# Sistema Integrado em Domótica para Gerenciamento de Inventários do Lar via Aprendizado de Máquina

**Camila dos Anjos Caetano, Filipe Anestidis Meneses, Gabriel Rufino de Barros Steffen, Mateus Reinhold Timm, Paulo Daniell Affonso Converso**

Departamento de Engenharia de Computação
Universidade Anhembi Morumbi (UAM)
São Paulo – SP – Brasil – 2023

***Abstract.*** *Home automation provides connectivity, security, financial savings, and comfort, however, the lack of solutions for managing home inventories can cause excessive purchases, out-of-stock products, as well as unwarranted financial and material waste. Combining a device capable of object recognition through machine learning and neural networks, sensors, a camera, and a household items management application in the form of virtual inventories, the solution hereby proposed and tested by this project shows promising results on facilitating home organization and assisting on avoiding food waste and with other household products.*

**Keywords** — Domotics. SmartHome. Organization. Machine Learning. Automation.

***Resumo.*** *A domótica proporciona conectividade, segurança, economia e conforto, porém, a falta de soluções para o gerenciamento de inventários do lar pode desencadear compras em excesso, falta de produtos, e o desperdício financeiro e material. Combinando um dispositivo capaz de reconhecimento de objetos através de aprendizado de máquina e o treinamento de redes neurais, sensores, uma câmera e um aplicativo de gerenciamento de itens domésticos em forma de inventários virtuais, esta solução tem como premissa facilitar a organização domiciliar, oferecendo um sistema de auxilio de decisão para evitar desperdício de alimentos e de outros produtos do lar.*

**Palavras-Chave** — Domótica. *SmartHome*. Organização. Aprendizado de Máquina. Automação.

## 1. Introdução

A evolução tecnológica na sociedade contemporânea tem transcendido os limites convencionais, remodelando drasticamente a maneira como as interações com o ambiente são feitas, sobretudo dentro dos lares. A fusão entre a domótica, que se diz respeito à aplicação tecnológica para automatização residencial, e os avanços da indústria 4.0, caracterizada na domótica pela integração de tecnologias digitais como visão computacional e aprendizado de máquina, tem sido a força propulsora de uma transformação inigualável na administração da residência e nas rotinas cotidianas.

Esta união não apenas introduz novos paradigmas, mas também redefine a maneira como se é vivenciado o espaço doméstico, permeando cada aspecto da vida diária. A capacidade de controlar a iluminação, temperatura, segurança e eletrodomésticos por meio de sistemas automatizados não só eleva a comodidade e bem-estar do dia a dia

a novos patamares, mas também a economia, a eficiência e gestão inteligente dos recursos residenciais, criando um ambiente mais acessível, adaptativo e sustentável.

Apesar destes notáveis avanços neste cenário tecnológico, uma área crucial que permanece carente de soluções eficientes e acessíveis é o gerenciamento de inventários domésticos[1], como armários, despensas ou geladeiras. Esta lacuna torna-se evidente diante das práticas ainda prevalentes, como o método manual de listar itens de compras[2], que resulta em compras excessivas ou na falta de produtos essenciais, resultando em perdas financeiras e descontrole do orçamento residencial, e diante das soluções em domótica ofertadas atualmente pelo mercado[3].

Neste setor da indústria, então, há um amplo espaço para inovação[4]. Através da incorporação de um dispositivo com uma câmera, algoritmos de redes neurais treinados em cima do reconhecimento de imagem de alimentos e uma plataforma *web* capaz de gerenciar o inventário do lar e recomendar receitas, foi criado um sistema integrado em domótica. Este sistema permite o usuário a instalar o dispositivo em um inventário do lar e gerenciar em tempo real os alimentos contidos nele, tanto localmente quanto remotamente, como, por exemplo, durante as compras do mercado, trazendo benefícios tangíveis para as famílias e avançando um passo crucial na direção de práticas mais sustentáveis e a conscientização de uma gestão eficiente dos bens domésticos.

## 2. Revisão Bibliográfica

A domótica converge a partir dos campos da engenharia, da programação, de telecomunicações e design de sistemas para criar ambientes inteligentes capazes de automatizar o lar através de dispositivos e sensores com o intuito de poupar o usuário de tarefas repetitivas e promover conforto e segurança. Esta indústria conta com uma das maiores expansões tecnológicas, com um crescimento esperado de 25,70% em 2025, atingindo os R$ 896,32 bilhões globalmente, e R$ 11,22 bilhões no Brasil[4]. Esta expansão é um testemunho da expectativa dos consumidores por lares mais conectados e da inovação na automação do lar, ambos estimulados pela popularização da domótica.

As soluções de domótica abrangem produtos e soluções como iluminação inteligente capaz de ajustar a intensidade, cor e programação de horários para funcionamento; sistemas de segurança integrados à câmeras de vigilância, sensores de movimentos, fechaduras inteligentes e alarmes; termostatos inteligentes que ajustam a temperatura do ambiente; assistentes virtuais domésticos controlados por voz e responsáveis na execução de tarefas de rotina, respostas a perguntas do usuário e o controle de outros diversos dispositivos de domótica; sistemas de irrigação; gestão de energia e eletrodomésticos inteligentes. Todas estas soluções têm como princípio as preferências e a privacidade do usuário, a automação e a conectividade do lar, o gerenciamento remoto, a economia financeira e a eficiência energética.

Apesar da inovação e crescimento no setor, o gerenciamento de inventários, como armários, despensas e geladeiras, e o de alimentos e produtos de higiene e de limpeza, ainda se dá, em maior parte, através de métodos convencionais e manuais[2], como planilhas ou listas físicas, métodos dos quais são suscetíveis ao erro humano, a negligência, a ineficiência e ao esquecimento, provocando as compras excessivas de itens já em estoque ou fora de iminência de expiração, criando desperdício e gasto desnecessários[6]. Ao se programar para as compras, se economiza tempo gasto em mercado e, consequentemente, reduz-se o gasto de dinheiro em compras não planejadas

e de impulso[7], chegando a possíveis ganhos de até 63,03%[8], e o monitoramento da quantidade e qualidade de alimentos e produtos de limpeza ou higiene torna possível evitar o desperdício em até 43%[9].

Analisando as soluções ofertadas no mercado global, nota-se uma escassez de soluções que atendam as necessidades e o poder de compra de usuários conscientes sobre a gestão de inventários do lar, com preços variando entre R$1.234 e R$34.566 nas soluções DIY (*Do It Yourself*, ou "faça você mesmo"), e R$740 e R$49.381 nas soluções de marcas estabelecidas[3]. Estas soluções, em maior parte, se dão através de aplicativos de listas de compras, empacotadores inteligentes com recipientes imutáveis e assistentes de voz que providenciam receitas baseadas em busca da internet. Uma solução em forma de geladeiras inteligentes, a Samsung Family Hub Refrigerator, permite observar e identificar através de uma câmera os alimentos guardados, com um preço de RS$ 25.690, se tornando uma opção inacessível e incapaz de se adaptar a outros inventários.

Para superar estes desafios e auxiliar a gestão do lar, o projeto propõe a solução em forma de um dispositivo eletrônico, sensores e uma câmera capaz de utilizar a visão computacional feita com aprendizado de máquina e redes neurais[10] para identificar o alimento e o inserir em um gerenciador de inventários virtual acessível visualmente através de uma plataforma *web* para o usuário tanto localmente quanto remotamente.

## 3. Metodologia

Através da pesquisa exploratória, foi determinado com agilidade e eficiência um tema central para o projeto baseado em um problema e possíveis soluções relacionadas a três tópicos relevantes para o projeto: automação, cotidiano e economia. A automação envolve o desenvolvimento de *hardware* e *software*, algo central para todos os membros do projeto, enquanto analisar o cotidiano permite ao engenheiro identificar necessidades reais em soluções tecnológicas, e as tornarem economicamente acessíveis permite o usuário se beneficiar de uma vida financeiramente equilibrada. A combinação destes três tópicos resulta em domótica e a automação do lar e de tarefas rotineiras em uma *smarthome* (lar inteligente).

Uma solução capaz de processar as imagens dos produtos capturadas foi fundamental para o projeto. Soluções de processamento em nuvem foram consideradas, contudo, levando em conta a natureza da privacidade[11] na utilização de uma câmera, optou-se por um dispositivo capaz de processar imagens capturadas localmente e de modo *off-line*. O *smartphone* foi considerado como uma segunda alternativa por não apenas já contar com uma câmera, mas também por ser capaz de processar e utilizar localmente algoritmos de aprendizado de máquina e redes neurais[12]. Porém, considerando a comodidade do usuário, utilizar um *smartphone* para a tarefa repetitiva de gerenciar os inventários do lar constantemente tornaria o processo em uma atividade cansativa.

Com a privacidade e comodidade em mente, foi escolhido o uso de um dispositivo dedicado capaz de processar todos os dados localmente para fácil acesso e segurança. Inicialmente, foram considerados o Arduino UNO e o ESP32, mas ambos não possuem a capacidade do processamento necessário para lidar com as informações fornecidas pelo reconhecimento de imagem ou o treinamento através do aprendizado de máquina. A terceira opção, a plataforma Raspberry Pi, do qual possui uma extensa adoção em projetos DIY de domótica, foi comparada às duas primeiras (tabela 1).

**Tabela 1. Comparação entre dispositivos para domótica.**

| CARACTERÍS-TICAS | ARDUINO UNO | ESP32-S3 | RASPBERRY PI 4 |
|---|---|---|---|
| **Tipo** | Microcontrolador | Microcontrolador | Computador |
| **Núcleos** | Single core | Dual core | Quad core |
| **Clock** | 16 Mhz | 160 Mhz | 1.8 Ghz |
| **Arquitetura** | 8 bits | 32 bits | 64 bits |
| **RAM** | 2 KB | 512 KB | 4 GB |
| **Armazenamento** | 32 KB | MicroSD (até 4 GB) | MicroSD (até 1 TB) |
| **Wi-fi** | Não | 2.4 Ghz 802.11b/g/n | WiFi 2.4 Ghz & 5.0 Ghz 802.11ac |
| **USB** | 1x 1.0 | 2x 2.0 | 2x 3.0, 2x 2.0 |
| **Porta de Câmera** | Não | Expansível | 2-lane MIPI CSI |
| **Sistema Operacional** | Xinu | FreeRTOS | Raspberry Pi OS (Raspbian) |
| **Dimensões (CxLxA)** | 76 x 19 x 64 mm. | 40 x 27 x 12 mm. | 86 x 54 x 17 mm. |

O uso do Raspberry Pi 4 se dá através de uma plataforma versátil e acessível que oferece o poder de processamento e conectividade suficiente para todo o projeto e serve como base de um protótipo de produto comercializável. Além do Raspberry Pi 4, foi utilizado uma câmera *web* de alta resolução conectada ao dispositivo, que permite a captura de imagens em alta resolução e frequência de quadros, e um sensor de movimento.

A abordagem de *software* do projeto foi dividida em duas partes. A primeira consiste na visão computacional utilizando algoritmos de redes neurais capazes de identificar alimentos a partir das imagens capturas pela câmera. A segunda consiste em uma plataforma *web* que disponibiliza a lista de itens do inventário para o usuário.

## 4. Desenvolvimento e Testes

### 4.1. Desenvolvimento

A fundamentação do dispositivo se iniciou com a instalação do sistema operacional Raspbian, do módulo de câmera e do sensor de movimento. Com o dispositivo pronto, para o desenvolvimento e implementação de programação em *software*, foi necessário a instalação dos complementos da linguagem de programação Python e os da biblioteca de algoritmos voltados à visão computacional OpenCV (*Open Source Computer Vision Library*, ou "biblioteca de visão computacional de código aberto"), e a criação do repositório no GitHub para o projeto[13] para desenvolvimento de código.

O algoritmo central da visão computacional é o YOLO (*You Only Look Once*, ou "você olha apenas uma vez"), criado por Joseph Redmon e Ali Farhadi, reconhecido por realizar a detecção e identificação de múltiplos objetos em uma imagem ou vídeo através de uma única execução. A escolha deste modelo é motivada pela redução de múltiplos sensores, como os de infravermelho, peso ou laser, e o processamento de dados coletados para alcançar os mesmos objetivos. A versão 4 desta rede neural, considerada o estado da arte e lançada em abril de 2020, possui 80 classes de instruções em seu algoritmo já treinadas. Porém, estas classes não correspondem à necessidade do projeto, já que não se

trata de alimentos ou produtos do lar, por isso foi requerido um novo treinamento de modelo para o algoritmo usado.

O novo treinamento em aprendizado de máquina para as novas classes iniciou-se com a busca de um *dataset* (conjunto de dados) organizado de forma sistemática para análise e aplicação destes dados. Para fins do projeto, era necessário um *dataset* voltado a alimentos em estado natural e cru. Contudo, os *datasets* disponíveis se deram em maior parte com o propósito e foco no reconhecimento de quantidade de calorias em refeições prontas. Além disto, as embalagens e etiquetas possuíam apenas as informações em outras línguas.

Diante desta limitação, foi necessário um *dataset* relevante ao projeto contendo imagens de diversos alimentos típicos de mercado para testar a capacidade do algoritmo de reconhecimento. Estes alimentos incluem abobrinha, batata, biscoito, brócolis, cachorro-quente, café, cenoura, cerveja, *croissant*, hamburger, leite, limão, maçã, morango, ovo, panqueca, pepino, pipoca, pizza, repolho, tomate, uva, vinho e *waffle*. A seleção, extração e anotação de imagens poderia ser feita em duas formas: manualmente com imagens da internet, ou diretamente através de um banco de imagens com anotações prontas.

Por questões educacionais, optou-se, inicialmente, pela elaboração de um *dataset* de autoria própria. Para isto, foram utilizadas ferramentas para Python de *data scraping* (raspagem de dados), uma técnica computacional para extração de dados e arquivos da internet, em conjunto com um *script* (sequência de instruções programadas em *software*) autoral para organizar estas imagens em seus devidos diretórios.

Após as anotações serem aplicadas de acordo com cada imagem, o conjunto de dados foi dividido em *training* (treinamento) e *testing* (teste ou ensaio), sendo o primeiro utilizado para o treinamento de algoritmos e o segundo para avaliar a precisão e generalização do modelo de aprendizagem de máquina. Esta divisão foi necessária para garantir que o modelo seja capaz de reconhecer corretamente os objetos não vistos durante o treinamento, validando assim a capacidade de generalização do algoritmo e do sistema.

Os testes iniciais do *dataset*, entretanto, demonstraram que a precisão do algoritmo está diretamente relacionada à quantidade de imagens fornecidas para cada classe de dado, causando um aumento exponencial na quantidade de imagens e anotações manuais necessárias, e que o *data scraping* feito possuía imagens de qualidade comprometida, tornando diversas partes do *dataset* em obsoletas, impactando significativamente a curva de aprendizagem do algoritmo e sua capacidade de generalização e eficiência. Além disso, suplementarmente, o algoritmo YOLO demanda um padrão específico de leitura das anotações em formato de arquivo texto (TXT), porém as anotações foram originalmente geradas em formato XML (*Extensible Markup Language*), o que exigiria um tempo significativo de correção.

Por isso, a criação de um *dataset* através do uso de imagens extraídas diretamente da internet e as anotações feitas manualmente, para propósito deste projeto, foi abandonada. Um banco de imagens com anotações prontas e diretórios devidamente segregados, fornecido em fonte aberta pelo Google Open Images Database, foi a solução mais prática. Através do programa de código aberto OIDv4_ToolKit, foi possível carregar em poucos segundos cerca de 500 imagens de treinamento para cada classe e cem imagens de teste, todas propriamente anotadas, totalizando cerca de 20% do necessário para uma classe de um *dataset* ideal.

Com as imagens do banco armazenadas e em seus devidos diretórios do sistema, foi necessário a conversão do *dataset* para um padrão acessível à rede neural YOLO, já que o algoritmo possui o seu próprio padrão, diferente do padrão usado pelo serviço Google Open Images Database e as anotações que acompanham as imagens. No padrão YOLO, cada arquivo digital é encapsulado em classes, e cada classe serve como um índice para o banco de imagens, seguindo as coordenadas do objeto na imagem capturada. Para realizar a conversão, foi utilizado o algoritmo TreinamentoCustomizadoYOLO por Beatriz Hemili.

Após o banco de imagens estar preparado e as imagens contidas nele anotadas utilizando o padrão YOLO, o próximo passo foi utilizar um *framework* de treinamento, permitindo condicionar o banco de imagens para o reconhecimento de objetos. Para este projeto, foi escolhido o Darknet Framework por possuir uma extensa documentação e ágil implementação com o restante da solução. Esta integração permite um ambiente flexível e otimizado para implementar as redes neurais convolucionais profundas, permitindo ajustes personalizados, processamento eficiente de dados e otimização do modelo de algoritmo YOLO para alcançar alta precisão e desempenho em processos singulares, como o reconhecimento de objetos feito pelo projeto, detectando múltiplos objetos em tempo real.

Esta integração exigiu alguns ajustes, incluindo modificações nas configurações da rede neural do algoritmo YOLO, das quais estão contidas em um arquivo que engloba todas as classes necessárias, e o arquivo que contém os detalhes para a orquestração do funcionamento do *framework* em si. Além disto, foi necessário separar o arquivo de classes em dois, para assim serem treinados cada um em seu arquivo dedicado. Essa separação foi importante para que, no arquivo de configuração, ele contenha os parâmetros da arquitetura da rede neural, enquanto o arquivo de dados contenha as informações sobre classes e diretórios para os conjuntos de treinamento e teste.

Uma vez configurado e compilado no ambiente do sistema, foi possível utilizar o *framework* para realizar o treinamento profundo de aprendizagem de máquina. Este treinamento é feito em épocas, ou passagens completas de todo o conjunto de dados em treinamento, do qual o *framework* Darknet Framework prontamente acompanha 127 épocas treinadas, com aproximadamente 6.000 épocas necessárias inicialmente para o projeto, um número do qual depende do tamanho do *dataset* utilizado.

O algoritmo de rede neural YOLO foi treinado no ambiente do Google Colab utilizando o *dataset* criado para o projeto, aproveitando recursos de processamento gratuitos e, eventualmente, recorrendo à aquisição de unidades computacionais extras para complementar o treinamento. Optou-se por treinar o modelo por duas mil vezes o número de classes em épocas, visando assegurar uma convergência robusta do modelo. Tal escolha permitiu à rede neural aprender padrões complexos e alcançar generalização eficaz para diferentes instâncias de cada objeto. Apesar do número aparentemente elevado de épocas, essa abordagem favoreceu a estabilidade e aprimoramento do desempenho do modelo. O treinamento foi conduzido com uma frequência de 200 épocas a cada 30 minutos, totalizando aproximadamente duas semanas de treinamento quase contínuo. Esse método refletiu em uma boa eficiência computacional, otimizando o tempo de treinamento sem comprometer a qualidade do modelo final. O resultado desse treinamento pode ser encontrado no caderno Google Colab do projeto[14].

Três *scripts* na linguagem de programação Python foram criados para o fluxo da

captura, identificação, catalogação e organização do inventário conforme os testes realizados em diferentes inventários necessitaram suas próprias especificações, todos dos quais integraram o uso do sensor de movimento, da câmera e da visão computacional para identificar a abertura do inventário, o tempo de inatividade, e a captura de imagem.

O primeiro *script*, responsável pela integração do sensor de movimento, após um período determinado de inatividade, enquanto ativo, utiliza a visão computacional em busca de um dos itens treinados no *dataset*. Dentro deste cenário, o *script* engatilha a captura de uma imagem e a disponibiliza para a análise de reconhecimento de objetos. O segundo *script*, responsável pelo reconhecimento de objeto, cria uma ordenação temporária de cada objeto capturado e identificado pela imagem em um *array* (estrutura de dados), começando pela quantidade geral de itens. A ordenação temporária é então comparada com a ordenação previamente criada pelo sistema, identificando se houveram adições para a quantidade total. Com base nisso, todos os itens que estavam armazenados no inventário são mantidos na ordenação, e os que foram adicionados ou removidos são modificados na ordenação através de uma atualização. A ordenação é então enviada através de uma requisição HTTP (*Hypertext Transfer Protocol*, ou "Protocolo de Transferência de Hipertexto") para o servidor através de um terceiro *script*, disponibilizando cada produto para manipulação e gerenciamento na plataforma *web* pelo usuário, efetivamente criando o inventário virtual.

Para o armazenamento dos dados da despensa e histórico de produtos, foi utilizado o Redis, um sistema de banco de dados em memória. A escolha do Redis se deu pela sua alta performance na manipulação de dados-chave, proporcionando rápida recuperação de informações e agilidade nas operações de inserção e remoção de itens da despensa. A integração do Redis com Node.js permitiu uma comunicação ágil entre a aplicação *web* e o banco de dados, viabilizando um gerenciamento em tempo real dos produtos.

A interface da plataforma *web* desenvolvida para gerenciar os itens da despensa foi implementada utilizando as tecnologias HTML (*HyperText Markup Language*, ou "Linguagem de Marcação de Hipertexto"), CSS (*Cascading Style Sheets*) e JavaScript, em conjunto com o *framework* Node.js para a construção do servidor. Node.js foi adotado devido à sua capacidade de facilitar o desenvolvimento de aplicações *web* escaláveis e eficientes, permitindo a execução de código JavaScript no lado do servidor. Esta escolha proporcionou uma interação dinâmica e responsiva na interface do usuário, otimizando a experiência do usuário final. Durante todo o processo de desenvolvimento, diversos designs da interface da plataforma web foram adotados a fim de tornar a experiência de navegação a mais eficiente e produtiva possível para o usuário.

A funcionalidade de exibição do inventário foi implementada para permitir ao usuário visualizar os itens presentes na despensa, facilitando o controle e a reposição dos alimentos. Além disso, o histórico de entrada e saída de produtos foi implementado para fornecer um panorama das movimentações na despensa ao longo do tempo.

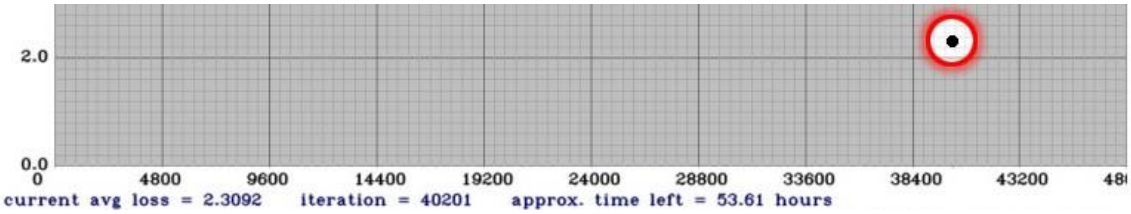
Para as sugestões de receitas, foi empregado a API (*Application Programming Interface*) da inteligência artificial ChatGPT, que constitui em um método de comunicação entre as implementações JavaScript, Node.JS e o serviço ChatGPT. O uso desta API é gratuito, apesar de ter uma limitação na quantidade de requisições diárias.

### 4.2. Testes

A fase inicial de testes da visão computacional se iniciou ao treinamento do modelo

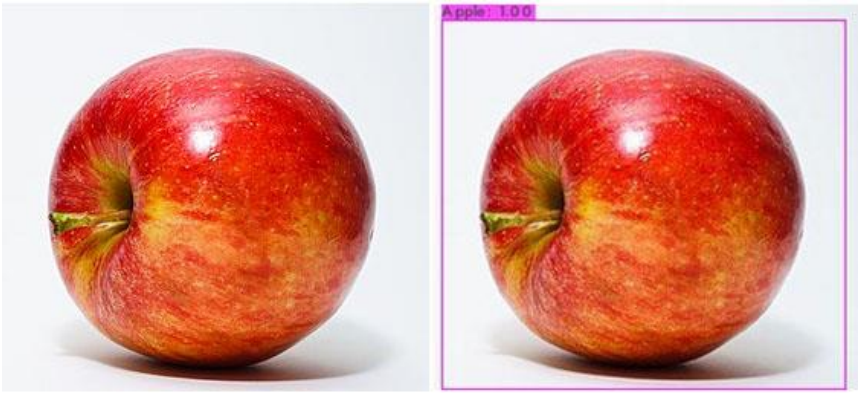
atingir 40.201 épocas de um total de 48.000 épocas e com cada classe superando a pontuação MAP (*Mean Average Precision*, ou "precisão média") mínima de 1% após 53 horas de treino, com algumas das classes de objetos atingindo 40% de precisão. Utilizando uma função para visualizar a curva de aprendizado do algoritmo, foi possível criar um gráfico (figura 1) que demonstra a margem de erro de aprendizado do algoritmo, representado pelo eixo Y, comparado com a quantidade de épocas treinadas, representado pelo eixo X. O resultado mostrou uma taxa de 2,3092 de precisão, próximo do ótimo 2,0.

**Figura 1. Margem de erro de aprendizado e a quantidade de épocas treinadas.**




O primeiro teste do modelo treinado foi feito utilizando as imagens do *dataset* (figura 2) em um ambiente virtual. Os resultados obtidos revelaram uma taxa de precisão de 100% de acerto, fornecendo evidências da eficácia do modelo desenvolvido. Este e outros dados obtidos através de testes subsequentes foram utilizados para avaliar o desempenho do modelo treinado ao longo do desenvolvimento. Por meio desta representação gráfica, foi possível observar a evolução das métricas de precisão e perda ao longo do processo de treinamento, demonstrando uma progressão consistente e gradual no aumento da precisão do modelo.

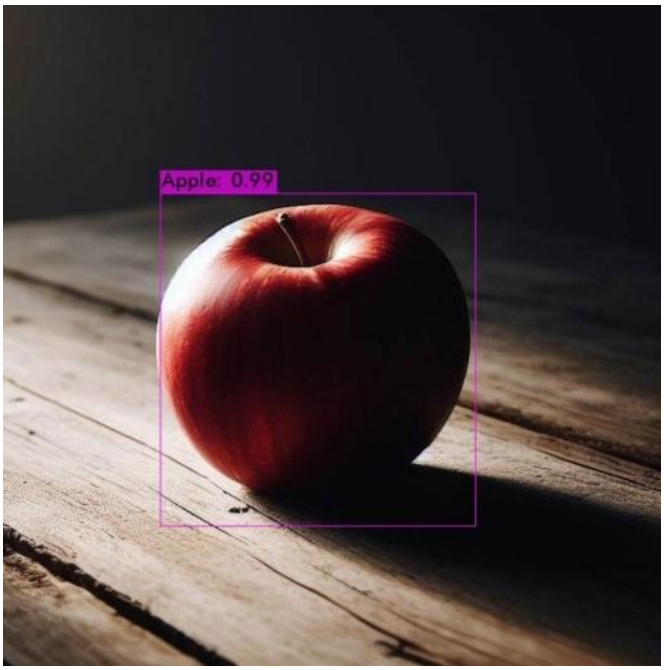
**Figura 2. Teste de detecção com conteúdo do *dataset* em ambiente virtual.**




Com resultados satisfatórios em um ambiente virtual, o modelo treinado foi compilado e instalado no dispositivo Raspberry Pi 4, permitindo a execução do algoritmo diretamente no dispositivo. Para adaptar o funcionamento do modelo no ambiente do Raspberry Pi e garantir a interação correta entre o algoritmo de detecção e a câmera, foi necessário utilizar um *script* para atualizar e configurar os caminhos de entrada e saída dos dados no código-fonte.

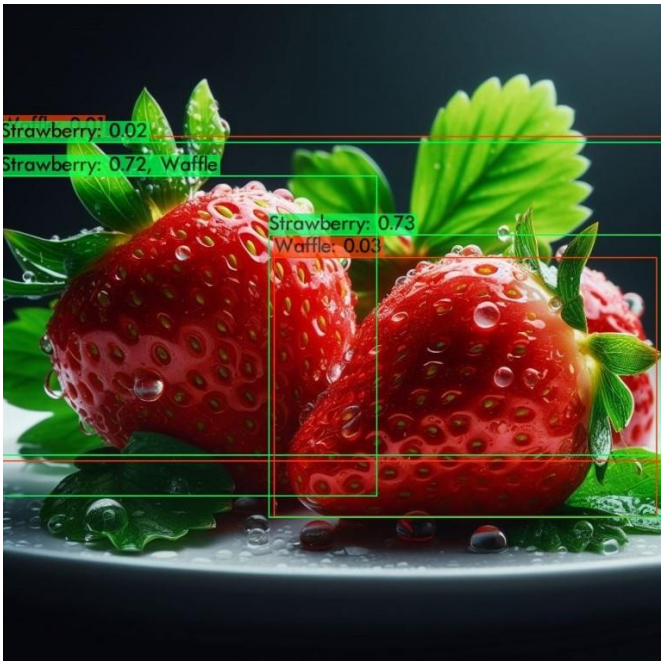
Após a configuração bem-sucedida do sistema no Raspberry Pi, procedeu-se à captura física de itens representativos das classes pertinentes ao treinamento do algoritmo YOLO através da câmera. Como base, utilizando uma maçã, a taxa de 99% de precisão (figura 3) foi alcançada.

**Figura 3. Teste de detecção em ambientes reais com uma maçã.**




Entre os testes subsequentes realizados, uma sequência foi feita utilizando múltiplos alvos para testar a capacidade do algoritmo de lidar com mais de um alimento em uma única captura. Um teste realizado com morangos (figura 4) detectou com taxas de precisão entre 1% e 3% o alimento *waffles*, apesar de não estar presente na imagem capturada. Foi com este resultado que um novo desenvolvimento no *script* de reconhecimento foi feito, utilizando um algoritmo baseado em matrizes que compara a taxa de precisão média das classes do modelo com os resultados obtidos através das capturas de imagens, permitindo remover resultados indesejáveis com taxas desproporcionalmente baixas em comparação às taxas mais resolutas dos alimentos alvos.

**Figura 4. Teste de detecção com múltiplos alvos, utilizando morangos.**



Outro teste que demonstrou resultados interessantes utilizando múltiplos objetos simultaneamente na captura de imagem foi o da pipoca (figura 5). Neste teste, diversos objetos que não obtiveram treinamento no modelo do *dataset* foram introduzidos. Apesar da pipoca ter sido identificada com apenas 19% de taxa de precisão, a capacidade do algoritmo lidar com múltiplos objetos ao mesmo tempo foi considerada um sucesso.

**Figura 5. Teste de detecção com múltiplos objetos com e sem treinamento.**

Após a geração inicial da análise dos alimentos capturados, foi realizada uma segunda análise com a remoção intencional de um item reconhecido anteriormente. Este procedimento visou avaliar a consistência e precisão do sistema de detecção de alimentos ao lidar com mudanças no ambiente capturado pela câmera do Raspberry Pi. A análise dos dados em todas as pastas de saída revelou a capacidade do algoritmo YOLO em atualizar adequadamente as informações, refletindo a remoção do item identificado na análise anterior. O inventário virtual foi ajustado conforme o esperado, indicando a ausência do item removido, enquanto a imagem atual capturada pela câmera do Raspberry Pi refletiu com precisão a nova composição do ambiente, confirmando a coerência dos dados nas diferentes saídas do sistema. O processo de comparação entre um item inserido e removido em um inventário contou com o uso do *script* de comparação autoral para a remoção de resultados indesejáveis.

Paralelamente aos testes, foi dada continuação ao treinamento do *dataset*, o que eventualmente se provou prejudicial para modelo final. Apesar de altas taxas de precisão com o conteúdo treinado, os testes de reconhecimento de capturas e cenários distintos ao do *dataset* demonstraram resultados inválidos ou inexistentes. Este problema é conhecido no campo de aprendizagem de máquina como *overfitting* ("sobreajuste"), que ocorre com o treinamento excessivo do *dataset*. Para propósito de finalização do projeto, foi criado um segundo *dataset* com menos classes.

Na fase de desenvolvimento da interface do gerenciador de itens e inventários, foram conduzidos diversos testes de funcionalidade e usabilidade. A plataforma *web* foi submetida a verificações para garantir a interatividade e a apresentação adequada dos

dados relacionados aos inventários domésticos. Utilizando uma abordagem de testes abrangentes, a compatibilidade entre diferentes navegadores foi verificada, assegurando a consistência e funcionalidade da interface em ambientes variados. Além disso, foram avaliadas as respostas do sistema em relação à manipulação e exibição de inventários em diferentes dispositivos, como *desktops*, *tablets* e *smartphones*, assegurando uma experiência de usuário uniforme e acessível em todas as plataformas. A análise dos testes revelou uma implementação eficaz das funcionalidades, garantindo a integridade das informações exibidas e a facilidade de interação para os usuários, cumprindo os objetivos propostos na criação da interface de gerenciamento de inventários.

## 5. Conclusões

No desfecho deste estudo, os resultados obtidos no desenvolvimento do sistema de gerenciamento de inventários inteligentes apresentaram-se consistentes em sua capacidade de detecção e reconhecimento de alimentos. A utilização do algoritmo YOLO treinado revelou uma eficiência notável na identificação dos itens, tanto em simulações virtuais quanto em ambientes físicos capturados pela câmera conectada ao Raspberry Pi. O sistema demonstrou habilidade na atualização dinâmica das informações, refletindo com precisão as alterações no ambiente e ajustando adequadamente o inventário.

Os principais acertos deste projeto incluem a implementação do modelo de visão computacional, a integração entre *hardware* (Raspberry Pi) e *software* (algoritmo YOLO), e a consistência na detecção e reconhecimento dos alimentos. Contudo, alguns desafios foram enfrentados, como a necessidade de otimização do desempenho do sistema para lidar com uma variedade maior de itens e condições de iluminação variáveis que afetam a precisão da detecção. A criação de um *dataset* de imagens com anotações detalhadas dos alimentos demandou um tempo considerável. A decisão foi feita em prol da adoção de um banco de imagens disponível publicamente e facilmente conversível, otimizando o processo de treinamento do algoritmo YOLO.

Além disso, o treinamento do algoritmo YOLO exclusivamente no Raspberry Pi demandaria um tempo excessivo de processamento. Nesse contexto, foi essencial recorrer ao Google Colab para realizar um treinamento mais ágil e eficiente, possibilitando uma implementação mais rápida do modelo treinado no dispositivo Raspberry Pi. Essa abordagem pode beneficiar outros usuários, uma vez que o treinamento do algoritmo pode ser executado uma única vez no Google Colab e, posteriormente, utilizado por diversos dispositivos, otimizando recursos e tempo de processamento.

Considerando os avanços alcançados e os desafios enfrentados, este estudo fornece uma base sólida para futuras melhorias e aplicações do sistema de gerenciamento de despensa inteligente, destacando a importância contínua da pesquisa e desenvolvimento nesse campo emergente da visão computacional aplicada à gestão de alimentos e outros produtos do lar. Sugestões para futuros desenvolvimentos incluem explorar a precisão na identificação de alimentos vencidos através da integração de modelos com treinamentos mais precisos, a utilização dos métodos *dropout* e *data augmentation* usados em aprendizagem de máquina para reverter o resultado de *overfitting* do modelo treinado, a identificação de códigos de barras para embalagens industrializadas, criação de listas de compras inteligentes e recorrentes conectadas a aplicativos de pedidos via internet, e a análise de consumo, todas das quais podem contribuir para práticas mais sustentáveis e maior comodidade.

## 6. Referências


1. Schanes, K.; Dobernig, K.; Gözet, B. (2018), "Food Waste Matters — A Systematic Review of Household Food Waste Practices and Their Policy Implications", https://doi.org/10.1016/j.jclepro.2018.02.030.
2. Hyungjoon, K. (2021), "Use of Mobile Grocery Shopping Application: Motivation and Decision-Making Process among South Korean Consumers", https://doi.org/10.3390/jtaer16070147.
3. Wasson, S. (2023), "How Expensive Are Smart Homes?", https://todayshomeowner.com/smart-home/guides/how-expensive-are-smart-homes/, acessado em 29/07/2023.
4. Wtihanage, C.; Ashok, R.; Yuen, C.; Otto, K. (2014), "A Comparison of the Popular Home Automation Technologies", https://doi.org/10.1109/ISGT-Asia.2014.6873860.
5. Rebouças, E. P. (2020), "Análise do Mercado de Casas Inteligentes no Brasil: Uma Pesquisa Exploratória por meio de Surveys.", http://repositoriosenaiba.fieb.org.br/handle/fieb/1099.
6. Chandon, P. E Wansink, B. (2006), "How Biased Household Inventory Estimates Distort Shopping and Storage Decisions", http://dx.doi.org/10.1509/jmkg.70.4.118.
7. De Angelo, C. F.; De Lara, S. J. P.; Fávero, L. P. L. (2003), "As Compras Não Planejadas Em Supermercados: A Importância Do Tempo E Da Organização Da Loja Na Determinação Dos Gastos", https://doi.org/10.1590/S1415-65552003000300008.
8. Baker, Scott R.; Johnson, S.; Kueng, L. (2020), "Financial Returns to Household Inventory Management", https://dx.doi.org/10.2139/ssrn.3677084.
9. Wageningen Food & Biobased Research (2023), "Does Extra On-Pack Information Impact Consumers' Understanding of Expiry Dates and Food Waste Behaviour? Insights from A Real-Life Experiment in The Netherlands", https://doi.org/10.18174/633798.
10. Kumar, J. N. A.; Chimmani, S. (2019), "Proposal of Smart Home Resource Management for Waste Reduction and Sustainability Using AI and ML", https://doi.org/10.1109/ICCES45898.2019.9002031.
11. Rodriguez-Conde, I., Campo, C., Fdez-Riverola, F. (2021), "On-Device Object Detection for More Efficient and Privacy-Compliant Visual Perception in Context-Aware Systems", https://doi.org/10.3390/app11199173.
12. Dhar, S.; Guo, J.; Liu, J.; Tripathi, S.; Kurup, U.; Shah, M. (2019), "On-Device Machine Learning: An Algorithms and Learning Theory Perspective", https://doi.org/10.48550/arXiv.1911.00623.
13. Autor (2023), https://github.com/UAMCGMP/TCC
14. Autor (2023), Google Colab — Caderno com Treinamento do Dataset, https://colab.research.google.com/drive/1G4bM5Go6qgiwpkhNWadAkZ33fCssW3Ic#scrollTo=u2H0pKeenitg