Universidade de São Paulo

Escola Politécnica

Departamento de Engenharia Mecatrônica
e de Sistemas Mecânicos

Trabalho de Conclusão de Curso

Automatização dos Caixas de Lojas a Partir da Implementação da Visão Computacional

Henrique Igai Wang - 8941957
Lucas Kyoshi Miyazaki - 9298976

Orientador: Prof. Dr. Marcos de Sales Guerra Tsuzuki

SÃO PAULO - SP
2020

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Com o objetivo de facilitar o cotidiano dos cidadãos, será desenvolvido um sistema que utiliza visão computacional para resolver um problema comum do dia-a-dia: as filas nos caixas dos etabelecimentos. Na maior parte do dia, as filas para realizar os pagamentos são pequenas. Entretanto, durante alguns horários de pico, as filas podem se tornar longas, principalmente em estabelecimentos movimentados. Foram identificadas duas possíveis causas para esse problema.

O primeiro motivo seria o número insuficiente de caixas e atendentes nos estabelecimentos. Se a quantidade de caixas aumentar, a vazão dos clientes será maior, diminuindo o tempo de espera nas filas. Não obstante, esta solução não é vantajosa aos estabelecimentos, uma vez que os caixas ocupam um espaço considerável e possuem um custo de mão de obra. A segunda causa para as longas filas seria o processo da identificação dos produtos para o pagamento da compra. Há varios casos em que o funcionário tem dificuldade em passar o código de barras do produto ou em identificar o produto que está sendo comprado. Consequentemente, o processo de finalização da compra é lento e há um acúmulo de clientes na fila de espera.

Analisando as duas possíveis causas para o problema das filas, decidiu-se buscar por uma solução atacando o segundo motivo, uma vez que não é desejado que o custo do lojista aumente com a mão de obra extra. Para isso, foi decidido que se desenvolveria uma forma de automatizar os caixas em lojas de doce e padarias.

Atualmente, a automação de processos vem se tornando uma tendência cada vez mais forte no mercado. Trazendo inúmeros benefícios às empresas - tais como redução de tempo de produção e de atividades operacionais, otimização da tomada de decisões e geração de relatórios mais precisos - pode-se observar um aumento bastante significativo nos investimentos no setor de automação, atuando em diversas áreas do mercado. Tendo isso em mente, foi pesquisado no mercado atual por tecnologias possíveis de se utilizar no projeto, e se concluiu que a identificação de produtos através de algoritmos de redes neurais seria a melhor alternativa, tendo em vista o seu principal objetivo para esta pesquisa: criar um sistema viável e barato que auxilie no processo de automatização dos caixas. Dessa forma, decidiu-se que seria desenvolvido uma solução de automatização dos caixas para lojas como confeitarias e padarias.

2 Revisão Bibliográfica

Antes de iniciar o desenvolvimento do projeto, foi necessário realizar o estudo de artigos científicos e de aplicações presentes no mercado, aprendendo sobre as diferentes tecnologias que poderiam ajudar com o objetivo do projeto - a identificação de produtos de lojas de doces e de padarias. Nesta seção são apresentadas estudos de diversas tecnologias aplicadas na tentativa de automatizar lojas de varejo.

2.1 Uso de Tags de RFID Passivos

Esta solução consiste na presença de um RFID passivo instalado na embalagem de cada produto da loja, que servirá para a identificação dos itens que o cliente deseja comprar, assim como o código de barras - sistema mais utilizado nos supermercados atualmente. Dessa forma, quando o usuário sair do supermercado, ele passará por um sensor - que identificará os produtos que carrega consigo - e deverá fazer a sua identificação para que possa ser cobrado o valor total referente a sua compra. Um exemplo de uma loja com este esquema no Brasil é a "Zaitt", uma loja de conveniência localizada na região Itaim Bibi da cidade de São Paulo.

Parada et al. [21] traçam um comparativo entre as tecnologias empregadas no mercado:

Tabela 1: Tabela comparativa das tecnologias de identificação de produtos

Tecnologia Empregada	Custo	Identificação do Item	Detecta Interação do Usuário
UHF RFID	baixo	sim	sim
QR Code	baixo	não	não
Visão Computacional	alto	não	sim

- Vantagens:
  - É uma opção confiável, não necessita de grandes esforços computacionais e nem de algorítimos complexos.
  - Os custos de instalação são baixos, como pode-se perceber na Tabela 1.
  - Capacidade para armazenar vários tipos de informações - tais como preço, data de fabricação, data de validade e código do produto - e a leitura delas pode ser feita de forma rápida e à distância com o auxílio de leitores, motivo pelo qual várias empresas estão investindo nesta tecnologia [9].
  - Possibilita uma melhora na administração de estoque [16] e da segurança [31], uma vez que ele possibilita o acompanhamento em tempo real do movimento e da quantidade do estoque.

- Desvantagem:
  - Lojas ou fornecedores de produtos necessitariam colocar um identificador RFID em todos os seus produtos.
  - Não é possível colocar identificadores em alguns produtos sem embalagens, como frutas, verduras e doces.
  - Atualmente não existe uma padronização na fabricação e programação dos RFID's [16].
  - Não há uma grande quantidade de fornecedores de RFID no mercado atual [16].

2.2 Uso de *Tags* de QR Code

Esta é a ideia que mais se assemelha a compra online em que o usuário vai adicionando os produtos desejados em um carrinho virtual. Cada produto dentro da loja possui um QR Code, e o cliente deve escaneá-los com o aplicativo da loja para adicioná-los ao carrinho. Ao finalizar a compra, o usuário deverá ir à catraca e escanear o seu QR Code, que registrará sua compra. Um exemplo de aplicação deste esquema na Índia é a "Decathlon's Scan and Go" [6]. Outro ótimo exemplo é o "April Gourmet", localizado na cidade de Pequim. Adicionando QR Codes em todos os produtos, a loja permite com que seus clientes verifiquem informações básicas sobre o item de interesse, e que o registre em seu carrinho de compra com o próprio aplicativo da loja. Dessa forma, antes do usuário sair da loja, ele deverá fazer o *checkout* no aplicativo para finalizar a sua compra.

- Vantagem:
  - Barato de se produzir. Hoje em dia, QR Codes podem ser produzidos por aplicações ou por websites [20].
  - Promove a marca da loja a partir do uso de tecnologias de ponta [11].
  - Possibilita o fornecimento de vários tipos de informações. Há museus que já utilizam esse tipo de serviço em suas exposições. Quando o visitante se interessa por uma peça, ele pode escanear o QR Code com a ajuda de seu celular ou *tablet*, e é redirecionado para uma página que contenha informações sobre o assunto que deseja [20].
- Desvantagens:
  - O sistema fica muito dependente de um aparelho celular ou tablet para a leitura dos QR Codes [20].
  - O cliente deve estar cadastrado no sistema do aplicativo que fornece o instrumento de compras [11].
  - Mesmo problema dos tags de RFID passivo: alguém deve colocar uma tag com QR Code em todos os produtos.
  - Há o risco do aplicativo leitor de QR Code não ser compatível com o aparelho celular do cliente e de ter problemas de conexão - necessita uma conexão com a Internet estável [11].

2.3 Uso de *Tracking*

Nesta solução, o cliente é rastreado e todas as suas ações são identificadas, desde pegar um produto da prateleira, até sair da loja. Para ter acesso ao estabelecimento, o usuário deve se identificar na entrada da loja através de um aplicativo de celular, e caso deseje comprar um produto, basta retirá-lo da prateleira e levá-lo consigo. Para finalizar a compra, basta sair da loja, que o valor da compra será descontado da conta bancária do usuário. Para esta solução, os autores deste artigo interpretaram que ela pode ser aplicada tanto por meio da visão computacional, quanto por pela tecnologia RFID.

Uma solução de visão computacional pode ser encontrada no artigo de Frontoni et al. [13]. O texto apresenta uma forma de rastrear e identificar a interação dos clientes com os produtos da loja a partir de câmeras RGBD posicionadas verticalmente en frente às prateleiras. Com a combinação de dois algoritmos - um de identificação do cliente e outro de identificação das mãos do próprio - o sistema consegue determinar duas ações: compra do cliente e análise seguido da desistência da compra do produto por parte do mesmo. O conjunto de etapas "mão vazia, mão cheia e mão vazia" indica que o consumidor olhou a mercadoria e desistiu de sua compra por algum motivo. Já "mão vazia, mão cheia, cliente saiu de sua posição inicial" sugere que ele deseja comprar o produto. Um bom exemplo de aplicação da visão computacional no ramo das vendas de varejo é o Amazon GO, localizado em Seattle. Para entrar neste estabelecimento, o cliente deve se identificar por meio de um aplicativo de celular. Quando dentro, basta retirar os itens que deseja das prateleiras e deixá-los em suas mãos, ou até mesmo em sua mochila, que câmeras espalhadas por todo estabelecimento identificarão suas ações. Após sair da loja, o total da compra será debitado no cartão de crédito do cliente [10].

Outra solução envolvendo visão computacional pode ser encontrada no artigo de Harikrishna G. N. Rai [14]. Nesse texto, seus autores propõem uma solução utilizando códigos de barras únicos presos aos carrinhos de supermercado. Câmeras de segurança posicionadas em toda loja identificam o carrinho pelo seu código de barra, e atualizam sua posição no sistema. Outra solução citada nesse mesmo texto, porém não desenvolvida pelos seus autores, é a instalação de RFIDs nos carrinhos de supermercados. Dessa forma, suas posições serão identificadas através de sensores wireless de RFID.

- Vantagens:
  - Soluciona o problema de furtos
  - É a opção mais prática para os clientes - ele não precisa passar por nenhum "*scanner*" e nenhuma catraca para a realização da compra.
  - O estabelecimento pode obter informações sobre as tendências de consumo de seus clientes e utilizá-las para diversas aplicações, tais como definições de promoções e do posicionamento dos produtos, além da disponibilização de cupons de acordo com o consumo do usuário [14]

- Desvantagens:
  - Custos computacionais altos
  - Necessidade de um alto investimento inicial para a instalação do sistema de traqueamento - câmeras ou sensores RFID
  - Necessidade do usuário necessitar de um cartão de crédito. De acordo com os dados apurados pelo governo brasileira, aproximadamente 44% da população brasileira realizou operações de crédito em 2017 [5]. Ou seja, o requisito de ter um cartão de crédito exclui os 66% restantes da população
  - Quando feito por visão computacional, o sistema fica sucetível a erros causados por baixa resolução das imagens capturadas pelas câmeras e pela constante variação da luminosidade do ambiente.

2.4 Uso de Câmeras nos Caixas

Após reunir todos os produtos desejado, o cliente deve ir a um dos caixas presentes na loja, colocar cada item na posição indicada. Quando todos os produtos desejados estiverem registrados no sistema, o usuário poderá pagar com o auxílio de uma máquina ou poderá fazer a transação de forma automática por meio de uma conta previamente registrada. No artigo de Wu et al. [30], múltiplos produtos são posicionados a frente de uma câmera fixa e os preços são calculados.

- Vantagens:
  - Fácil adaptabilidade a infraestrutura já existente de lojas e supermercados, ou seja, provável maior aceitamento tanto pelos donos das lojas, quanto pelos seus clientes.
  - Requer um número pequeno de câmeras - igual à quantidade de caixas. Conquentemente seu custo computacional é menor em relação à solução de traqueamento citada no subitem acima.
  - Ambiente onde é realizada a análise é bem controlado. A luminosidade de um pequeno espaço delimitado é muito mais controlável do que de um estabelecimento inteiro.
- Desvantagens:
  - Menor nível de automação, pois o cliente necessita passar pelo caixa automático para a identificação da compra e para a efetuação do pagamento.
  - Na etapa inicial, logo após a instalação do sistema, é necessário a presença de um número mínimo de funcionários para auxiliar os clientes a utilizarem essa nova tecnologia, assim como na primeira experiência de Meiling Du no estabelecimento da Amazon Go [10]. Ele relatou que o aplicativo fornece um manual de instrução, porém, caso tivesse alguma dúvida, haviam funcionários dentro da loja para esclarecê-las.

3 Metodologia de Projeto

Após analisar todas estas possibilidades, foi decidido que serão utilizadas câmeras nos caixas para este projeto. Acredita-se que esta solução possui maior flexibilidade e, portanto, se adaptará melhor nas lojas brasileiras.

Neste projeto pretende-se focar em lojas de doces, confeitarias e padarias, cujos produtos não possuem etiquetas. O algoritmo será desenvolvido a partir do YOLO, uma rede neural que classifica múltiplos objetos de forma mais rápida que a maioria das redes [32].

Outros projetos de visão computacional para identificação de produtos com resultados bem convincentes são as pesquisas de Franco et al. [12] e de Santra and Mukherjee [24]. Em ambos os artigos são descritas técnicas de distinção e de identificação de produtos nas prateleiras de supermercados. É utilizado a integração de algoritmos de visão computacional baseados na distinção das formas, das cores, dos gradientes e até das palavras presentes em cada produto. Neles, também foram utilizados conceitos de Deep Learning para o treinamento das máquinas. Os autores de ambos artigos alegaram que as informações obtidas por meio desses recursos podem ser muito valiosas para os donos das lojas de varejo, uma vez que elas podem ser usadas para a elaboração de um planograma da loja.

Ao fim do projeto, deseja-se descobrir se é possível criar um algoritmo de visão computacional robusto o suficiente para identificar pães, doces e outros alimentos presentes em padarias e docerias brasileiras.

3.1 Redes Neurais

Uma rede neural artificial é um modelo computacional baseado no sistema central nervoso animal capaz de se adaptar e aprender. É composto de diversas unidades (neurônios) conectados que funcionam em paralelo. As conecções entre cada nó possuem pesos e cada nó possui uma função de ativação previamente definida. O processo de aprendizagem de uma rede consiste na adaptação de cada um destes pesos de acordo com um banco de dados de treinamento [17].

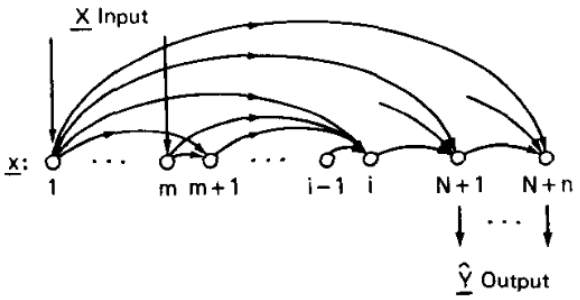


Figura 1: Modelo de rede neural totalmente conectada (retirada do artigo de Werbos [29]).

De acordo com Werbos [29] a rede neural pode ser descrita como na Fig. 1. Uma rede neural é descrita algebricamente pela equação

$$x_i = X_i, \text{ para } 1 \leq i \leq m \tag{1}$$

que representa as entradas da rede neural, pela equação

$$Y_j = x_i, \text{ para } N \leq i \leq N + m, 1 \leq j \leq n \tag{2}$$

que define as saídas e pelas equações

$$net_i = \sum_{j=1}^{i-1} W_{ij} x_j, \text{ para } m \leq i \leq N + m \tag{3}$$

$$x_i = s(net_i), \text{ para } m \leq i \leq N + m \tag{4}$$

que mostram que a saída de cada neurônio é a função ativação da soma entre sinais de entrada do neurônio onde X são as entradas, Y são as saídas, x são todas as conexões entre neurônios, m é o número de entradas, $N + n$ o número de neurônios, n o número de saídas, *net* é o potencial de excitação do neurônio e $s(x)$ é a função de ativação. Normalmente, as redes neurais são divididas em camadas: a camada de entrada, as do meio denominadas como *hidden layers* e a de saída. Os pesos W entre unidades de uma mesma camada são zerados.

Dada uma amostra de treinamento, para se treinar uma rede neural, deve-se minimizar o erro E da equação

$$E = \sum_{t=1}^{T} \sum_{i=1}^{n} \frac{(\dot{Y}_i(t) - Y_i(t))^2}{2} \tag{5}$$

entre o Y real da amostra e o $\dot{Y}$ previsto. Para minimizar E ao longo das T épocas e para as n saídas, utiliza-se um algoritmo de backpropagation para encontrar as derivadas parciais de E em relação aos pesos, e então, ajustam-se os pesos iterativamente.

Podem ser aplicadas diferentes funções ativação em cada camada que podem alterar o desempenho da rede. As equações

- ReLU
$$s(x) = \alpha(x - th) \tag{6}$$
- Leaky ReLU
$$s(x) = \begin{cases} \alpha x, & \text{se } x < 0 \\ x, & \text{senão} \end{cases} \tag{7}$$
- Sigmoid
$$s(x) = \frac{1}{1 + e^{-x}} \tag{8}$$
- Softmax
$$s(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}, \text{ para } x = \{x_1, x_2, \ldots, x_k\} \tag{9}$$

são exemplos de algoritmos utilizadas para funções de ativação encontrados na documentação do Keras [4], onde α, th e k representam, respectivamente, o hiperparâmetro, o *threshold* e o *sharpness parameter* da rede.

3.2 Redes Convolucionais

Para a identificação de imagens, uma rede neural pode funcionar. Porém, alimentar uma rede neural com centenas de variáveis como é o caso de uma imagem, gerará um rede com milhares de pesos, e isto pode acarretar em uma rede sobre ajustada (*overfitted*). Redes convolucionais, por sua vez, extrai características locais de uma imagem facilitando a análise.

Nas camadas iniciais de uma rede convolucional, normalmente, são extraídas características mais simples como bordas e cantos. Nas camadas seguintes, estas informações são combinadas, podendo se identificar formas mais complexas. A extração destas características é feita a partir da convolução entre a imagem e uma matriz de pesos (*kernel*). A saída desta operação (veja a Fig. 2) é um mapa de caracterìsticas (*features map*) de dimensões $n_{out} \times n_{out}$. Dadas as dimensões de entrada $n_{in} \times n_{in}$, do *kernel* $k \times k$, do preenchimento (*padding*) $p \times p$ e do passo (*stride*) $s \times s$, Hien [15] calcula as dimensões de saída n_{out} pela equação

$$n_{out} = \frac{n_{in} + 2p - k}{s} + 1. \tag{10}$$

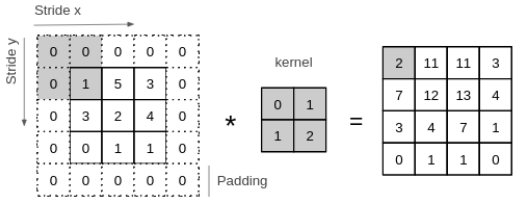


Figura 2: Ilustração de uma operação de convolução aplicada a imagem antes de ser processada pela rede neural totalmente conectada, as janelas de pixels são multiplicadas por um *kernel* gerando uma nova imagem

No artigo de LeCun and Bengio [18] é proposto uma rede convolucional neural (CNN) cujas camadas se intercalam entre camadas convolucionais e camadas que reduzem a resolução do mapa de características. No artigo é proposto uma operação de averaging, reduzindo o tamanho do mapa por dois para tornar o algoritmo mais robusto.

Outro método de redução de dimensionalidade listado na documentação do keras [4] é o max pooling ilustrado na Fig. 3. A operação seleciona o valor máximo dentre os valores de uma região de dimensão $n \times n$ para simplificar a informação nela contida. Segundo Boureau et al.. [8] o *max pooling* é ideal em casos que as features estão esparsas.

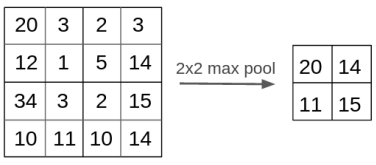


Figura 3: Ilustração de uma operação de *MaxPool* aplicada a imagem, em uma janela de pixels apenas o pixel de maior valor é selecionado.

3.3 YOLO

Publicado em 2012 por Joseph Redmon e Ali Farhadi, o *You Only Live Once*, também conhecido por YOLO, é um sistema de redes com camada convolucionais e camadas totalmente conectadas cujo principal objetivo é detectar e identificar objetos. Explicando de forma sucinta, diferente de modelos CNN tradicionais, sua principal vantagem é a sua capacidade de reconhecer múltiplos objetos presentes em uma imagem, acessando apenas uma vez a rede neural. O output da rede são um conjunto de caixas que segmentam o objeto (posição do centro e dimensões da caixa) e o conjunto de probabilidade de pertencer a cada classe.

Atualmente, o YOLO está em sua quarta versão, sendo que esta é a primeira versão em que seus originais autores não participam da pesquisa. De acordo com o texto de Roman Orac [19] - "*What's new in YOLO v4?*" - Joseph Redmon e Ali Farhadi decidiram se retirar das pesquisas do YOLO, uma vez que eles não podiam ignorar alguns problemas éticos que algumas aplicações de seu sistema estavam gerando, como o desenvolvimento de armas inteligentes.

Para esta pesquisa, inicialmente foi decidido utilizar a terceira versão do YOLO (*YOLOv3*) [23], uma vez que a documentação oficial da quarta versão ainda não havia sido publicada. Não obstante, dado o fato de que ela foi publicada em abril de 2020, optou-se em começar a utilizar o *YOLOv4* [7]. A Fig. 4 apresenta o esquema da estrutura desta nova versão.

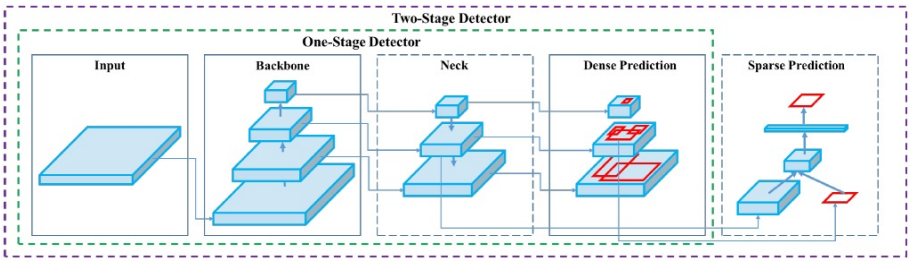


Figura 4: Estrutura do YOLOv4 (extraído da documentação oficial [7]).

De acordo com o artigo oficial [7], o *YOLOv4* foi desenvolvido com o objetivo de melhorar a performance, permitindo o treinamento e a detecção de objetos utilizando GPU's convencionais. Para ser utilizado como referência, o autor mencionou que para até GPU's GTX 2080/1080 Ti, o sistema apresentou bons resultados em questão ao tempo gasto nos testes. Ademais, a nova versão também obteve uma melhoria em sua acurácia. No texto do *YOLOv4* é citado que ele obteve uma melhora de 10% no AP (*Average Precision*) e de 12% no FPS (*Frames per Second*) em comparação ao *YOLOv3*.

O mAP (*mean Average Precision*) é um indicador bastante utilizado para medir a acurácia em modelos de *machine learning*. A equação

$$mAP = \frac{TP}{TP + FP} \tag{11}$$

descreve o mAP como a razão entre verdadeiros positivos e a soma entre verdadeiros TP e falsos positivos FP, ou seja, no caso do YOLO seria o número de vezes que o algoritmo acertou dividido pelo número de vezes que o algoritmo identificou algum objeto. A Fig. 5 (retirada do

documento oficial [7]) mostra essa melhora considerável de acurácia. Enquanto que o *YOLOv4* obteve mAP superiores a 42%, o *YOLOv3* não alcançou 34% para o teste *MS COCO*.

Ao comparar o *YOLO* aos demais modelo, pode-se perceber que ele é muito mais rápido. A versão 4 do modelo manteve esta velocidade chegando a processar 120 frames por segundo. Porém, neste projeto, o indicador para velocidade do algoritmo não será o FPS, já que não serão utilizadas GPUs para a identificação de imagens e não será necessária a identificação de um vídeo, mas de apenas uma foto. Portanto, o tempo de processamento médio será utilizado para esta avaliação. O tempo de processamento é a diferença entre os *timestamps* de início e de fim de processamento.

Caso o leitor deseje obter mais informações do *YOLOv4* e de suas versões anteriores, elas podem ser encontradas em [23], [7] e [22], ou no site oficial (`http://pjreddie.com`).

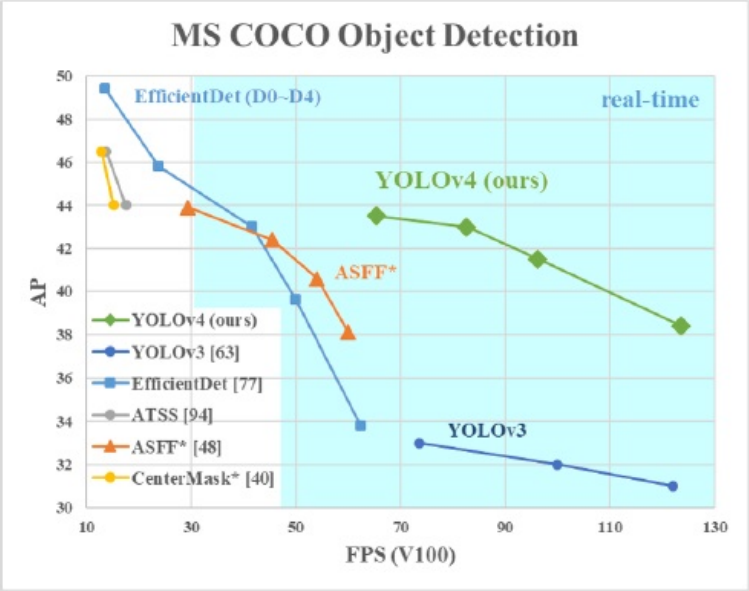


Figura 5: Comparação de desempenho (extraído do documento oficial do YOLOv4 [7]).

3.4 Treinamento da Rede

Para adicionar um novo produto ao banco de dados são feitas fotos de vários exemplares de produtos de uma mesma classe e posicionados de forma distinta nos padrões já estabelecidos na secção de requisitos. As imagens devem ter anotações sobre a segmentação de objetos e suas respectivas classes. Para isso, utiliza-se o *software LabelImg* [28] que gera um arquivo .txt com as posições do centro das caixas, suas dimensões e classes. A segmentação do software é feita de forma manual pelo usuário na interface gráfica disponível.

Para se treinar a rede neural foi utilizado a plataforma do *Google Colab* que é um *jupyter notebook* em nuvem para programação em Python. A plataforma permite o uso de GPUs de forma gratuita agilizando o processo. O banco de dados é armazenado no próprio google drive e os pesos calculados no algoritmo de treinamento também são salvos na nuvem.

3.5 Segmentação Automática

Para facilitar o processo de treinamento, foi desenvolvido um algoritmo de *region growing* (RG) que segmenta um objeto em uma bandeja de cor uniforme em diversos *frames* de um vídeo e já prepara todas as amostras para o treinamento. Desta forma, apenas é necessário alguns vídeos do produto a ser adicionado e a checagem manual pós-processamento para se ter certeza que nenhum *frame* foi segmentado erroneamente.

O RG é um algoritmo recursivo que procura por pontos candidatos de um determinado grupo [26, 27]. O algoritmo começa por um ponto seed aleatório e expande checando se os *pixels* vizinhos pertencem a um grupo candidato ou não. Para se obter estes pixels candidatos, é necessária uma foto de calibração da bandeja. Para se ignorar qualquer efeito que a variação da iluminação local possa causar (como sombras, fontes de luz não difusas e mudanças na intensidade luminosa), os valores de RGB são normalizados. A equação

$$R_{norm} = \frac{100 \cdot R}{\sqrt{R^2 + G^2 + B^2}} \tag{12}$$

demonstra a normalização da cor vermelha.

A partir destes valores normalizados, as médias μ e os desvios padrões σ da imagem são calculados. Considerando o sistema RGB como uma base ortogonal e a distribuição de cor dos pixels (RGB) normal. Para se definir quais pontos são candidatos e quais são parte da bandeja, o intervalo $[\mu - 3\sigma, \mu + 3\sigma]$ é definido dada que a probabilidade de falsos positivos de cada pixel (detectar como parte objeto quando não é) será baixa. As equações

$$z = \frac{R - \mu_R}{\sigma_R} = 3 \tag{13}$$

e

$$P(\text{identify\_as\_obj}|\text{is\_tray}) = 0.0016 \tag{14}$$

demonstram que a probabilidade de um pixel que faz parte do objeto ser classificado como parte da bandeja é inferior a 1%. Seja z a variável para o cálculo da curva normal e R a quantidade de vermelho do pixel, mas as mesmas equações são aplicáveis para o azul e verde do RGB.

Para que não ocorram falsos negativos é ideal se ter um contraste elevado entre a cor do produto e da bandeja.

Após se encontrar os grupos candidatos, o Algoritmo 1 elimina aqueles que estão na borda do *frame*, pois não fazem parte do objeto segmentado.

3.6 Servidor em Nuvem

Uma maneira de implementar um servidor em nuvem é com o uso do *AWS Elastic Beanstalk* [2]. Uma das formas desta implementação é com o uso do *Docker* [3]. O Docker é um serviço de virtualização de pacotes em forma de contâiner. Cada container é independente e capaz de rodar o código programado junto com suas bibliotecas. Portanto, há uma padronização de ambiente quando se utiliza o docker. As mesmas bibliotecas, programas e configurações são rodadas independente do ambiente (seja produção ou desenvolvimento).

Algorithm 1: Segmentação com regionGrowing

```
def regionGrowing(seed):
    neighbor = seed.get_neighbors()
    while i < n_neighbors:
        if neighbor[i].is_valid?():
            group.append(neighbor[i])
            neighbor[i].in_group!()
            regionGrowing(neighbor[i])
        else:
            neighbor[i].not_in_group!()
```

Um *Dockerfile* deve ser configurado especificando todas as versões de bibliotecas e processos a serem rodados. Ao obter sucesso em rodar o container localmente, o mesmo deve rodar em um servidor remoto. Para isto, o *eb client* da AWS deve ser instalado pelo terminal do desenvolvedor. O serviço da AWS faz o upload e configuração do serviço de forma automática, disponibilizando um servidor Apache junto a uma URL de acesso e diversos serviços para o monitoramento da saúde da aplicação.

4 Características do Projeto

Nesta seção serão apresentadas as características definidas do projeto - desde diagramas explicitando as lógicas por trás do sistema, até os requisitos do projeto - para um melhor entendimento por parte leitor.

4.1 Requisitos do Projeto

O projeto visa ser implementado no cenário brasileiro, dentro de uma confeitaria ou de uma padaria convencional. Os requisitos de projeto foram definidos dentro do contexto apresentado nos casos de uso da Fig. 6.

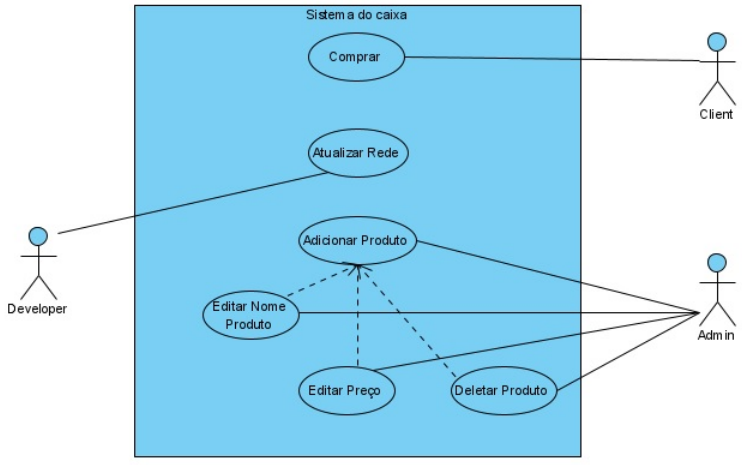


Figura 6: Casos de uso do caixa.

- O *software* deve funcionar de forma automática sem intervenção de um funcionário da loja.
- No local haverá uma webcam conectada a um computador com acesso à internet.
- A *webcam* deve sempre se manter na mesma posição a uma altura longe o suficiente para captar a bandeja inteira, mas deve estar perto o suficiente para conseguir reconhecer os produtos, para a câmera testada esta altura deve ser de aproximadamente 30 cm.
- O caixa deve possuir uma estrutura mecânica (como esquematizado na Fig. 7) estável de modo que a bandeja não saia do enquadramento da câmera.
- A iluminação local deve ser preferencialmente difusa e, em caso de uso de lâmpadas na loja, a luz deve ser branca.
- Os produtos serão posicionados pelo cliente em uma bandeja de cor uniforme em uma posição sempre igual e à uma distância da câmera sempre igual.
- Os produtos devem ser esteticamente padronizados.
- Múltiplos produtos podem ser colocados na bandeja sem sobreposição com uma distância mínima de 4 cm de espaçamento.
- Apenas objetos do catálogo da loja poderão estar presentes na bandeja durante o processo de identificação da compra.
- Após o cliente posicionar a bandeja no local indicado, o software deve responder com os produtos comprados e o preço calculado em menos de 5 segundos.
- O *software* deve ser robusto o suficiente para suportar pequenas variações na iluminação local e na estética do produto.
- O *software* deve ter uma mAP superior a 95%.

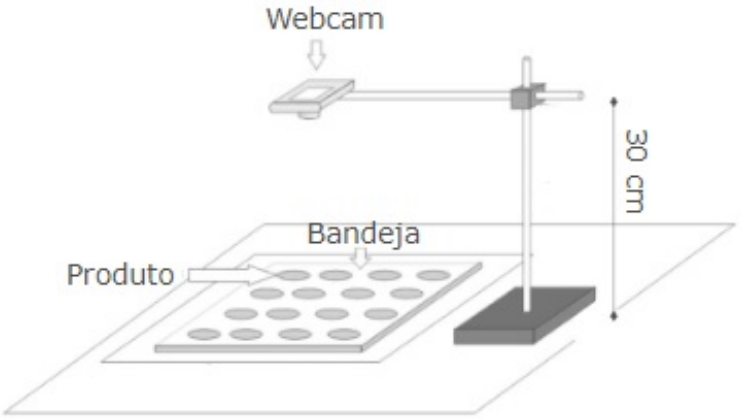


Figura 7: Esquema do caixa utilizado para realização de testes.

4.2 Definição dos Produtos para Testes

Para esta pesquisa, foram definidos 10 produtos diferentes para a realização dos testes de identificação. Foram escolhidos possíveis produtos de uma padaria, doceria ou mercearia de forma que fosse possível cobrir todos os casos de testes considerados importantes, como produtos de formato parecido como o pão de queijo e o choux cream, produtos de cor parecida como a maçã e a coca-cola e produtos de diferentes gêneros como os que vem em uma embalagem com rótulo e os produtos mais caseiros. Segue abaixo a lista definida:

- Coca-cola
- Guaraná
- Coxinha
- Pão francês
- Bomba de chocolate
- Pão de queijo
- Choux cream
- Banana
- Maçã
- Chá mate

4.3 Especificações das Máquinas Locais

Antes de iniciar o desenvolvimento do projeto, é necessário levar em consideração as especificações das máquinas que serão utilizadas. Muitas vezes são elas quem definirão algumas escolhas de soluções durante o desenvolvimento do projeto, uma vez que a eficiência de uma solução pode variar para diferentes configurações. Assim, as especificações da máquina que será utilizada para processar todas as informações foram definidas.

As especificações da máquina utilizada para o servidor local são:

- Processador: Intel(R) Core i7-3517U CPU
- Mémoria (RAM): 8.00 GB

4.4 O Processo de Compras

O projeto foi desenvolvido pensando em um modelo de *self-service*, comum em docerias. Assim, a ordem de processos para um consumidor finalizar a sua compra em um estabelecimento com o caixa automatizado seria da seguinte forma:

1. Caminhar pela loja e inserir todos os itens que deseja comprar sobre a sua bandeja de forma que não fiquem sobrepostos;
2. Dirigir-se ao caixa e inserir a bandeja com seus produtos no local sinalizado;
3. Pagar o valor calculado pelo sistema
4. Retirar a bandeja com sua compra e se dirigir para o local de consumo;

A máquina de estados ilustrando melhor o funcionamento dos caixas é apresentada na Fig. 8.

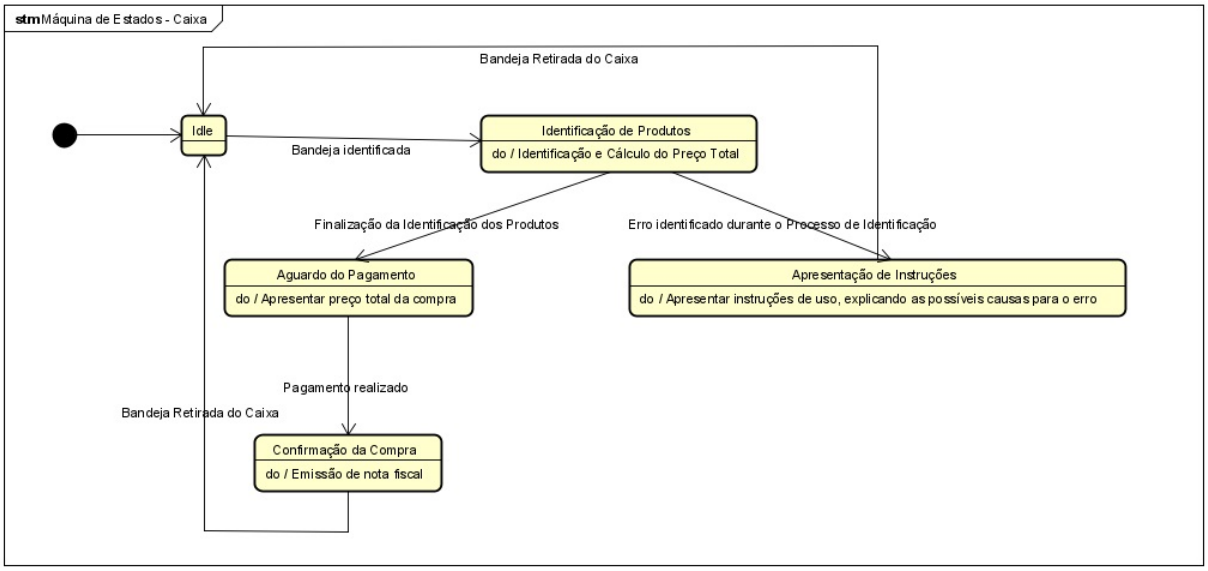


Figura 8: Máquina de estados do caixa.

4.5 A Inclusão de Novos Produtos

Com o objetivo de garantir a praticidade do adminsitrador da loja e o bom desempenho da rede foi criado um protocolo para a preparação das entradas para a alimentação da rede. Segue abaixo as regras definidas:

- Deve-se utilizar dois itens ou mais do mesmo produto para o preparo das fotos;
- Cada item deve possuir 50 fotos, contabilizando 100 fotos por cada tipo de produto;
- As imagens devem ser retiradas através dos frames de um vídeo;
- O vídeo deve ser feito através da interface do administrador e deve ser enviado automaticamente para um servidor remoto.
- Durante o vídeo, a bandeja deve estar realizando um movimento circular de tal forma que garanta uma variação de sombreamento sobre o produto;

- A bandeja utilizada durante a adição de produto deve ser a mesma utilizada pelos clientes durante o proceso compra;
- A bandeja deve estar bem iluminada durante as filmagens;

5 Estudo das Tecnologias

Antes de iniciar a implementação do projeto, foi realizada uma série de estudos das ferramentas e das tecnologias disponíveis a fim de entender como elas funcionam e como poderiam ser utilizadas no sistema. Nesta seção serão apresentadas desde as etapas iniciais, onde foram realizados testes com o YOLO, até o estudo do comportamento da rede para situações consideradas como "extremas" - situações em que um ou mais dos requisitos de operação especificados no item 4.1 não foram atendidos.

5.1 Testes Iniciais

Foram utilizadas duas abordagens para se criar o banco de dados. A primeira foca na quantidade de imagens e a segunda na qualidade. Enquanto que na primeira foram extraídas 150 fotos de pão francês do google, na segunda foram tiradas 20 fotos de pão francês seguindo os padrões convencionados na secção de requisitos.

As Figs. 10 e 9 representam o estágio da preparação de uma das entradas para o treinamento do modelo, utilizando o *software LabelImg* [28], para identificar objetos da classe pão francês. A Fig. 9 faz parte do banco de dados de treinamento seguindo a primeira abordagem cujas imagens são retiradas da internet, e portanto, são mais poluídas com outros elementos, existem por vezes mais de 3 exemplares do objeto a ser identificado na mesma imagem e as dimensões e resoluções variam de imagem para imagem. Já a Fig. 10 é um exemplo de imagem seguindo a segunda abordagem com o fundo limpo e apenas 1 produto na bandeja.

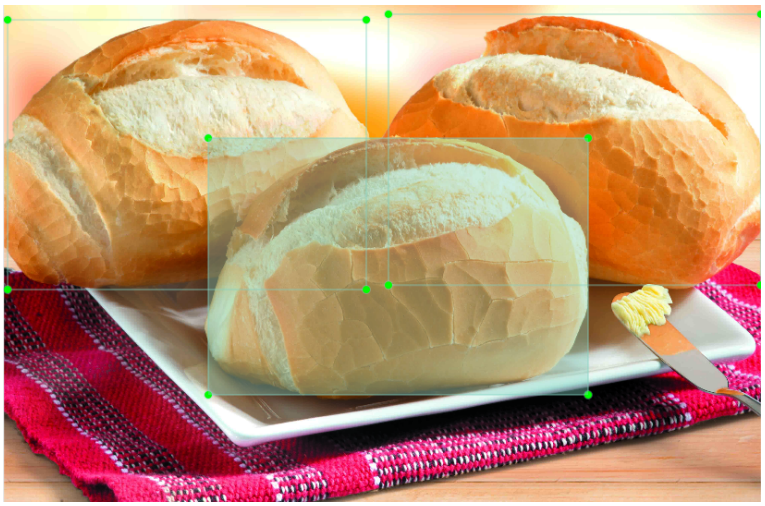


Figura 9: Preparação das entradas para o treinamento do modelo pela abordagem quantitativa.

Figura 10: Preparação das entradas para o treinamento do modelo pela abordagem qualitativa.

Após o treinamento finalizar, algoritmo não funcionou para a abordagem quantitativa, mas obteve resultados satisfatórios para a abordagem qualitativa. A Fig. 11 ilustra um caso de sucesso ao classificar uma imagem retirada da internet.

Figura 11: Resultado do teste inicial para a identificação de um objeto da classe pão francês.

5.2 Novos Resultados com Melhora do Banco de Dados

Analisando os resultados obtidos nos testes iniciais, decidiu-se aumentar a quantidade de fotos por produto a fim de melhorar a precisão do modelo. Entretanto, a forma como a segmentação estava sendo realizada inicialmente - de forma manual com a ajuda do software Labellmg - tornava este incremento inviável, uma vez que o processo é demorado. Dessa forma, optou-se por desenvolver um software que realizasse a segmentação de forma automática, com o objetivo de diminuir a necessidade de mão de obra e o tempo necessário para tal processo.

A aplicação foi desenvolvida utilizando o algoritmo descrito no item 3.5 “Segmentação Automática” como base. Antes de realizar a segmentação, um vídeo deve ser gravado e dividido em frames. Após a obtenção dos frames, basta o usuário indicar quais fotos deseja segmentar e qual classe o produto presente pertence para o software gerar todos os arquivos necessários para o treinamento da rede - que será utilizado para a identificação de produtos futuramente. Além disso, foi definido um protocolo - descrito no item 4.5 - para garantir um número de fotos e de variações de luminosidade considerado como recomendável para cada produto.

A taxa de acerto obtida durante um teste composto por setenta imagens ao todo foi de 93%. Entretanto, o tempo médio gasto para o servidor receber a imagem, processar as informações e devolver ao cliente foi de aproximadamente 10 segundos no pior dos casos, utilizando o TensorFlow.

A Fig. 12 e Fig. 13 são exemplos dos resultados obtidos do processo de classificação:

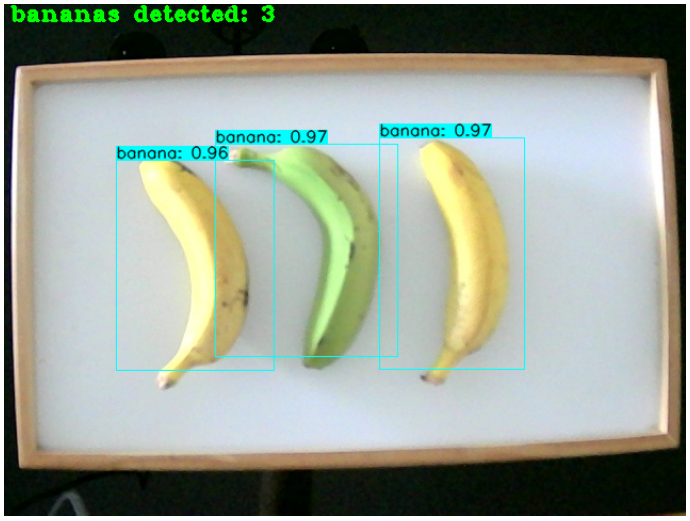


Figura 12: Resultado da identificação de um objeto da classe banana.

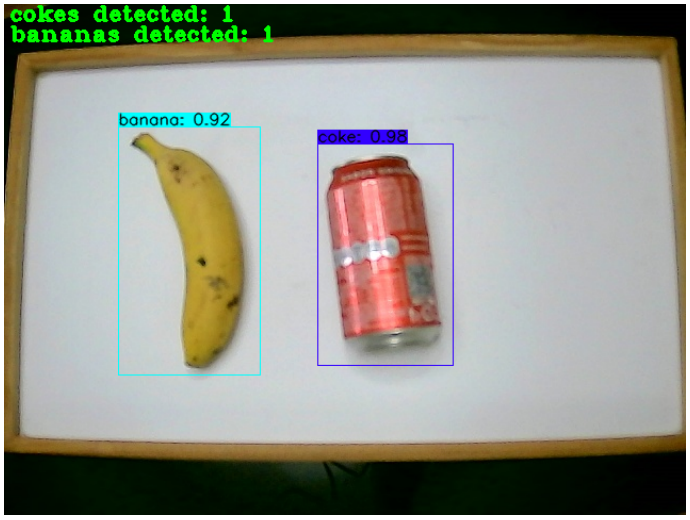


Figura 13: Resultado da identificação de um objeto da classe banana e coke.

Como pode-se observar, o sistema consegue identificar tanto a classe, quanto a quantidade de cada produto presente na bandeja.

5.3 Comparação entre Bibliotecas

As principais preocupações para o projeto eram tanto a precisão, quanto o tempo necessário para a identificação dos objetos sobre uma bandeja. Dessa forma, foi feito uma comparação entre o Tensorflow e o OpenCV a fim de encontrar a melhor solução para atender os requisitos de projeto [item 4.1] de acordo com as especificações do sistema [item 4.3]. Foram utilizadas cinco imagens com quantidade de objetos diferentes para obter um melhor mapeamento.

Tabela 2: Comparação de tempo de classificação entre OpenCV e TensorFlow

Quantidade de objetos	OpenCV [s]	TensorFlow [s]
1	0.11	3.60
2	0.09	3.14
3	0.09	2.40
4	0.11	2.35
6	0.11	2.60

Analisando a Tabela 2, conclui-se que para as especificações do sistema utilizado [4.3], o OpenCV é a melhor opção, considerando o tempo de processamento. Existe também a opção do TensorFlow Lite, que exige menor capacidade de processamento. Entretanto, esta escolha foi eliminada logo de início, uma vez que há uma perda expressível de precisão do modelo [1] em comparação ao TensorFlow e o OpenCV. Dessa forma, foi escolhido utilizar o OpenCV para se utilizar no projeto.

5.4 Casos Excepcionais - Objeto Desconhecido Sobre a Bandeja

Seja por desatenção ou por conveniência, é muito comum durante as compras que o cliente deixe alguns de seus pertences pessoais sobre a bandeja. Pode-se observar logo na Fig. 14 que a banana é reconhecida pelo sistema, e o celular não é classificado, atendendo à necessidade de evitar cobranças por itens que não constam no catálogo da loja.

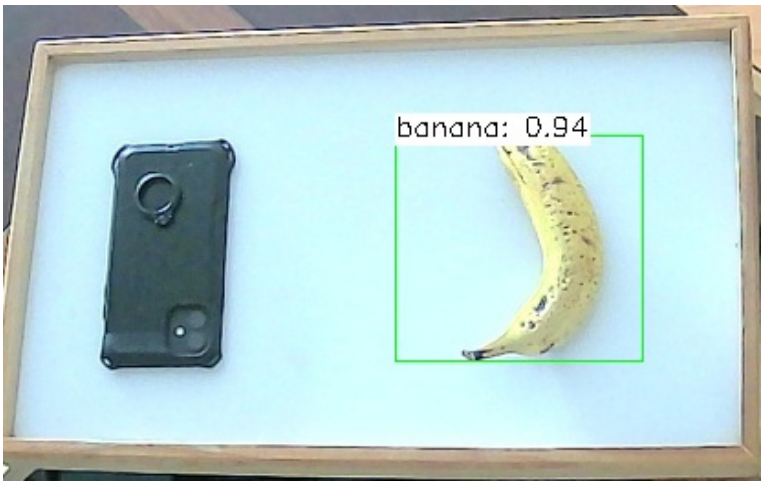


Figura 14: Bandeja com um produto e um objeto desconhecido - celular e banana.

Entretanto, em alguns casos, o sistema classificou um objeto desconhecido como um dos itens do catálogo da loja. A Fig. 15 ilustra um caso em que um celular - objeto desconhecido pelo sistema - foi classificado como guaraná - item do catálogo da loja. Portanto, optou-se por deixar como requisito que o cliente não pode deixar nenhum objeto não catalogado em cima da bandeja durante a compra.

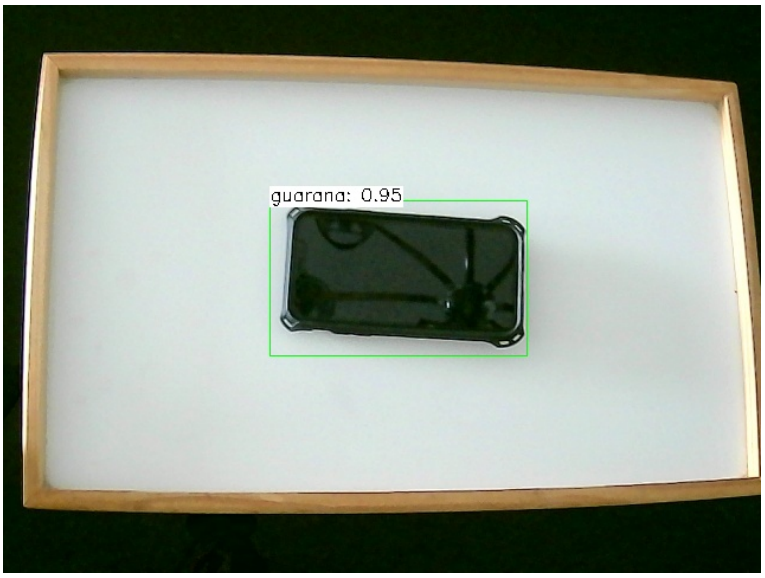


Figura 15: Celular identificado como guaraná pelo sistema.

5.5 Casos Excepcionais - Objetos Próximos

Quando não há um espaçamento expressivo entre cada produto sobre a bandeja, em alguns casos há a correta classificação e segmentação dos produto, isto normalmente ocorre quando há um contraste grande de cor entre os produtos. Porém em outros casos, percebeu-se uma dificuldade do sistema para reconhecê-los. Em casos em que a banana é posicionada na bandeja ainda no cacho (veja a Fig. 16) o algoritmo não reconhece o conjunto por possuir uma forma distinta da imagem da banana individual. Por isso, decidiu-se colocar como requisito que os objetos deve ter um espaçamento mínimo de 4 cm entre si na hora da compra.

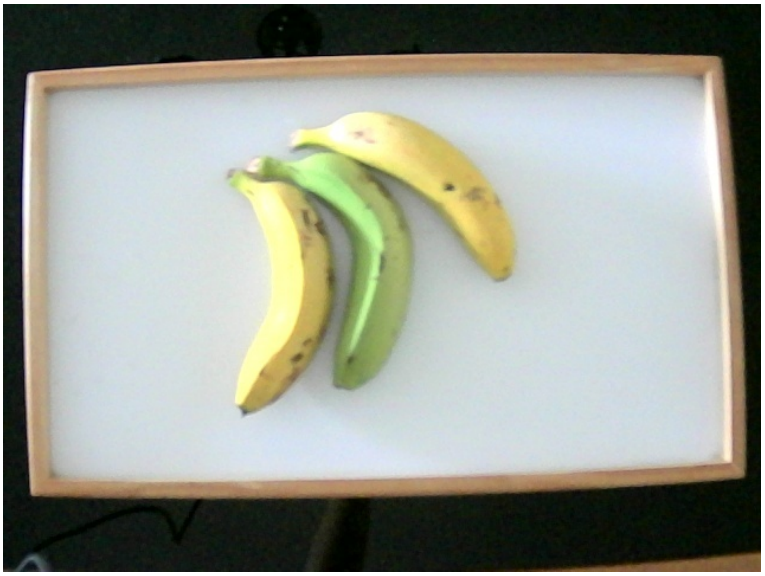


Figura 16: Bananas muito próximas não reconhecidas pelo sistema.

Tanto o caso do objeto desconhecido na bandeja, quanto o caso de objetos muito próximos um do outro poderiam ter uma pequena melhora caso a rede neural fosse treinada com maior variabilidade de imagens. Porém, este problema não seria resolvido por se tratar de uma limitação da própria rede. Como constatado ao se realizar os testes iniciais com o modelo pré treinado do YOLO, o modelo tem dificuldades de identificar que um objeto não faz parte da sua lista de objetos conhecidos, ao invés disso, ele identifica o objeto como algum outro que lhe é conhecido e parecido com o objeto da imagem. O algoritmo também encontra dificuldades em identificar e contar objetos próximos ou sobrepostos, muitas vezes identificando o conjunto de objetos como um objeto apenas ou nem identificando.

6 Implementação do Projeto

Esta seção tem como objetivo expor o sistema definido para realizar o projeto, apresentando uma breve explicação de cada estrutura e da própria solução como um todo. Além disso, também será exibido um fluxo de dados a fim de facilitar o entendimento da lógica por trás do sistema.

6.1 Arquitetura

Como pode-se observar na Fig. 17, a solução desenvolvida é composta por quatro sistemas principais: a máquina do desenvolvedor, o servidor na nuvem, o servidor local e a interface do usuário.

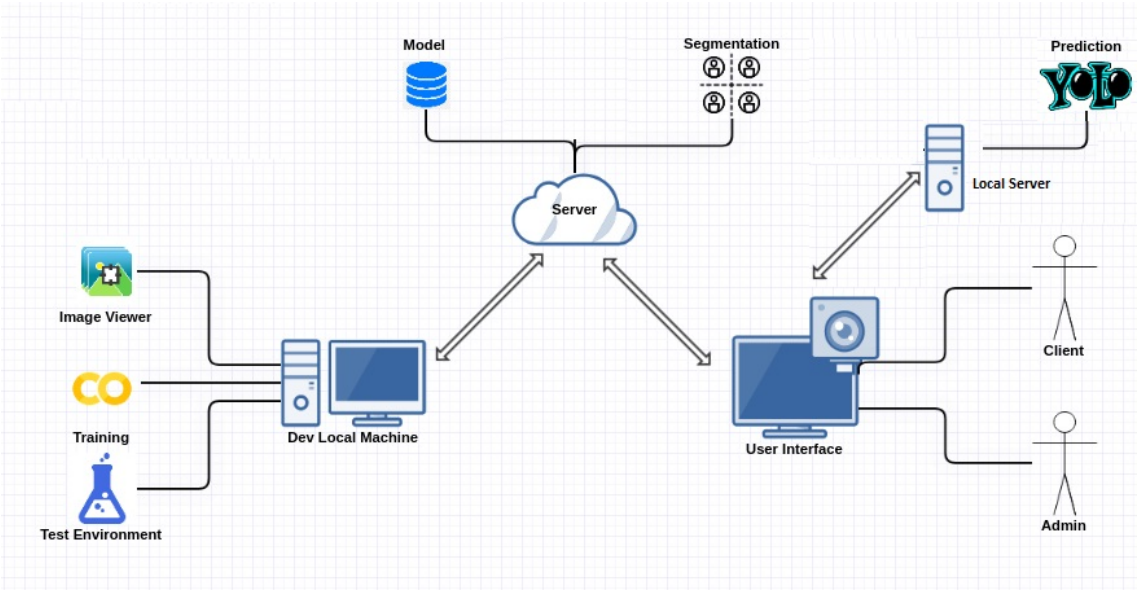


Figura 17: Arquitetura da solução definida ao projeto.

- A máquina do desenvolvedor é responsável por todas as ações realizadas pelo técnico: treinamento do modelo, visualização das imagens obtidas como entradas para o treinamento, atualização do modelo do cliente na nuvem e um ambiente de teste.
- O servidor na nuvem é responsável por armazenar as imagens dos produtos de cada loja, pela segmentação das imagens recebidas pelo administrador na hora de adicionar um novo produto ao cardápio da loja, e como um mediador na troca de dados entre o servidor local do estabelecimento e a máquina do desenvolvedor. O servidor remoto está hospedado no serviço *AWS Elastic Beanstalk*.
- O servidor local faz a identificação dos itens de compras presentes nas fotos recebidas pela interface do usuário. Além disso, ele também calcula o preço total da compra e retorna à interface do usuário. Vale ressaltar que o servidor local possui sempre a última versão mais atualizada do modelo da rede treinado e do banco de dados com as informações de todos os produtos da loja.
- Já a interface do usuário é responsável em obter todas as ações do cliente e do administrador e apresentá-los as respostas geradas.

Um problema enfrentado foi para se desenvolver uma arquitetura que fosse prática para o treinamento e ao mesmo tempo rápida para a identificação do produto. Este problema ocorre, pois o arquivo .weights é muito grande impossibilitando a transferência do mesmo do servidor em nuvem para o repositório local via um protocolo HTTP. Uma possibilidade levantada foi o envio da imagem para o servido remoto para que o YOLO remotamente fizesse a predição. Porém, esta arquitetura torna o processo mais lento, já que aumenta a quantidade de tráfego de dados. Portanto, a solução encontrada foi a de o administrador da conta baixar manualmente o arquivo weights e atualizar na pasta local.

6.2 Fluxo de Dados

Para melhor ilustrar o fluxo de dados e a interação entre as várias partes da aplicação com o cliente e o admin foram desenhados dois diagramas de sequência. O primeiro descrevendo o caso de uso de compra (veja a Fig. 18) e o segundo o caso de uso de criação/atualização de produtos (veja a Fig. 19).

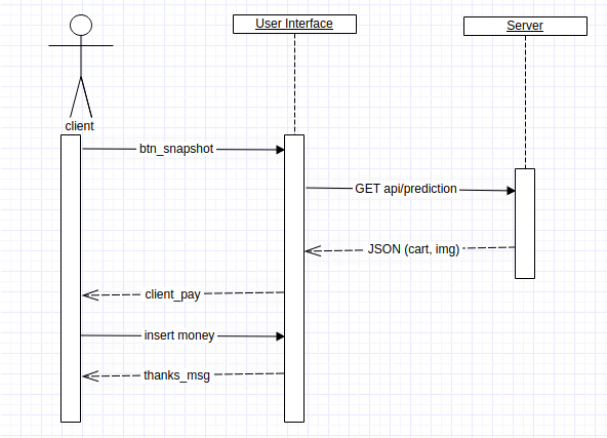


Figura 18: Diagrama de sequência para a compra do produto.

No caso de uso de compras, vide Fig. 18, o cliente pressiona o botão para tirar a foto, a interface do usuário envia um JSON com a imagem para o servidor local que faz a predição e envia a resposta com o preço total a pagar para o usuário novamente.

Já no caso de uso de adição de edição dos produtos, vide Fig. 19, o administrador, após receber a lista de produtos existentes, pode escolher se quer editar ou adicionar um produto. E então, pode enviar pela interface do usuário para o servidor remoto uma foto de calibração que é armazenada no banco de dados. Em seguida, o administrador grava um vídeo que é enviado frame a frame por meio de requisições, "*POST*"se for um novo produto ou "*PUT*"se for um já existente, à "*REST API*"do servidor remoto. Por último, o desenvolvedor que acessa o servidor remoto de outra máquina pode fazer requisições "*GET*"para obter as imagens segmentadas.

6.3 Banco de Dados

Como o objetivo final deste projeto não é criar um banco de dados otimizado, mas apenas um funcional para registrar e obter informações dos produtos, o modelo utilizado foi definido da seguinte maneira:

- Para cada loja, o sistema como um todo possui dois banco de dados: no servidor local, e no servidor localizado na nuvem.
- O banco de dados local possui uma tabela onde armazena o nome do produto e preço do produto.
- O banco de dados na nuvem possui uma tabela com o nome do produto, preço do produto e localização (*path*) de onde as fotos dos itens da loja foram armazenados.

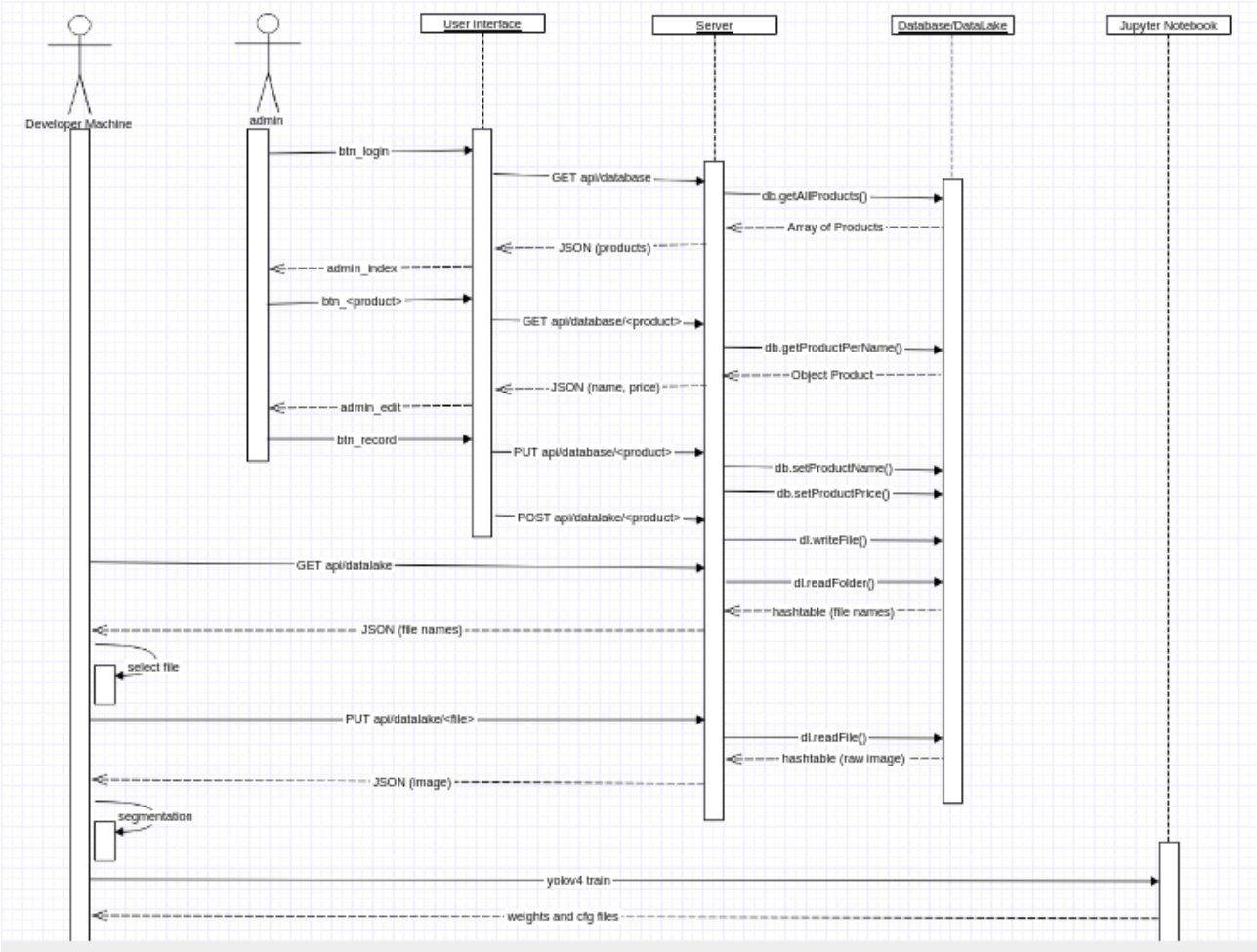


Figura 19: Diagrama de sequência para criação de produto.

Para o armazenamento da imagens e arquivos "*.txt*" em nuvem, seria utilizado um "*data lake*". Porém, durante esta pesquisa, os testes foram feitos em servidores locais e com armazenamento em pastas locais.

6.4 Interface do Usuário

Foram criadas duas interface de usuário: uma para os clientes e outra para o administrador da loja.

- Interface para o administrador: concede ao administrador da loja o direito de verificar e editar as informações de todos os produtos (preço e nome), deletar um produto e adicionar um novo item ao banco de dados da loja.
- Interface para o cliente: permite que o usuário consiga realizar a compra de produtos, identificando os itens presentes em sua bandeja e calculando o valor total da compra.

6.5 Preparação das Entradas para Treinamento do Modelo

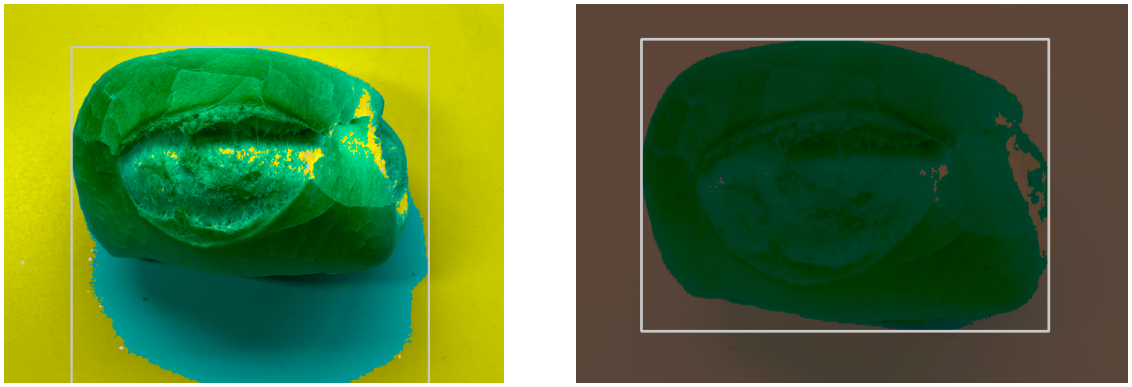
O *software* de segmentação de imagens inicialmente levava em consideração a luminosidade de cada ponto, porém, tal estratégia era falha (veja a Fig. 20 a), pois as sombras e os pontos de luz acabavam por atrapalhar na segmentação do objeto. Por este motivo, a expressão (12) foi empregada para remover qualquer efeito de intensidade luminosa (veja a Fig. 20 b).

Para se encontrar a média e o desvio padrão de cor da bandeja e poder segmentar os objetos da forma correta, é necessária uma calibração. Desta forma, toda vez que um novo produto é adicionado ou novas imagens são adicionadas a um banco de dados do produto existente, o administrador da conta deve fazer uma foto da bandeja. Após a calibração, o administrador deve gravar um vídeo do produto em cima da bandeja de forma que o produto esteja centralizado na foto e não haja pixels do produto que não estejam envoltos por pixels da bandeja.

E então, o aplicativo do lado do administrador da conta gera uma série de imagens dos produtos do cardápio da loja a partir de vídeos gravados pelo administrador. Estas fotos são enviadas para o servidor remoto que as processa e salva tanto a imagem e quanto um arquivo ".txt". O desenvolvedor pode extrair todos estes dados com o uso de um programa simples que faz uma requisição HTTP get para extrair cada arquivo (em formato JSON). O técnico ainda precisa analisar as imagens e checar se todas foram segmentadas da forma correta. Após este processo, o técnico deve juntar todas as pastas para alimentar o YOLO.

6.6 Treinamento do Modelo

O treinamento da rede é realizado dentro do ambiente do *Google Colab* pelo desenvolvedor através de um arquivo *notebook* desenvolvido propriamente para esta função. Para treinar o modelo, o técnico deve informar a quantidade total de classes (produtos) e utilizar as saída obtidas da aplicação desenvolvida citada no item acima para alimentar a rede. Após terminar o treinamento, será gerado um arquivo "*yolov4.weights*" com os pesos necessários para a classificação dos produtos.



(a) (b)

Figura 20: Segmentação por meio do *Region Growing* (a) de um objeto com sombra e (b) com a sombra removida.

7 Resultados

Nesta seção serão apresentados os resultados obtidos após a implementação do sistema ser finalizada, onde serão verificados se todos os requisitos do sistema [item 4.1] foram cumpridos, e se há a presença de falhas durante o processo de classificação dos produtos.

7.1 Tempo de Resposta do Sistema Final

Para avaliar o tempo de resposta do sistema, foi calculado o tempo necessário para o servidor receber a foto da bandeja, processar e devolver o preço total ao cliente. Assim, foram realizados cinco testes, variando o número total de objetos por bandeja, para se obter uma média do tempo de resposta. A Tabela 3 apresenta os resultados obtidos.

Tabela 3: Tempos de resposta obtidos para bandejas com quantidades diferentes de produtos

Quantidade de objetos	1	2	3	4	6
Tempo de resposta [s]	2.95	3.08	3.85	3.09	2.92

Como pode ser observado na Tabela 3, o maior tempo de resposta obtido durante os teste foi de 3.09 segundos, atendendo ao requisito de projeto de 5 segundos de resposta [item 4.1] com uma margem de 1.91 segundos.

7.2 Identificação e Classificação de Objetos

A métrica utilizada para avaliar o modelo de classificação foi o mAP (*mean Average Precision*), que apresenta uma média das precisões de todas as classes presentes no sistema [25]. Dessa forma, em um conjunto de teste de 85 fotos, o modelo do sistema obteve um mAP de 92.73%.

Quando todos os requisitos de operação do sistema são atendidos, o sistema de classificação obteve bons resultados (taxa de 96,25% de acerto). A Fig. 21 e a Fig 22 são exemplos de classificações de objetos que ocorreram com sucesso:

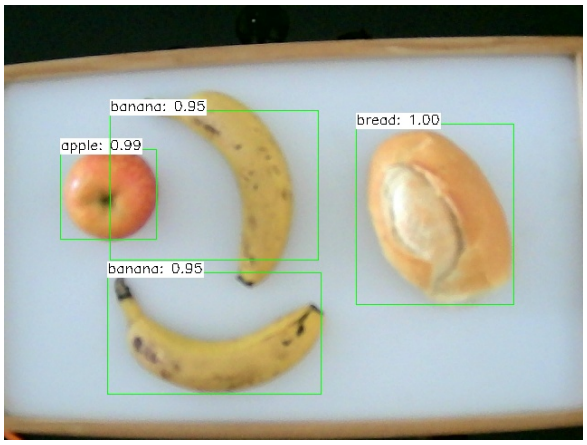


Figura 21: Identificação da maçã, das bananas e do pão realizada com sucesso.

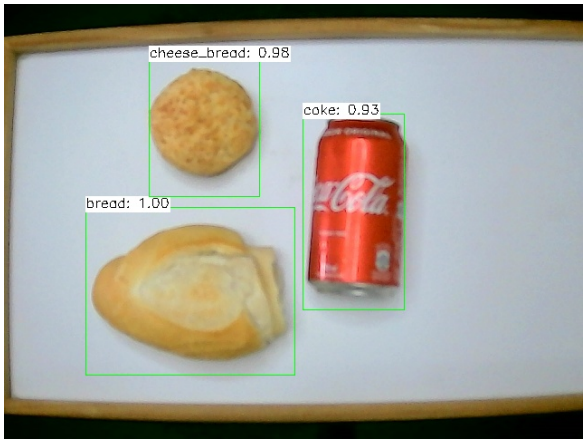


Figura 22: Identificação do pão de queijo, do pão e da coca-cola realizada com sucesso.

7.3 Falha na Distinção de Produtos com Cores e Formatos Semelhantes

Ao testar o algoritmo com produtos de cores, formatos e texturas parecidas, muitos dos caso obteve-se sucesso como mostra a Fig. 25. Porém, em outros casos o sistema apresentou falhas. Os erros ocorreram entre dois grupos de produtos: eclair e guaraná (Fig. 24), choux cream e coxinha e choux cream e pão de queijo (vide Fig. 23). Tal fato deve-se provavelmente às cores predominantes de ambos os casos serem muito parecidas, aos seus formatos semelhantes do ângulo em que a foto é captada e à baixa definição da câmera utilizada. Outra hipótese válida seria a falta de variedade de cada produto para a alimentação do modelo.

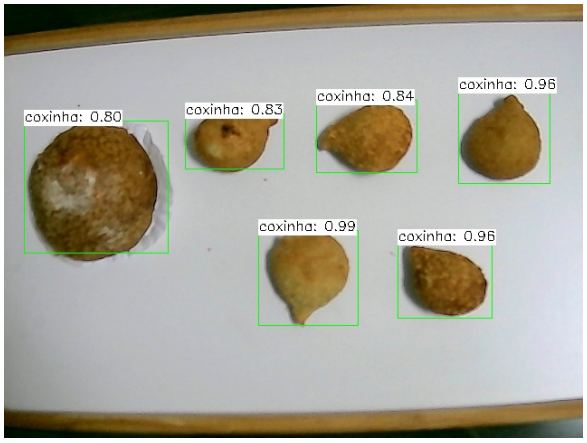


Figura 23: Choux cream classificado como coxinha.

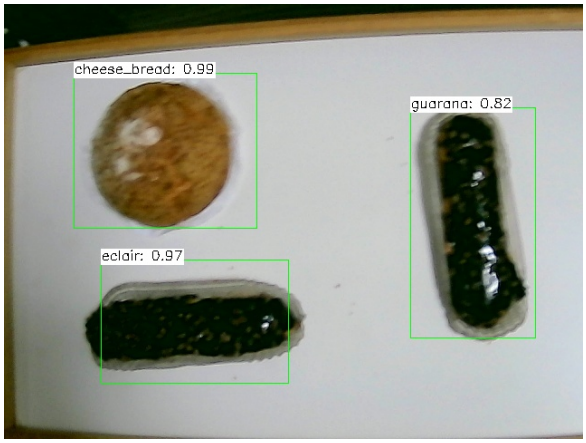


Figura 24: Eclair classificado como guaraná e choux cream classificado como pão de queijo

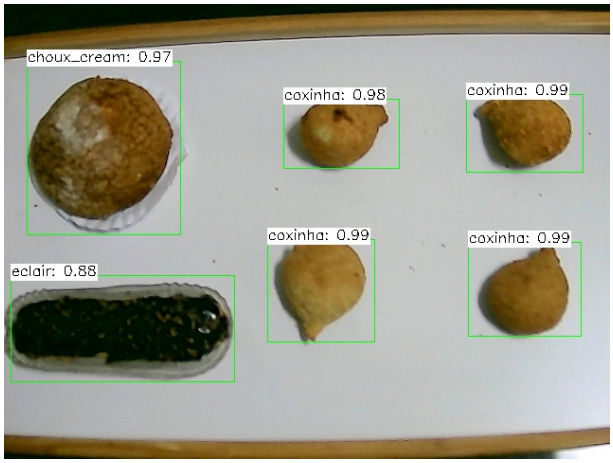


Figura 25: Choux cream e outros produtos classificados de forma correta.

8 Conclusão

Durante esta pesquisa, foi possível concluir que a melhor maneira de se treinar a rede neural é aplicando um treinamento com imagens mais bem selecionadas. Portanto, não é uma boa estratégia extrair da internet as imagens de forma randômica. Mas, sim, produzir as fotos em um ambiente controlado com uma variabilidade em cada amostra suficiente apenas para tornar o algoritmo robusto para a situação proposta. Desta forma, o treinamento foi feito com um produto de cada vez em um fundo limpo de cor uniforme. Para se obter uma grande quantidade de fotos nestas condições, foi desenvolvida uma técnica para facilitar o treinamento do algoritmo com a extração de frames do vídeo, a aplicação da segmentação automática (*region growing*) e o uso de um servidor remoto.

Porém, ainda há alguns problemas que precisam ser corrigidos em uma pesquisa futura. Um deles é a dificuldade do algoritmo de diferenciar objetos parecidos. Este problema pode ser corrigido melhorando o banco de dados de treinamento. O outros dois, são a dificuldade do modelo de contar objetos sobrepostos e o problema do algoritmo de contar objetos não conhecidos indentificando-os erroneamente. Ambos os problemas foram resolvidos aumentando a quantidade de requisitos, mas precisariam ser resolvidos por meio de uma melhora no modelo da rede neural em uma pesquisa futura.

Apesar destes pequenos problemas, o algoritmo obteve um resultado bom com mais de 95% de acurácia e estaria pronto para ser testado em uma padaria ou doceria. Os próximos passos seriam a adaptação e o teste do projeto para um estabelecimento real para saber se o modelo e a arquitetura do projeto são de fato práticos e funcionais.

Referências

[1] Model optimization. https://www.tensorflow.org/lite/performance/model_optimization. [Online; accessed 16-Novembro-2020].

[2] Aws elastic beanstalk. https://aws.amazon.com/pt/elasticbeanstalk/, 2011. [Online; accessed 11-Novembro-2020].

[3] Docker engine. https://www.docker.com/, 2013. [Online; accessed 11-Novembro-2020].

[4] Keras documentation. https://keras.io/, 2015. [Online; accessed 02-Maio-2020].

[5] Relatório de cidadania financeira - parte 1: As pessoas estão acessando serviços financeiros? https://www.bcb.gov.br/nor/relcidfin/cap01.html, 2018. [Online; accessed 05-Maio-2020].

[6] Qr codes in retail: Does it help? https://www.beaconstac.com/qr-codes-in-retail, 2020. [Online; accessed 05-Maio-2020].

[7] Hong-Yuan Mak Liao Alexey Bochkovskiy, Chien-Yao Wang. Yolov4: Optimal speed and accuracy of object detection, April 2020.

[8] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 111–118, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

[9] Wen-Yuan Jenc Chia-Chen Chaoa, Jiann-Min Yangb. Determining technology trends and forecasts of rfid by a historical review and bibliometric analysis from 1991 to 2005. *Elsevier Technovation*, 2006. ISSN 0957-4174.

[10] Meilin Du. Examining the user experience of amazon go shopping — just walk out. https://blog.prototypr.io/examining-the-user-experience-of-amazon-go-shopping-just-walk-out-bfeda3c9ce39, 2018. [Online: accessed 05-Maio-2020].

[11] MiYoung Lee Eun Young Kim. An exploratory study of perceived benefits and risks for qr code based virtual fashion stores. *Korean Journal of Human Ecology*, 2013. ISSN 477–490.

[12] Annalisa Franco, Davide Maltoni, and Serena Papi. Grocery product detection and recognition. *Expert Systems with Applications*, 81:163–176, 2017. ISSN 0957-4174.

[13] Emanuele Frontoni, Paolo Raspa, Adriano Mancini, Primo Zingaretti, and Valerio Placidi. Customers' activity recognition in intelligent retail environments. In Alfredo Petrosino, Lucia Maddalena, and Pietro Pala, editors, *New Trends in Image Analysis and Processing – ICIAP 2013*, pages 509–516, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-41190-8.

[14] P. Radha Krishna Harikrishna G. N. Rai, Kishore Jonna. Video analytics solution for tracking customer locations in retail shopping malls. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, USA, 2011.

[15] Dang Ha The Hien. A guide to receptive field arithmetic for convolutional neural networks. https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807, 2016. [Online; accessed 02-Maio-2020].

[16] Luke McCathie Katina Michael. The pros and cons of rfid in supply chainmanagement. *IEEE*, 2005. ISSN 623-629.

[17] Ben Krose and Patrick Van Der Smagt. *An introduction to Neural Networks*. The University of Amsterdam, 8 edition, 1996.

[18] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. In *The handbook of brain theory and neural networks*, page 255–258, 1998.

[19] Roman Orac. What's new in yolov4? https://towardsdatascience.com/whats-new-in-yolov4-323364bb3ad3, May 2020. [Online: accessed 10-Jun-2020].

[20] Elif Ozkaya, H. Erkan Ozkaya, Juanita Roxas, Frank Bryantand, and Debbora Whitson. Factors affecting consumer usageof qr codes. *MACMILLAN PUBLISHERS LTD.*, 2015.

[21] Raúl Parada, Joan Melià-Seguí, Marc Morenza-Cinos, Anna Carreras, and Rafael Pous. Using RFID to detect interactions in ambient assisted living environments. *IEEE Intelligent Systems*, 30:16–22, 07 2015.

[22] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger, 2016.

[23] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[24] Bikash Santra and Dipti Prasad Mukherjee. A comprehensive survey on computer vision based approaches for automatic identification of products in retail store. *Image and Vision Computing*, 86:45–63, 2019. ISSN 0262-8856.

[25] Tarang Sarah. Measuring object detection models — map — what is mean average precision? https://towardsdatascience.com/what-is-map-understanding-the-statistic-of-choice-for-comparing-object-detection-mo 2018. [Online; accessed 15-Novembro-2020].

[26] M. S. G. Tsuzuki, F. K. Takase, M. A. S. Garcia, and T. C. Martins. Converting CSG models into meshed B-rep models using Euler operators and propagation based marching cubes. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 29(4): 337–344, 2007.

[27] M. S. G. Tsuzuki, A. K. Sato, E. K. Ueda, T. C. Martins, R. Y. Takimoto, Y. Iwao, L. I. Abe, T. Gotoh, and S. Kagei. Propagation-based marching cubes algorithm using open boundary loop. *Visual Computer*, 34(10):1339–1355, 2018.

[28] Tzutalin. Labelimg. https://github.com/tzutalin/labelImg, 2015. [Online: accessed 05-Maio-2020].

[29] Paul J Werbos. Backpropagation through time: What it does and how to do it. In *Proceedings of the IEEE*, volume 78, pages 1550–1560. IEEE, 1990.

[30] B. Wu, W. Tseng, Y. Chen, S. Yao, and P. Chang. An intelligent self-checkout system for smart retail. In *2016 International Conference on System Science and Engineering (ICSSE)*, pages 1–4, July 2016.

[31] David C. Wyld. 24-karat protection: Rfid and retail jewelry marketing. *International Journal of UbiComp (IJU)*, 2010. ISSN 623-629.

[32] S. Xu, A. Savvaris, S. He, H. Shin, and A. Tsourdos. Real-time implementation of YOLO+JPDA for small scale UAV multiple object tracking. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1336–1341, June 2018.