritmética entre os valores encontrados, obtém-se:

$$C_1B = \frac{n \cdot \sqrt{1 + \sin^2\gamma} + m \cdot \sqrt{1 + \sin^2\delta + \frac{d}{m} \cdot \left(\frac{d}{m} - 2 \cdot \cos\beta\right)}}{2} \tag{15}$$

Esse mesmo raciocínio pode ser aplicado para calcular a distância do balão em relação à câmera C_2.

Com as fórmulas de posicionamento determinadas, foi possível calcular o vetor de movimentação do balão. Para isso, basta calcular a diferença nas coordenadas x, y e z entre dois *frames* consecutivos. Já a obtenção do vetor de velocidade pode ser feita dividindo o deslocamento do balão pelo intervalo de tempo entre os quadros. Para suavizar o movimento, usa-se a média dos últimos 10 vetores calculados para descrever a velocidade de um balão em um dado instante. O vetor resultante é visualizado por meio de uma seta partindo do objeto, e seu tamanho é proporcional à magnitude do vetor (vide Figura 16 abaixo). Essa implementação também foi feita utilizando a biblioteca CV2.

Figura 16 – Quadro de um vídeo de balão com seu vetor de velocidade.



Fonte: Imagem de autoria própria obtida utilizando YOLOv10 e CV2.

Após a determinação das fórmulas, foi desenvolvido um programa em linguagem Python que as implementasse utilizando a biblioteca NumPy, reconhecida mundialmente por realizar operações matemáticas complexas com alta velocidade. O código criado

está disponível no repositório do projeto, na pasta *relevant_parameters_calculation*, e pode ser acessado pelo link: ⟨https://github.com/laraeleo/SafeSkies/tree/main/relevant_parameters_calculation⟩.

4.6 Protótipo funcional

Inicialmente, é comum imaginar que bastaria conhecer os ângulos horizontais e verticais de ambas as câmeras para que os cálculos pudessem ser realizados. Contudo, ao analisar as equações deduzidas na seção anterior, algo importante deve ser observado: nem sempre os ângulos formados em relação às câmeras serão iguais aos ângulos formados em relação ao balões. Na verdade, o único caso em que isso ocorre é quando o balão se encontra no centro da imagem capturada pela câmera. Logo, tal diferença é significativa, de modo que, na maioria dos casos, realizar os cálculos se tornaria inviável.

Logo, para resolver essa problemática, foi desenvolvido um protótipo capaz de movimentar as câmeras e manter o balão centralizado na imagem, permitindo a realização dos cálculos de maneira satisfatória. Os passos metodológicos para a sua criação estão descritos a seguir.

4.6.1 Escolha da câmera

Para iniciar a construção do protótipo, foram compradas duas câmeras Logitech C270 HD Webcam. A principal motivação por trás dessa escolha foi a excelente qualidade do dispositivo e seu baixo custo quando comparado aos seus pares. A câmera possui resolução máxima de 720p e 30fps, $0,9$ Megapixels, foco fixo e campo de visão (CDV) diagonal de $55°$, com um custo de aproximadamente 160 reais. Além disso, ela possui compatibilidade com os principais sistemas operacionais — Windows, macOs e chromeOS — e conecta-se ao dispositivo facilmente por meio de uma porta USB-A.

Figura 17 – Foto da câmera Logitech C270 HD Webcam.

Fonte: Disponível em: ⟨https://dante.pro/cameralogitech⟩. Acesso em: 21 mai. 2025.

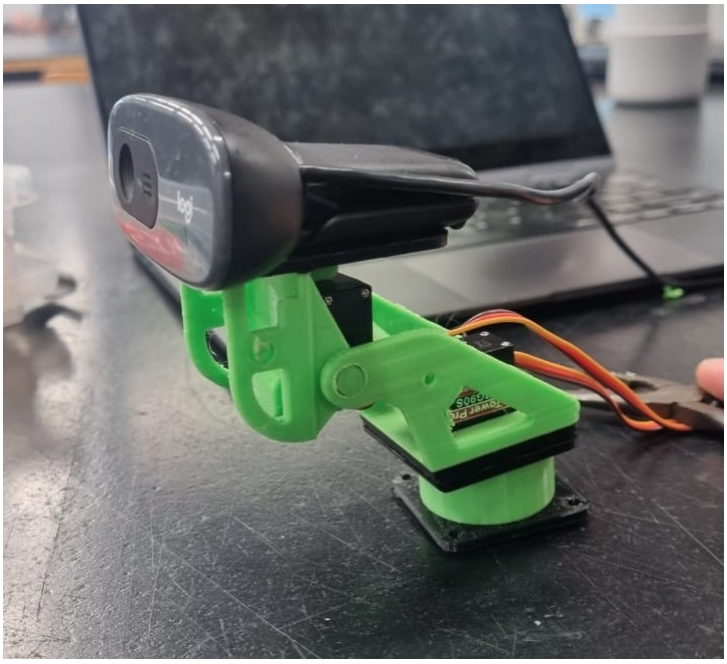
4.6.2 Sistema de movimentação

Em seguida, foi criado um sistema de movimentação horizontal e vertical, mais conhecido como sistema *pan* e *tilt*, o qual permite a movimentação das câmeras.

Com relação ao hardware (parte física do protótipo), foi utilizada uma impressora 3D para imprimir um modelo disponibilizado gratuitamente na internet em PLA, o qual era específico para esse tipo de aplicação. O modelo utilizado foi o "*Heavy Duty Pan&Tilt For FPV*", desenvolvido por Filatech e disponibilizado sob a licença *Creative Commons – Atribuição – Não Comercial – Sem Derivações* (CC BY-NC-ND), acessível em: ⟨https://www.thingiverse.com/thing:3050358⟩ (FILATECH, 2019).

Em seguida, foram colocados servo-motores nesse modelo, permitindo a sua movimentação de 0° a 180° tanto verticalmente quanto horizontalmente. Por fim, os servos foram conectados a um Arduino Uno, uma placa microcontroladora simples e versátil usada para controlar dispositivos eletrônicos.

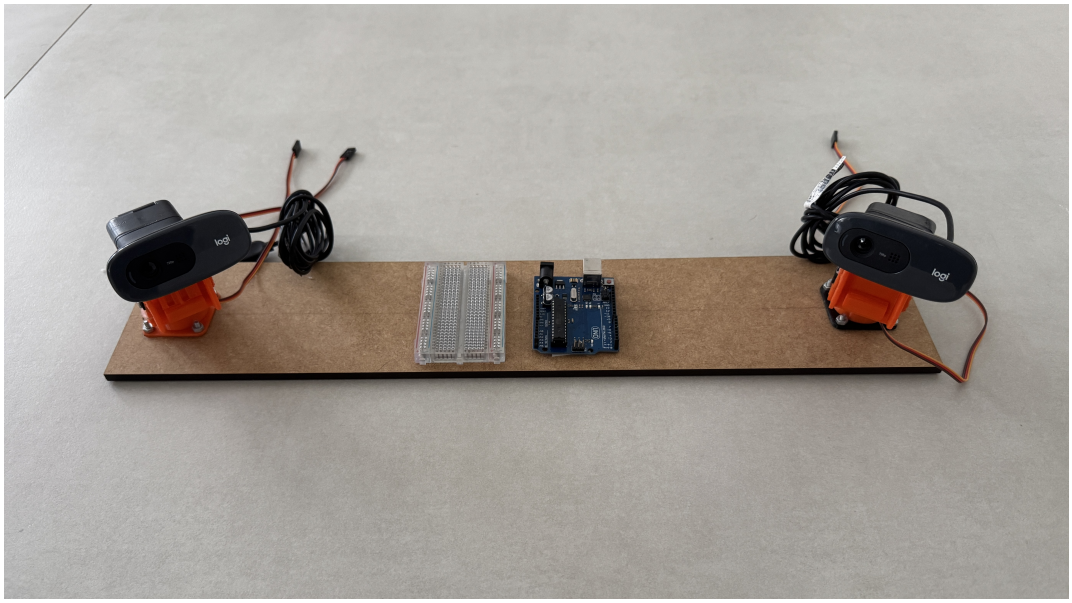
Figura 18 – Foto do protótipo para movimentação das câmeras.



Fonte: Imagem de autoria própria.

Uma barra de madeira de 0,5 metro foi cortada para servir como linha de base fixa, com duas câmeras Logitech C270 HD posicionadas em cada extremidade, a fim de garantir a captura precisa dos ângulos (β, α, γ, δ) necessários para os cálculos pelo método de triangulação. O Arduino Uno e a *protoboard* também foram fixados na barra para facilitar o seu transporte.

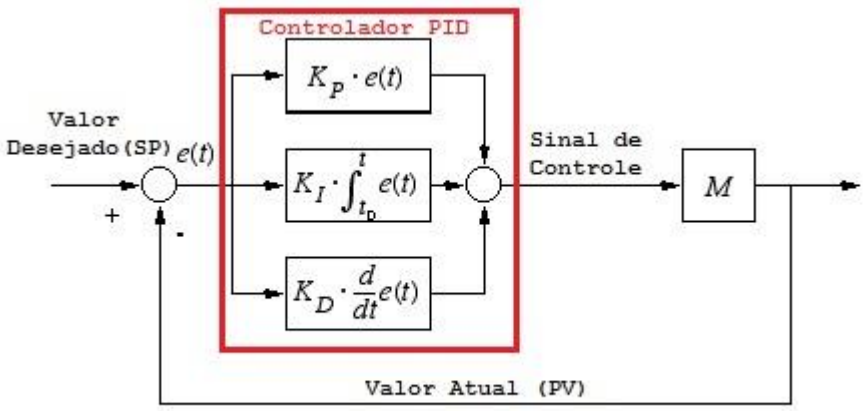
Figura 19 – Foto das duas câmeras montadas em uma barra de 0,5 metro.



Fonte: Autoria própria.

Para realizar o cálculo da correção a ser realizada pelos servo-motores, foi utilizado um controlador PID. Um controlador PID é uma metodologia de engenharia de controle que combina três elementos: proporcional (P), integral (I) e derivativo (D). Em outras palavras, o controlador calcula continuamente o erro — ou seja, a distância entre o centro da *bounding box* que contém o balão e o centro da imagem captada por cada câmera — e aplica ajustes proporcionais, integrais e derivativos para minimizá-lo. Por conta de sua eficiência e precisão, os sistemas PID são amplamente usados na indústria e no setor elétrico (FERMINO *et al.*, 2014). No projeto, a implementação do controlador foi feita utilizando a biblioteca Simple-pid, disponível em linguagem Python.

Figura 20 – Esquema representativo do funcionamento de um controlador PID.



Fonte: Disponível em: ⟨https://dante.pro/esquemapid⟩. Acesso em: 20 mai. 2025.

Em suma, o software do projeto é composto por dois programas. O primeiro, escrito em Python, é responsável por:

- realizar inferências em vídeo utilizando o modelo YOLO e a biblioteca CV2;
- determinar a posição do balão e as distâncias até as câmeras utilizando a biblioteca NumPy;
- calcular os ângulos de rotação desejados utilizando a biblioteca Simple-pid;
- enviar essas informações para o Arduino via comunicação Serial.

Já o segundo programa, escrito em C++ utilizando a IDE do Arduino, é responsável por:

- receber as informações enviadas pelo primeiro programa;
- interpretar os dados recebidos;

- enviar comandos para os servomotores.

A comunicação entre os programas escritos em Python e aqueles escritos usando a Arduino IDE será realizada por meio da porta serial, que conecta o computador ao Arduino. Para isso, será utilizada a biblioteca Serial no Python, a qual enviará o comando ao Arduino Uno. Esse, por sua vez, irá ler o comando e intepretá-lo conforme o seu código, girando os servos até o valor instruído.

Desse modo, ao combinar um modelo de detecção YOLOv10 com o sistema *pan* e *tilt*, um controlador PID e as fórmulas deduzidas, é possível determinar a posição de um objeto no espaço a qualquer instante, assim como sua velocidade, direção e sentido de movimentação.

4.6.3 Expoente de Lyapunov

Após a finalização dos componentes essenciais do protótipo, foram realizadas algumas melhorias adicionais. Dentre elas, destaca-se a implementação do cálculo do expoente de Lyapunov.

O expoente de Lyapunov é uma quantidade amplamente utilizada para quantificar o comportamento caótico de sistemas dinâmicos e pode ser utilizado para avaliar o quanto trajetórias próximas em sistemas dinâmicos se afastam ao longo do tempo. Em outras palavras, o expoente mede o afastamento médio entre duas trajetórias que eram inicialmente quase idênticas, de modo a fornecer uma estimativa da caoticidade do sistema (BARROSO, 2019). A seguir, será realizada uma breve explicação da dedução do expoente para o caso unidimensional, entretanto, a dedução pode ser estendida para sistemas de maiores dimensões.

Inicialmente, escolhe-se uma função $f : X \to X$ e um ponto $x_0 \in X$. Assim, define-se que a órbita de x_0 sob f é a sequência de pontos obtidos ao aplicar f repetidamente:

$$(x_0, x_1 = f(x_0), x_2 = f \circ f(x_0), x_3 = f \circ f \circ f(x_0), ...) \tag{16}$$

De forma geral:

$$x_n = f^n(x_0), n \in \mathbb{N} \tag{17}$$

Agora, define-se uma condição inicial x_0 e um ponto próximo $x_0 + \delta_0$. Seja δ_n a separação entre as órbitas a partir de x_0 e $x_0 + \delta_0$:

$$\delta_n = f^n(x_0 + \delta_0) - f^n(x_0) \tag{18}$$

Assumindo λ como o expoente de Lyapunov, a evolução de δ ao longo do tempo pode ser aproximada por:

$$|\delta_n| \approx |\delta_0| \cdot e^{\lambda n} \tag{19}$$

Dessa forma, $\lambda > 0$ é um indicador de caos, já que indica um aumento exponencial na diferença entre as trajetórias (δ_n). Já um $\lambda = 0$, pode estar relacionado com uma estabilidade marginal (isto é, órbitas próximas não se afastam nem se aproximam, em média) no sistema e $\lambda < 0$ indica uma convergência das trajetórias.

Tirando logaritmos de ambos os lados, temos:

$$\lambda \approx \frac{1}{n} \ln \left| \frac{\delta_n}{\delta_0} \right| \tag{20}$$

Substituindo δ_n:

$$\lambda \approx \frac{1}{n} \ln \left| \frac{f^n(x_0 + \delta_0) - f^n(x_0)}{\delta_0} \right| \tag{21}$$

Quando $\delta_n \to 0$, pode-se aproximar a diferença com uma derivada:

$$\ln \left| \frac{f^n(x_0 + \delta_0) - f^n(x_0)}{\delta_0} \right| \approx (f^n)'(x_0) \tag{22}$$

$$\lambda \approx \frac{1}{n} \ln |(f^n)'(x_0)| \tag{23}$$

Se usarmos a Regra da Cadeia para composições repetidas, obtém-se:

$$(f^n)'(x_0) = f'(x_{n-1}) \cdot f'(x_{n-2}) \cdot \ \ldots \cdot f'(x_0) \tag{24}$$

De forma geral, com $x_i = f^i(x_0)$, temos:

$$(f^n)'(x_0) = \prod_{i=0}^{n-1} f'(x_i) \tag{25}$$

Pela propriedade do produto de logaritmos:

$$\lambda \approx \frac{1}{n} \sum_{x=0}^{n-1} \ln |f'(x_i)| \tag{26}$$

Assim, o expoente de Lyapunov é dado por:

$$\lambda = \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} \ln |f'(x_i)| \tag{27}$$

A finalidade do expoente nesse projeto é estimar um horizonte de previsibilidade, ou seja, o momento até o qual previsões sobre o comportamento do sistema ainda são viáveis antes que o sistema se torne caótico. Como λ é uma frequência, esse período de previsibilidade T pode ser obtido calculando o inverso do expoente (desde que $\lambda > 0$; caso contrário, o sistema não é caótico e, teoricamente, a previsibilidade é infinita):

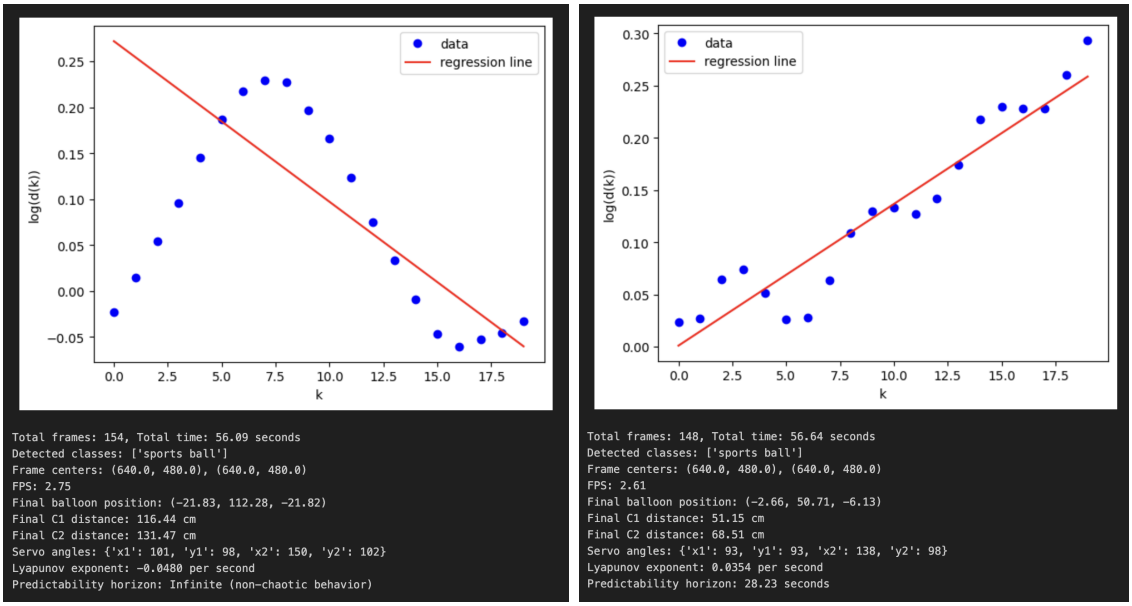
$$T = \frac{1}{\lambda} \tag{28}$$

Conforme a Equação 27, a determinação do expoente exigiria computar infinitos pontos. Como isso é inviável, existem alguns alguns algoritmos que nos permitem realizar estimativas de seu valor com menor custo computacional. Dentre eles, destacam-se os algoritmos de Wolf (WOLF *et al.*, 1985) e Rosenstein (ROSENSTEIN; COLLINS; LUCA, 1993). Nesse projeto, optou-se por utilizar o algoritmo de Rosenstein por meio da biblioteca Nolds, disponível na linguagem Python.

Ademais, vale destacar que o cálculo do expoente deve ser feito para um determinado eixo de movimentação. Para definir qual o melhor eixo de movimentação a ser usado, foi utilizado o *Principal Component Analysis* (PCA), um algoritmo de *machine learning* de redução de dimensionalidade capaz de determinar o eixo que melhor distribui um certo conjunto de dados (ARAUJO; COELHO, 2009). No caso do nosso projeto, o PCA é capaz de encontrar o eixo que melhor representa a trajetória do balão, tornando o cálculo do expoente de Lyapunov muito mais preciso do que seria caso fossem usados os eixos x, y ou z individualmente. A implementação foi feita usando a biblioteca Scikit-learn da linguagem Python.

Com isso, foi possível integrar ao protótipo um mecanismo que permite estimar por quanto tempo pode-se prever a posição de um balão com confiança. Por fim, após o cálculo do expoente, é fornecido também um gráfico da evolução do erro ao longo do tempo para melhor visualização do sistema (Figura 21 abaixo).

Figura 21 – Gráficos da separação de trajetórias ao longo do tempo.

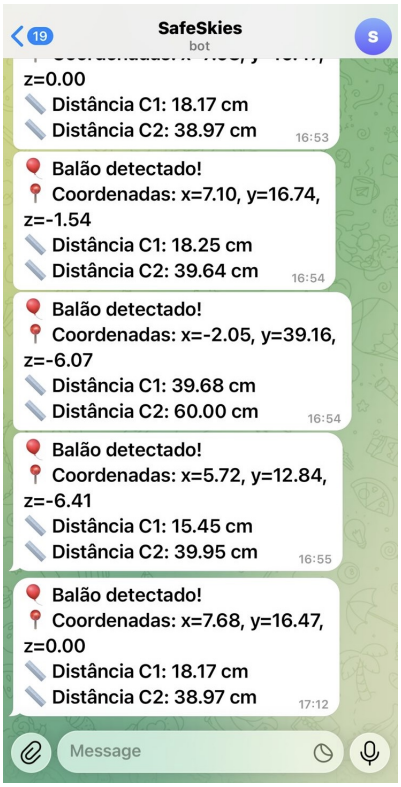


Fonte: Imagem de autoria própria obtida através do uso da biblioteca Nolds.

4.6.4 Alerta automático

O próximo passo metodológico foi a implementação de um sistema de alertas automatizado, cuja finalidade é emitir avisos às autoridades locais ao detectar balões na região, de modo a não só alertar sobre a sua presença mas também enviar informações sobre a sua localização. Para isso, optou-se por utilizar o Telegram, tanto pela facilidade de implementação do código quanto pela sua gratuitidade — diferente de outras plataformas, como WhatsApp e Messenger, que cobram taxas por mensagem enviada. A implementação do sistema foi feita utilizando a biblioteca Requests, da linguagem Python, que se comunica com a API do Telegram para realizar o envio da mensagem de alerta (exemplo na Figura 22 abaixo).

Figura 22 – Exemplo de alertas de avistamento de balão usando o Telegram.



Fonte: Imagem de autoria própria.

Vale lembrar que todos os programas descritos nesta seção estão disponíveis no repositório do projeto, nas pastas *initial_prototype* e *working_directory*, acessíveis pelos links: ⟨https://github.com/laraeleo/SafeSkies/tree/main/initial_prototype⟩ e ⟨https://github.com/laraeleo/SafeSkies/tree/main/working_directory⟩.

4.7 Perspectivas futuras

Primeiramente, pretende-se adaptar o protótipo existente para execução em ambientes reais, sujeitos a intempéries climáticas e com câmeras distantes entre si para melhor cálculo de posicionamento. Para isso, o Arduino Uno e o computador serão substituído por dois microcontroladores Raspberry Pi, cada um próximo de uma câmera. Eles serão responsáveis por executar o código Python e movimentar os servo-motores. Ademais, um terceiro componente — seja ele um terceiro Raspberry Pi ou um servidor hospedado na nuvem — será responsável por coletar os ângulos dos dois outros microcontroladores e realizar o cálculo de posicionamento, de vetores de movimentação, e previsões de deslocamento.

Após a implementação dessa funcionalidade, serão realizados testes para verificar o erro nos cálculos em função da distância entre as câmeras, de modo a encontrar a distância ideal para o seu posicionamento. Além disso, será avaliada qual a distância máxima de detecção com as câmeras atuais. Assim, caso verifique-se que as câmeras atuais são capazes de detectar balões à distância com boa resolução, as câmeras em questão serão mantidas; caso contrário, câmeras de maior resolução serão compradas para melhorar a precisão das detecções.

Em seguida, pretende-se aprimorar o sistema de previsão atual. Para isso, serão incluídas bases de dados meteorológicas — como a Open Weather Map ou a Meteostat — para considerar fatores como vento, além dos vetores de velocidade obtidos, para determinar a trajetória do balão e seu potencial local de queda. Pretende-se implementar essa funcionalidade usando campos vetoriais, de modo que não só seja possível prever deslocamentos futuros, como também determinar um raio de onde o balão pode ter sido lançado. Assim, uma colaboração com as autoridades poderá prevenir acidentes e encontrar locais de soltura e possíveis responsáveis por essas atividades criminosas.

Posteriormente, o protótipo será instalado em uma unidade de conservação ambiental ou em um aeroporto, locais com grande número de quedas de balões. Dessa forma, será finalmente possível comprovar se o projeto foi de fato bem sucedido ao localizar e prever quedas de balões na região, evitando acidentes e incêndios. O local em que as câmeras serão instaladas ainda será definido, mas diálogos estão em andamento com os parques estaduais do Juquery e do Jaraguá, no estado de São Paulo.

Para concluir a metodologia, os avistamentos de balões feitos pelos habitantes locais serão comparados às sinalizações enviadas pelo protótipo às autoridades de modo a avaliar possíveis incongruências. Caso tenha ocorrido alguma queda de balão no local, também será verificado se o local de queda previsto pelo protótipo estava correto. Com isso, será possível concluir se o protótipo está funcionando perfeitamente ou se será necessário realizar alterações para aprimorar o seu funcionamento. Se não for encontrado nenhum problema, a metodologia será considerada finalizada.

5 Resultados Parciais

Apesar de o projeto ainda estar em fase de desenvolvimento, alguns resultados relevantes já foram obtidos, e estão descritos a seguir.

5.1 Treinamento do modelo

Conforme apresentado na Metodologia, a análise de desempenho realizada por meio da utilização de uma matriz de confusão revelou que o modelo treinado utilizando o YOLOv10s apresentou excelente desempenho na detecção de balões em imagens, alcançando uma precisão de 94%, *recall* de 92%, *F1 Score* de 93% e mAP@0.5 de 96%.

Já com relação à detecção em vídeo, os testes realizados com processamento assíncrono indicaram que o modelo foi capaz de rastrear os balões com sucesso, com elevadas velocidades de processamento e acurácia. Os testes em tempo real serão realizados em breve em ambientes como aeroportos ou áreas de proteção ambiental.

Portanto, o fato de todos os indicadores se encontrarem acima de 90%, mesmo com a utilização de diversas fotos de baixa resolução, indica que o modelo apresentou uma quantidade de erros reduzida, o que permitiu que este passo metodológico fosse considerado concluído.

5.2 Protótipo funcional

Foram realizados testes iniciais para avaliar o funcionamento do protótipo e do cálculo de parâmetros, os quais apresentaram grande sucesso. O sistema foi capaz de detectar bolas de tênis com baixa latência — usando um modelo pré-treinado do YOLOv10 para fins de testes — e segui-las usando o sistema *pan* e *tilt* controlado por um PID. Além disso, as equações foram capazes de calcular a posição das bolas com precisão, o expoente de Lyapunov foi calculado de forma satisfatória e os alertas foram enviados assim que a detecção ocorreu.

Pretende-se realizar ainda novos testes para quantificar o erro das medidas em função da distância entre as câmeras (já que ele tende a diminuir conforme a distância entre as câmeras aumenta) e a distância máxima de detecção com as câmeras atuais.

Assim, será possível estimar a aplicabilidade real do protótipo e suas possíveis limitações, além de diagnosticar a possível necessidade de uma nova câmera de maior resolução.

6 Conclusão

Considerando o estágio atual do projeto, pode-se afirmar que o protótipo ainda está em desenvolvimento, de modo que ainda é necessário realizar novos testes e desenvolver novos sistemas para que a metodologia possa ser finalizada. Contudo, as etapas que foram concluídas até o presente momento — treinamento do modelo, inferências em vídeo, cálculo de parâmetros e protótipo funcional — apresentaram excelentes resultados. Assim, acredita-se que foi demonstrada a eficácia parcial da hipótese proposta, e sua eficácia total poderá ser averiguada com os testes em ambientes reais a serem realizados nos próximos meses.

Referências

AGHDAM, H. H.; HERAVI, E. J. Convolutional neural networks. In: *Guide to Convolutional Neural Networks*. [S.l.]: Springer, 2017. p. 85–130. Citado 2 vezes nas páginas 19 e 24.

Amazon Web Services. *What is Computer Vision?* 2023. Disponível em: ⟨https://dante.pro/aws⟩. Acesso em: 17 out. 2023. Citado na página 21.

Amazon Web Services. *How Object Detection Works*. 2024. Disponível em: ⟨https://dante.pro/awsdocs⟩. Acesso em: 18 fev. 2024. Citado na página 21.

ARAUJO, W. O. de; COELHO, C. J. Análise de componentes principais (pca). *University Center of Anápolis, Annapolis*, 2009. Citado na página 51.

Arduino. *Arduino Uno Rev3*. 2025. Disponível em: ⟨https://dante.pro/arduinouno⟩. Acesso em: 20 mai. 2025. Citado na página 29.

AXIMOFF, I.; BARRETO, L.; KURTZ, B. Ações cooperativas para prevenção e combate a incêndios florestais em área protegida urbana na cidade do Rio de Janeiro. *Biodiversidade Brasileira - BioBrasil*, v. 10, p. 96–109, 2020. Citado na página 10.

BARROSO, M. F. Cálculo do maior expoente de lyapunov usando estimador recursivo com fator variável. *Simpósio brasileiro de automação inteligente*, 2019. Citado na página 49.

BERTONCELLO, M.; WEE, D. Ten ways autonomous driving could redefine the automotive world. *McKinsey & Company*, v. 6, 2015. Citado na página 20.

BRANDIZZI, L. *Visão computacional: O que é? Como funciona?* 2020. Disponível em: ⟨https://dante.pro/serpro⟩. Acesso em: 14 ago. 2024. Citado na página 22.

BRANDON, K.; FONSECA, G. A. B. d.; RYLANDS, A. B.; SILVA, J. M. C. d. Conservação brasileira: desafios e oportunidades. *Megadiversidade*, v. 1, n. 1, p. 7–13, 2005. Citado na página 11.

ĈADÍK, M. Perceptual evaluation of color-to-grayscale image conversions. *Computer graphics forum*, v. 27, n. 7, p. 1745–1754, 2008. Citado na página 23.

CMR College of Engineering & Technology. *Digital Image Representation*. 2024. Disponível em: ⟨https://dante.pro/pixel⟩. Acesso em: 13 ago. 2024. Citado na página 20.

Conservation International. *Biodiversity Hotspots: Targeted investment in nature's most important places*. 2024. Disponível em: ⟨https://dante.pro/hotspots⟩. Acesso em: 14 ago. 2024. Citado na página 11.

DECEA. *DECEA alerta sobre os riscos da soltura de balões*. 2023. Disponível em: ⟨https://dante.pro/riscosbaloes⟩. Acesso em: 29 set. 2023. Citado na página 9.

EUFRASIO, A. R.; OLIVEIRA, A.; FERNANDES, C.; VALENTE, C.; SANTOS, A.; SANTOS, D. M. C. Os impactos causados por queimadas: uma análise da abordagem em livros didáticos do ensino regular. In: *Encontro Latino Americano de Iniciação Científica*. [S.l.: s.n.], 2022. Citado na página 10.

Faculdade de Computação da Universidade Federal de Uberlândia. *Representação da Imagem Digital*. 2014. Disponível em: ⟨https://dante.pro/facom⟩. Acesso em: 13 ago. 2024. Citado na página 21.

Fantástico. *Veja a trajetória do balão que causou prejuízos e deixou bairros sem energia em SP*. 2024. Disponível em: ⟨https://dante.pro/balaozonaleste⟩. Acesso em: 20 out. 2024. Citado na página 13.

FERMINO, F. *et al. Estudo comparativo de métodos de sintonia de controladores PID*. Tese (Doutorado) — UNIVERSIDADE DE SÃO PAULO, 2014. Citado na página 48.

FILATECH. *Heavy Duty Pan&Tilt For FPV*. 2019. Disponível em: ⟨https://www.thingiverse.com/thing:3050358⟩. Acesso em: 9 mai. 2025. Citado na página 46.

Força Aérea Brasileira. *DECEA alerta sobre os riscos da soltura de balões*. 2024. Disponível em: ⟨https://dante.pro/avistamentos2024⟩. Acesso em: 14 ago. 2024. Citado na página 12.

G1 Campinas e Região. *Viracopos registra média de uma queda de balão a cada oito dias em área interna do aeroporto e soma chega a 76% do total de 2022*. 2023. Disponível em: ⟨https://dante.pro/viracopos⟩. Acesso em: 05 set. 2023. Citado na página 12.

G1 SP. *Levantamento mostra que incêndio no Parque do Juquery consumiu 53% da área verde; veja antes e depois*. 2021. Disponível em: ⟨https://dante.pro/juqueryg1⟩. Acesso em: 13 ago. 2024. Citado na página 10.

G1 SP. *Balão arrasta carro, ergue moto, atinge rede elétrica e cai sobre imóveis e creche em SP; VÍDEO*. 2024. Disponível em: ⟨https://dante.pro/noticiabalao⟩. Acesso em: 13 ago. 2024. Citado na página 13.

GLOBO, O. *Companhias aéreas estimam em R$ 15 mi prejuízo com incidente em Congonhas*. 2022. Disponível em: ⟨https://oglobo.globo.com/economia/noticia/2022/10/companhias-aereas-estimam-em-r-15-mi-prejuizo-com-incidente-em-congonhas.ghtml⟩. Acesso em: 19 ago. 2025. Citado na página 12.

Google. *Google Colaboratory*. 2024. Disponível em: ⟨https://dante.pro/colab⟩. Acesso em: 22 out. 2024. Citado na página 33.

Google Cloud. *What is deep learning?* 2024. Disponível em: ⟨https://dante.pro/deeplearning⟩. Acesso em: 21 out. 2024. Citado na página 16.

Governo de São Paulo. *Incêndios florestais: autuações contra balões saltam 135% em SP*. 2024. Disponível em: ⟨https://dante.pro/dados2024⟩. Acesso em: 13 ago. 2024. Citado na página 14.

IBM. *What is artificial intelligence (AI)?* 2024. Disponível em: ⟨https://dante.pro/ibm⟩. Acesso em: 13 ago. 2024. Citado na página 15.

ISLAM, M.; CHEN, G.; JIN, S. An overview of neural network. *American Journal of Neural Networks and Applications*, Science Publishing Group, v. 5, n. 1, p. 7–11, 2019. Citado na página 20.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, p. 436–444, 2015. Citado na página 19.

LECUN, Y.; BOTTOU, L.; ORR, G.; MüLLER, K. *Neural Networks: Trick of the Trade*. Heidelberg: Springer Berlin, 2002. 9-50 p. Citado na página 19.

LIANG, J. A review of the development of YOLO object detection algorithm. *Applied and Computational Engineering*, v. 71, p. 39–46, 08 2024. Citado na página 29.

MARQUES, M. Z. de S.; GARCIA, C. M. Os riscos causados pelo lancamento de balões não tripulados: Um olhar sobre a segurança da aviação brasileira. *Revista Brasileira de Aviação Civil & Ciências Aeronáuticas*, v. 1, n. 2, p. 83–111, 2021. Citado na página 12.

MATOS, D. S.; SANTOS, C.; CHEVALIER, D. Fire and restoration of the largest urban forest of the world in Rio de Janeiro City, Brazil. *Urban Ecosystems*, v. 6, p. 151–161, 2002. Citado na página 11.

MENINO, J. d. M. Perigo baloeiro: aspectos culturais e regulamentares da atividade no brasil. *Ciências Aeronáuticas-Unisul Virtual*, 2019. Citado na página 15.

Metrópoles. *Na madrugada, tráfico usa balão e foguetório em homenagem a "Rabicó"*. 2024. Disponível em: ⟨https://dante.pro/balaotrafico⟩. Acesso em: 20 out. 2024. Citado na página 14.

Ministério dos Transportes, Portos e Aviação Civil. *Soltar balão é crime e põe em risco o espaço aéreo*. 2018. Disponível em: ⟨https://dante.pro/ministerio⟩. Acesso em: 29 nov. 2023. Citado na página 13.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. *Sistemas inteligentes - Fundamentos e aplicações*, v. 1, n. 1, p. 32, 2003. Citado na página 15.

NAQA, I. E.; MURPHY, M. J. What is machine learning? In: *Machine Learning in Radiation Oncology*. [S.l.]: Springer, 2015. p. 3–11. Citado na página 16.

Nutrição FSP. *Festas juninas: origem e celebração*. 2023. Disponível em: ⟨https://dante.pro/origemdafesta⟩. Acesso em: 16 set. 2023. Citado na página 9.

OpenCV. *OpenCV AI*. 2025. Disponível em: ⟨https://dante.pro/opencv⟩. Acesso em: 20 mai. 2025. Citado na página 29.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. *ArXiv e-prints*, 2015. Citado na página 24.

Presidência da República. *Lei nº 9.605, de 12 de fevereiro de 1998*. 1998. Disponível em: ⟨https://dante.pro/lei9605⟩. Acesso em: 25 ago. 2023. Citado na página 10.

QUEIROZ, J. E. R. de; GOMES, H. M. Introdução ao processamento digital de imagens. *Revista de Informática Teórica e Aplicada*, v. 13, p. 11–42, 2006. Citado na página 23.

RAUBER, T. W. Redes neurais artificiais. *Universidade Federal do Espírito Santo*, v. 29, 2005. Citado na página 16.

Revista Veja Rio. *Ameaça no ar: balões ganham impulso com apoio de tráfico e milícia*. 2023. Disponível em: ⟨https://dante.pro/baloestrafico⟩. Acesso em: 20 out. 2024. Citado na página 14.

ROSENSTEIN, M. T.; COLLINS, J. J.; LUCA, C. J. D. A practical method for calculating largest lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, Elsevier, v. 65, n. 1-2, p. 117–134, 1993. Citado na página 51.

SCHMITT, G. K. O risco baloeiro na aviação brasileira: o problema causador do rebaixamento da classificação de segurança do espaço aéreo brasileiro. *Ciências Aeronáuticas - Unisul Virtual*, 2018. Citado na página 12.

SHAHA, M.; PAWAR, M. Transfer learning for image classification. In: IEEE. *2018 second international conference on electronics, communication and aerospace technology (ICECA)*. [S.l.], 2018. p. 656–660. Citado na página 24.

SILVA, R. Inteligência artificial. *Enciclopédia da Conscienciologia*, 2013. Citado na página 15.

SP Notícias. *Parque do Juquery: as lições deixadas pelo combate ao incêndio que durou 4 dias*. 2024. Disponível em: ⟨https://dante.pro/juquery⟩. Acesso em: 13 ago. 2024. Citado na página 10.

TAMBASCO, F. *Train-yolov8-custom-dataset-step-by-step-guide*. 2023. Disponível em: ⟨https://dante.pro/predictvideo⟩. Acesso em: 30 nov. 2024. Citado na página 40.

TENÓRIO, A. *Festa Junina: Conheça a história*. 2022. Disponível em: ⟨https://dante.pro/historiadafesta⟩. Acesso em: 22 set. 2023. Citado na página 9.

TV Globo. *VÍDEO: Morador filma queda de balão momentos antes de galpão pegar fogo na Zona Oeste de SP*. 2024. Disponível em: ⟨https://dante.pro/noticiagalpao⟩. Acesso em: 13 ago. 2024. Citado na página 14.

Ultralytics. *Machine Learning Best Practices and Tips for Model Training*. 2024a. Disponível em: ⟨https://dante.pro/ultralyticsdocs⟩. Acesso em: 30 nov. 2024. Citado na página 33.

Ultralytics. *YOLO Performance Metrics*. 2024b. Disponível em: ⟨https://dante.pro/yolometrics⟩. Acesso em: 30 nov. 2024. Citado na página 38.

Universidade Federal de Alfenas. *Histologia Interativa: Tecido Nervoso*. 2022. Disponível em: ⟨https://dante.pro/tecidonervoso⟩. Acesso em: 21 out. 2024. Citado na página 16.

WALBERT, A. *Você conhece as histórias da bandeirinha, balão e fogueira de São João?* 2013. Disponível em: ⟨https://dante.pro/historiabaloes⟩. Acesso em: 22 set. 2023. Citado na página 9.

WANG, S. Artificial neural network. In: *Interdisciplinary Computing in Java Programming*. Boston, MA: Springer, 2003, (The Springer International Series in Engineering and Computer Science, v. 743). Citado na página 18.

WOLF, A.; SWIFT, J. B.; SWINNEY, H. L.; VASTANO, J. A. Determining lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, v. 16, n. 3, p. 285–317, 1985. ISSN 0167-2789. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/0167278985900119⟩. Citado na página 51.

YING, X. An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, IOP Publishing, v. 1168, n. 2, p. 022022, 02 2019. Citado na página 34.

YUILL, S.; HALPIN, H. Python. *Python releases for Windows*, Citeseer, v. 24, 2006. Citado na página 28.

ZOU, J.; HAN, Y.; SO, S.-S. Overview of artificial neural networks. In: *Artificial neural networks: methods and applications*. [S.l.]: Humana Press, 2009. p. 14–22. Citado na página 16.