LUIS FILIPE ARAÚJO DE SOUZA

Desenvolvimento de Sistemas Embarcados para Classificação de Susceptibilidade ao Fungo da Ferrugem em Folhas de Café

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 402 - Projeto de Engenharia II e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Aprovada em 04 de julho de 2025.

COMISSÃO EXAMINADORA

gov.br
Documento assinado digitalmente
RODOLPHO VILELA ALVES NEVES
Data: 04/07/2025 15:15:15-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Rodolpho Vilela Alves Neves - Orientador
Universidade Federal de Viçosa

gov.br
Documento assinado digitalmente
LEONARDO BONATO FELIX
Data: 07/07/2025 17:21:08-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Leonardo Bonato Felix - Membro
Universidade Federal de Viçosa

gov.br
Documento assinado digitalmente
VINICIUS MARTINS ALMEIDA
Data: 08/07/2025 15:45:04-0300
Verifique em https://validar.iti.gov.br

Me. Vinicius Martins Almeida – Membro
Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

Desenvolvimento de Sistemas Embarcados para Classificação de Susceptibilidade ao Fungo da Ferrugem em Folhas de Café

Luis Filipe Araujo de Souza, Rodolpho Vilela Alves Neves†

***Abstract*—This project aims to develop an embedded application to integrate and perform inference with a coffee leaf classification model, focusing on resistance to Hemileia vastatrix, the coffee rust disease. The application will be initially targeted at researchers, providing an intuitive interface that allows the analysis of leaf samples and quick results on their resistance to the disease. The project will include the integration of the trained model, the development of the embedded interface, and the evaluation of the system's usability and accuracy, aiming to facilitate research work in the field of coffee plant pathology.**

***Index Terms*—Machine Learning, Hemileia vastatrix, Coffee diseases, Embedded Systems.**

I. Introdução

A CULTURA do café exerce papel fundamental na economia de diversas regiões do mundo, especialmente no Brasil, um dos maiores produtores e exportadores globais. Contudo, a produção cafeeira enfrenta desafios significativos devido a doenças que acometem as plantas, em particular a *Hemileia vastatrix*, conhecida como ferrugem do café. Essa fitoterapia está entre as mais destrutivas para a cultura cafeeira e pode ocasionar perdas econômicas expressivas quando não detectada e tratada prontamente.

Uma das estratégias de controle mais sustentáveis e economicamente viáveis é o uso de cultivares geneticamente resistentes. Contudo, o método tradicional de seleção fenotípica é um processo lento, subjetivo e altamente dependente da avaliação de especialistas [1]. Para superar essas limitações, técnicas de Aprendizado de Máquina, especialmente a Visão Computacional, surgiram como ferramentas de alta precisão para automatizar e tornar essa análise mais objetiva. Apesar dos avanços, muitos dos modelos desenvolvidos em ambiente acadêmico, como os discutidos por Neto et al., enfrentam dificuldades para serem amplamente utilizados por pesquisadores e produtores, devido a necessidade de computadores e programas especializados, além de conhecimento técnico em Aprendizado de Máquinas para realizar inferências com os mesmos.

A raspberry-pi é um dispositivo amplamente utilizado na prototipagem de soluções que utilizam aprendizado de máquina para solução de algum problema real, seja ele na indústria ou no campo. A presença de sensores de câmera de alta qualidade nos kits desse dispositivo, além de seu poder computacional relativamente bom e um ambiente de desenvolvimento muito semelhante a aqueles que encontramos em dispositivos dedicados, fez com que o escolhessem como um dos aparelhos facilitadores para a utilização dos modelos. O outro aparelho, um dispositivo celular android, foi escolhido também pela presença de sensores de câmera com relativa qualidade, mas principalmente pela sua ampla utilização em território nacional.

Logo, este estudo detalha o desenvolvimento de dois sistemas embarcados para a classificação da ferrugem em folhas de café por meio de aprendizado de máquina. Para isso, foram treinados dois modelos distintos utilizando a técnica de transfer-learning com a rede ResNet50 [2]. O primeiro é um classificador binário treinado com o dataset CoffeeRust, cuja abordagem foi baseada no trabalho que introduziu o próprio dataset [1], projetado para diferenciar folhas suscetíveis de resistentes. O segundo, um classificador multiclasse também treinado com o dataset CoffeeRust e seguindo a mesma metodologia, foi desenvolvido para identificar diferentes níveis de severidade da doença, com base nas escalas propostas por Tamayo et al. (1995) [3]. Para aplicação prática em campo, os modelos foram integrados em um módulo Raspberry Pi acoplado a uma câmera de alta resolução, que executa ambos os classificadores, e também em um aplicativo Android, que disponibiliza o classificador binário para smartphones. Ambas as soluções contam com uma interface gráfica intuitiva, permitindo que o usuário realize análises de forma rápida e precisa diretamente no local.

II. Metodologia

A metodologia deste projeto foi estruturada em quatro etapas sequenciais. Primeiramente, realizou-se a preparação e otimização dos modelos de aprendizado de máquina previamente treinados para o uso em dispositivos com menor capacidade computacional. A segunda etapa consistiu no desenvolvimento do software responsável pela inferência, com o código sendo adaptado para as especificidades de cada sistema embarcado alvo: o módulo Raspberry Pi e a plataforma Android.

Após a validação dos programas em ambiente de desenvolvimento, a terceira etapa envolveu o deployment das soluções para os dispositivos finais. Por fim, conduziu-se uma fase de testes para avaliar a qualidade e o desempenho dos sistemas em campo. Nesta fase, foram realizadas medições para

Luis Filipe Araujo de Souza está no Departamento de Engenharia Elétrica, Universidade Federal de Viçosa, Viçosa-MG, Brazil (e-mail: luis.f.filipe@ufv.br).

Rodolpho Vilela Alves Neves† era do Departamento de Engenharia Elétrica, Universidade Federal de Viçosa.

comparar a acurácia dos modelos no ambiente embarcado com os resultados obtidos em um ambiente de desenvolvimento controlado, além de terem sido avaliadas a estabilidade e a usabilidade de ambas as aplicações.

A. Preparação dos modelos

Nesta etapa, os modelos utilizados inicialmente em um ambiente de experimentação - treinados com os dados fornecidos pelo NIAS (CoffeeRust) [1] - foram adaptados para execução em um ambiente de deployment com recursos computacionais consideravelmente limitados, como um dispositivo embarcado.

Para otimizar a performance em dispositivos com recursos limitados, os modelos de machine learning foram convertidos de seu formato original para uma versão "lite", mais leve e eficiente [4]. Essa conversão visava a dois objetivos principais: reduzir o tamanho do modelo em disco e o consumo de memória, e acelerar significativamente o tempo de inferência (ou "forward pass").

O principal método utilizado nesse processo é a quantização. Esta técnica transforma os pesos da rede neural, originalmente representados por números de ponto flutuante de 32 bits (float32), para inteiros de 8 bits (int8). Ao reduzir a representação de cada peso de 32 para 8 bits, o tamanho do modelo e a demanda computacional diminuíram drasticamente.

Embora essa mudança resultou em uma leve e controlada perda de precisão, o ganho em velocidade e eficiência foi fundamental para garantir que os modelos executem de forma responsiva no nosso hardware com capacidade restrita, viabilizando a inferência rápida em cenários de baixa potência.

B. Sistema de Inferência

O desenvolvimento do sistema de inferência está diretamente relacionado ao processo de conversão do modelo para o formato lite. Embora a escolha do framework ou ferramenta utilizada para essa conversão possa variar [4], [5], o fluxo de trabalho necessário para criar nosso sistema de inferência embarcado seguiu a metodologia descrita a seguir.

Primeiramente, foi necessário definir quais tipos de pré-processamento iriam ser aplicados aos dados de entrada, caso fossem necessários, antes de enviá-los ao modelo. Após essa etapa, desenvolvemos a interface que permite ao usuário carregar uma foto armazenada ou capturá-la diretamente no momento, transferindo-a para a memória do sistema. No caso do Raspberry-pi, o usuário tem a opção de salvar também num arquivo Json [6] um link para a imagem salva em disco, junto com o output do modelo para aquela imagem.

Com a imagem carregada na memória e devidamente pré-processada, ela é inserida no modelo para análise. A partir desse ponto, foi preciso definir como iria ser realizado o pós-processamento dos resultados. Esse pós-processamento envolve a transformação da saída probabilística do modelo em uma classificação final. No caso do modelo binário, esse resultado final da inferência é um diagnóstico da saúde da planta doente ou saudável. Já para o modelo embarcado na Raspberry-pi (modelo maior de 6 classes) o output representa um nível de resistência à ferrugem.

O fluxo completo que começa com o upload, ou registro, da imagem pelo usuário, passa pelo processamento no modelo, e termina com a obtenção da classificação desejada é chamado de inferência. A arquitetura geral desse fluxo é apresentada na Figura 1. Por fim, foi necessário o desenvolvimento de uma interface de saída que permita ao usuário visualizar o resultado gerado pelo modelo (Figura 2), garantindo que o sistema de inferência seja acessível e funcional para o público-alvo.

O sistema de classificação foi desenvolvido com duas implementações principais, visando atender a diferentes plataformas: uma para dispositivos Android e outra para o Raspberry Pi. A arquitetura subjacente de ambos os programas é fundamentalmente similar, diferenciando-se nos detalhes de implementação da interface do usuário e na integração com o hardware específico de cada plataforma.

Uma característica unificadora entre as duas versões é o pré-processamento das imagens. Em ambos os casos, as imagens são submetidas a um padronizado processo que inclui, primeiramente, um redimensionamento (resize) para se adequar às dimensões de entrada exigidas pelos modelos de inferência. Subsequentemente, as imagens são convertidas para o formato de cores RGB. É importante notar que, embora essas etapas de pré-processamento sejam realizadas de forma semelhante em ambas as plataformas, a etapa de normalização dos dados de entrada é intrínseca aos nós de entrada do próprio modelo de inferência, garantindo a consistência na alimentação dos dados.

A implementação no Raspberry Pi se destaca por incorporar uma Interface Gráfica do Usuário (GUI), que não apenas facilita a interação do usuário, mas também oferece a funcionalidade de salvar as imagens capturadas e seus respectivos resultados de classificação diretamente no dispositivo.

Além da interface gráfica, o sistema no Raspberry Pi é fundamentado em uma robusta arquitetura cliente-servidor. A principal característica dessa arquitetura é sua modularidade, projetada para desacoplar completamente a lógica de inferência da aplicação vista pelo usuário. Essa separação garante que futuras atualizações como a substituição do modelo de deep learning por uma versão mais precisa, a modificação da pipeline de pré-processamento ou até mesmo a troca do framework de inferência (atualmente TFLite [4]) possam ser realizadas inteiramente no lado do servidor. Qualquer mudança na lógica de IA não impacta ou exige qualquer alteração no cliente, garantindo manutenibilidade e flexibilidade a longo prazo.

O cliente, por sua vez, é uma aplicação interativa desenvolvida com Streamlit [7] que atua como interface para o usuário. Ele oferece dois métodos para a entrada de imagens: captura em tempo real através da câmera conectada ao Raspberry Pi (Figura 3) ou o upload de um arquivo de imagem existente (Figura 4). Após a seleção, a única responsabilidade do cliente é enviar a imagem ao servidor para processamento.

O servidor é uma aplicação FastAPI [8] que opera como o motor de inferência do sistema, hospedando dois serviços essenciais: um dedicado à inferência do modelo binário, que classifica a imagem como "suscetível" ou "resistente" à ferrugem, e outro para a inferência do modelo multi-classes, que fornece uma classificação mais granular do nível de

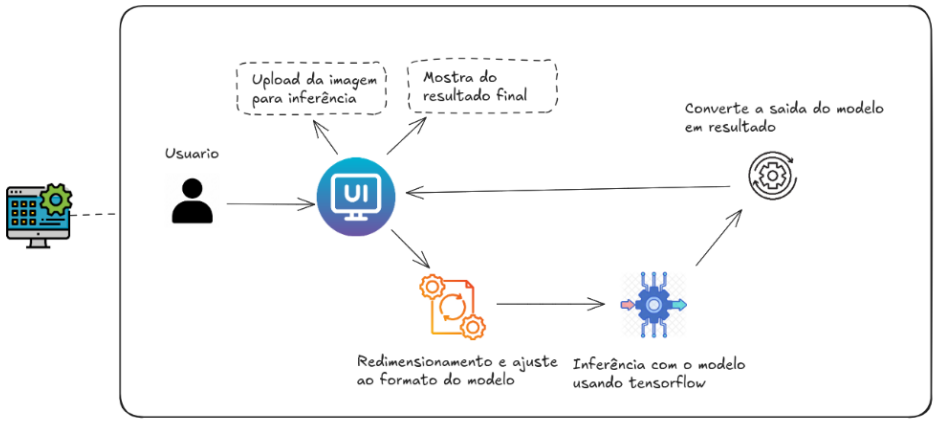

Fig. 1: Arquitetura do sistema de inferência.

Fig. 2: Inferência completa, da imagem ao resultado.

Fig. 3: Interface do sistema para captura de imagem em tempo real via Picamera.

Fig. 4: Interface do sistema para upload de arquivo de imagem.

resistência. A seleção entre os modelos é feita diretamente na interface, como mostra a Figura 5.

Um aspecto crucial do sistema no Raspberry Pi é o mecanismo de persistência de dados. Toda inferência realizada pelos modelos é persistentemente armazenada no disco do dispositivo. As imagens processadas são salvas em um diretório padrão denominado `inference_results`. Além da imagem, um arquivo json [6] é criado dentro dessa mesma pasta, contendo uma lista detalhada das inferências. Cada item dessa lista associa o nome da imagem à qual a inferência pertence com o resultado obtido. Para o modelo binário, o resultado é textual (ex: "suscetível" ou "resistente"), enquanto para o modelo multi-classes, o resultado é um número inteiro que sugere o nível de resistência para a imagem correspondente.

Fig. 5: Menu lateral para seleção do modelo de classificação (Binário ou Multiclasse).

C. Requisitos de Hardware Estimados

O sistema foi projetado com foco em acessibilidade e baixo custo, permitindo sua utilização em campo com hardware facilmente disponível.

1) Plataforma Raspberry Pi: Para a implementação completa, que inclui o servidor de inferência e a interface do cliente, recomenda-se um Raspberry Pi 4 Modelo B com, no mínimo, 2 GB de RAM. Essa configuração garante recursos suficientes para executar o servidor web (FastAPI), a aplicação interativa (Streamlit) e as inferências dos modelos TFLite de forma simultânea e responsiva. Modelos mais antigos, como o Raspberry Pi 3 B+, também podem ser utilizados, embora possam apresentar maior latência. Além da placa, são necessários um cartão microSD para o sistema operacional e armazenamento, e um módulo de câmera compatível (como o Raspberry Pi Camera Module) ou uma webcam USB.

2) Plataforma Android: A versão para Android, que executa apenas o classificador binário, é consideravelmente menos exigente. Um smartphone com sistema Android 8.0 (Oreo) ou superior, equipado com uma câmera funcional, é suficiente para a execução do aplicativo. A otimização do modelo via TFLite garante que o consumo de bateria e o processamento sejam mínimos, tornando-o viável para a maioria dos dispositivos modernos.

A otimização dos modelos através de quantização foi um fator crucial para viabilizar a execução em hardware com recursos tão limitados, alinhando-se ao objetivo de criar uma ferramenta prática e de baixo custo para pesquisadores.

D. Deployment

Após a validação rigorosa dos programas no ambiente de desenvolvimento, realizamos o processo de deployment para os sistemas embarcados. Essa etapa se mostrou crucial para garantir que os modelos e todo o sistema de inferência operarem corretamente em um dispositivo com recursos limitados, característica central deste projeto.

O deployment envolveu a transferência do programa do ambiente de desenvolvimento para o dispositivo final, considerando as diferenças de arquitetura e performance entre os dois ambientes. Enquanto no ambiente de desenvolvimento o sistema foi testado em máquinas com maior capacidade de processamento e memória, no ambiente embarcado enfrentamos e superamos restrições que exigiram otimizações específicas.

Uma das primeiras tarefas nesse processo foi garantir que o dispositivo embarcado possuísse o sistema operacional e as bibliotecas necessárias para executar o programa. Isso incluiu a instalação de dependências, a configuração de drivers para câmeras e outros periféricos, além da criação de scripts de inicialização automática para que o sistema de inferência começasse a rodar assim que o dispositivo fosse ligado.

Durante o deployment, também monitoramos o tempo de resposta do modelo para verificar se a inferência ocorria dentro de um intervalo aceitável, garantindo que a aplicação se tornasse prática e eficiente para uso em campo.

E. Testes

Nesse estágio, focamos na realização de uma série de testes que realizamos para garantir a qualidade das classificações efetuadas pelo modelo no ambiente embarcado. Esses testes abrangeram a medição da acurácia do modelo, que foi comparada entre o ambiente de desenvolvimento, onde o modelo havia sido inicialmente treinado, e o ambiente embarcado, onde o modelo foi executado com recursos limitados. Essa comparação permitiu identificar possíveis quedas no desempenho do modelo, assegurando que o sistema continuasse funcionando de forma precisa e eficaz após a transição para o dispositivo final. Realizamos também um teste de velocidade de inferência, onde medimos o tempo que o modelo tomou para concluir uma forward pass com o framework TFLite [4].

Além disso, dedicamos atenção especial à estabilidade e usabilidade do sistema de inferência. A estabilidade foi verificada através de testes contínuos para garantir que o sistema operasse sem falhas ou quedas de desempenho durante períodos prolongados de operação. Para isso mantivemos o servidor de inferência em funcionamento durante um dia inteiro e a noite fora utilizado junto ao cliente. Já a usabilidade foi avaliada com base na interação do usuário com a interface do sistema, assegurando que o processo de upload de imagens e a obtenção de resultados fossem intuitivos, rápidos e eficientes, atendendo às necessidades do usuário final, como pesquisadores da área de fitopatologia. A realização desse teste foi feita por duas pessoas leigas que testaram a interface de cliente desenvolvida para o projeto. Esses testes foram fundamentais para assegurar que o sistema se tornasse não apenas tecnicamente preciso, mas também funcional e adequado para o uso em campo.

TABLE I: Comparação da acurácia dos modelos.

Modelo	Versão	Acurácia
Classificador Binário	TFLite	94,25%
	Full-size	94,54%
Classificador Multiclasse	TFLite	76,16%
	Full-size	76,34%

TABLE II: Tempo médio de inferência dos modelos em formato lite.

Modelo	Binário	multi-classes
Velocidade	80.02 ms	81.98 ms

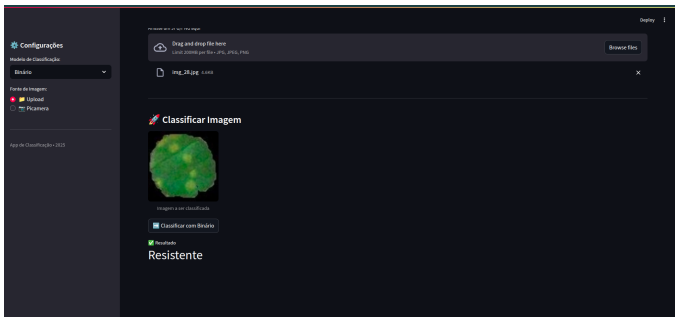

Fig. 6: Apresentação do resultado ("Resistente") na interface após a classificação de uma imagem.

III. Resultados e Discussões

Os resultados dos testes relacionados à qualidade e velocidade de inferência dos modelos, incluindo a comparação da acurácia entre os modelos quantizados e os modelos de precisão total, são apresentados na Tabela I e II respectivamente.

Nos testes de usabilidade e estabilidade também obtivemos resultados positivos, onde o sistema se provou fácil e intuitivo de ser utilizado por pessoas inexperientes com tecnologia. A Figura 6 ilustra a tela de resultado após uma inferência bem-sucedida. Além disso, mesmo após mantermos o servidor de inferência em funcionamento por um dia inteiro, não tivemos nenhum problema de conexão com o mesmo para a realização das inferências no fim do dia. Isso mostra que a solução pode ser utilizada por longos períodos de tempo, incluindo períodos de ociosidade, como é o caso de experimentos feitos em laboratório e até mesmo no campo.

Portanto, este trabalho resultou no desenvolvimento de um sistema robusto que capacita pesquisadores e colaboradores do NIAS a realizar inferências de forma ágil em amostras de folhas de café preparadas em ambiente laboratorial.

O objetivo final foi plenamente alcançado, proporcionando uma ferramenta eficiente para a detecção da ferrugem. Como uma extensão potencial, ainda em avaliação, exploramos a implementação de um modelo de segmentação de regiões, conforme referência [9]. Essa funcionalidade permitiria o pré-processamento automatizado das folhas de café, convertendo-as para um formato circular compatível com a entrada do modelo pré-treinado, otimizando significativamente o desempenho do sistema em cenários futuros.

Além da funcionalidade principal, asseguramos que o sistema fosse intuitivo e de fácil utilização. Os testes de usabilidade confirmaram que o processo de upload de imagens e a obtenção de resultados são simples e diretos, atendendo às expectativas de pesquisadores e usuários finais.

Um aspecto fundamental avaliado foi a acurácia do modelo quantizado em comparação ao modelo original. Conforme detalhado na Tabela I, observamos uma leve redução na precisão do modelo quantizado. No entanto, essa diferença não se mostrou significativa a ponto de comprometer a eficácia geral do sistema. Os resultados demonstraram que a quantização foi um sucesso, permitindo que o modelo operasse eficientemente em ambientes de recursos limitados, como o Raspberry Pi, mantendo um alto nível de acurácia. A velocidade de inferência dos modelos quantizados, apresentada na Tabela II, validou a escolha pela quantização, evidenciando ganhos substanciais de desempenho sem perdas inaceitáveis de precisão.

Em suma, os testes confirmaram a viabilidade e a eficácia do sistema proposto. Ele não apenas cumpriu os requisitos de precisão e agilidade nas inferências, mas também provou ser uma ferramenta estável e fácil de usar, pronta para auxiliar no trabalho de campo e pesquisa do NIAS.

IV. Considerações Finais

Desenvolvemos um sistema embarcado inovador para classificar folhas de café quanto à resistência à ferrugem (*Hemileia vastatrix*). Este sistema emprega técnicas de aprendizado de máquina e um modelo quantizado otimizado, feito sob medida para operar eficientemente em dispositivos com recursos limitados. Nosso objetivo principal era simplificar o processo de inferência em campo, entregando uma ferramenta intuitiva e eficaz para pesquisadores e profissionais da fitopatologia cafeeira.

Além da classificação automática, garantimos que o sistema oferecesse alta usabilidade e precisão em um ambiente embarcado, mesmo com as inerentes limitações de hardware e a demanda por respostas rápidas e confiáveis. Conforme confirmado por nossos testes, a leve redução na acurácia do modelo quantizado, em comparação ao modelo original, não comprometeu o desempenho geral. O sistema manteve-se dentro dos parâmetros aceitáveis para a aplicação prática, validando a eficácia da otimização para ambiente embarcado.

Para o futuro, uma possível extensão do projeto poderia incluir a implementação de um detector de regiões de praga nas folhas ou outras técnicas de segmentação de imagem. Isso permitiria um pré-processamento ainda mais avançado das amostras, elevando a precisão das inferências e expandindo as capacidades do sistema. Acreditamos que este sistema contribuirá significativamente para o manejo de doenças no cultivo de café, ajudando a reduzir perdas econômicas e a otimizar as práticas agrícolas.

References

[1] e. a. Neto, Antônio Teixeira Santana, "Coffeerust: An image database for classification of coffea arabica resistance to coffee leaf rust."

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] P. Tamayo, F. Vale, L. Zambolim, G. Chaves, and A. Pereira, "Resistencia do catimor a ferrugem e virulencia de ra cas fisiologicas de hemileia vastatrix berk. & br," *Fitopatología Brasileira, Brasilia*, vol. 20, no. 4, pp. 572–576, 1995.

[4] Google, "Tensorflow lite framework," acessado em: [Data de Acesso]. [Online]. Available: https://www.tensorflow.org/lite

[5] ——, "Litert," acessado em: [Data de Acesso]. [Online]. Available: https://ai.google.dev/edge/litert?hl=pt-br

[6] W3Schools, "Json introduction," acessado em: [Data de Acesso]. [Online]. Available: https://www.w3schools.com/js/js_json_intro.asp

[7] "Streamlit," acessado em: [Data de Acesso]. [Online]. Available: https://streamlit.io/

[8] "Fastapi," acessado em: [Data de Acesso]. [Online]. Available: https://fastapi.tiangolo.com/

[9] Y. Mo, Y. Wu, X. Yang, F. Liu, and Y. Liao, "Review the state-of-the-art technologies of semantic segmentation based on deep learning," *Neurocomputing*, vol. 493, pp. 626–646, 2022.