UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
ENGENHARIA ELÉTRICA

VINÍCIUS DOS SANTOS PINHEIRO

IDENTIFICAÇÃO DE LIXO HOSPITALAR POR MEIO DE IMAGENS DE *SMARTPHONES* COM *DEEP LEARNING*

Manaus
2023

VINÍCIUS DOS SANTOS PINHEIRO

IDENTIFICAÇÃO DE LIXO HOSPITALAR POR MEIO DE IMAGENS DE *SMARTPHONES* COM *DEEP LEARNING*

Projeto de pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Carlos Mauricio Serodio Figueiredo

Manaus
2023

Universidade do Estado do Amazonas – UEA
Escola Superior de Tecnologia - EST

Reitor:
André Luiz Nunes Zodahib
Vice-Reitor:
Kátia do Nascimento Coureiro
Diretora da Escola Superior de Tecnologia:
Ingrid Sammyne Gadelha Figueiredo
Coordenador do Curso de Engenharia Elétrica:
Israel Gondres Torné
Banca Avaliadora composta por: Data da defesa: <11/09/2023>.
Prof. Carlos Maurício Serodio Figueiredo, Dr. *(Orientador)*
Prof. Antonio Luiz Alencar Pantoja, Dr.
Prof. Jozias Parente de Oliveira, Dr.

CIP – Catalogação na Publicação

Dos Santos Pinheiro, Vinícius

Identificação de lixo hospitalar por meio de imagens de *smartphones* com *deep learning* / Vinícius dos Santos Pinheiro; [orientado por] Carlos Maurício Serodio Figueiredo – Manaus: 2023.
68 p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Universidade do Estado do Amazonas, 2023.

1. Resíduos Hospitalares. 2. Deep Learning. 3. Swift.
I. Figueiredo, Carlos Maurício Serodio.

VINÍCIUS DOS SANTOS PINHEIRO

IDENTIFICAÇÃO DE LIXO HOSPITALAR POR MEIO DE IMAGENS DE *SMARTPHONES* COM *DEEP LEARNING*

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheiro Eletricista.

Nota obtida: 10,0 (Dez vírgula zero)

Aprovada em 11/09/2023.

Área de concentração: Inteligência Artificial

BANCA EXAMINADORA

Orientador: Carlos Maurício Serodio Figueiredo, Dr.

Avaliador: Antônio Luiz Alencar Pantoja, Dr.

Avaliador: Jozias Parente de Oliveira, Dr.

Manaus
2023

Dedicatória

À todos aqueles que ainda ousam criar sua felicidade em um mundo que tenta controlar o que ela deve ser.

AGRADECIMENTO

Agradeço primeiramente a Deus por guiar os caminhos que sigo na minha vida.

Agradeço às pessoas que moldaram meu coração, Maria José e Júlio Quentino, por me ensinarem a ser verdadeiro e sempre me incentivaram a ser maior que minhas dificuldades, além de terem sido os melhores avós que eu poderia ter.

Agradeço ao Carlos Maurício, uma das minhas maiores fontes de inspiração e orientador, por, em uma palestra no momento em que estava profundamente infeliz com meu curso, reacender minha paixão pela engenharia e me fazer acreditar que as invenções vistas em filmes como "Homem de Ferro"podem se tornar realidade através do estudo. E claro, por me orientar.

Agradeço à minha companheira na jornada da vida, Ana Begnini, por me inspirar a ser a minha melhor versão todos os dias, além de ser parte dos melhores momentos que consigo lembrar.

Agradeço aos amigos que estiveram comigo na caminhada da vida e me fortaleceram nesse momento que aqui encerro. Ao Gustavo Guedes, por ter me ensinado a programar quando os nossos professores desistiram, tornando possível eu chegar onde cheguei. Também ao Caio Viga, a pessoa que é a prova viva de que o esforço nos faz crescer profissionalmente. Ao meu amigo, Gustavo Albuquerque, por não temer disciplina alguma e me ensinar a fazer o mesmo. À minha amiga, Sarah Ninsi, por sempre agregar do mínimo ao maior detalhe com suas opiniões. E principalmente à Eduarda Menezes, por enriquecer e amadurecer os valores que sigo.

Agradeço à todos os mentores e professores que tive durante todas as fases da vida até chegar aqui.

RESUMO

A gestão inadequada de resíduos hospitalares é um risco para ambientes urbanos, agravado pela pandemia de COVID-19. Este trabalho buscou desenvolver um aplicativo que utilize aprendizado profundo para identificar descartes inadequados desses resíduos. Uma base de dados foi cuidadosamente preparada, envolvendo a coleta de uma ampla variedade de imagens provenientes de diversas fontes distintas. Foram construídas máscaras de segmentação para cada imagem, preparando os dados e aplicando técnicas de aumento para criar um segundo conjunto de dados. Doze modelos foram treinados com técnicas específicas para melhorar o desempenho, usando a arquitetura YOLOv8 para segmentação de imagem. A escolha final foi o modelo YOLOv8x, que apresentou os melhores resultados em métricas como uma taxa de revocação de 0,62, um mAP50 de 0,67 e um mAP50-95 de 0,43, especialmente quando treinado com um conjunto de dados aumentado. O modelo selecionado foi implementado em um servidor local e conectado ao aplicativo por meio de requisições web assíncronas. Com o aplicativo, desenvolvido em Swift, os usuários podem enviar imagens da câmera para que o servidor analise as imagens, desenhe máscaras onde encontra descartes inadequados e envia a imagem de volta ao aplicativo.

Palavras chave: resíduos hospitalares, ambientes urbanos, aplicativo, segmentação, *Deep Learning*, YOLO, *Swift*.

ABSTRACT

The inadequate management of hospital waste poses a significant risk to urban environments, which has been further exacerbated by the COVID-19 pandemic. This study has created an application utilizing deep learning techniques to identify instances of improper waste disposal. A meticulously curated database was compiled, encompassing a wide variety of images from various sources. Segmentation masks were meticulously generated for each image, and data preprocessing and augmentation techniques were applied to create a secondary dataset. Subsequently, we trained twelve distinct models using specialized techniques to enhance their performance, employing the YOLOv8 architecture for image segmentation. The final model of choice was YOLOv8x, which yielded the most promising results in key metrics such a recall rate of 0.62, an mAP50 of 0.67, and an mAP50-95 of 0.43, particularly when trained on the augmented dataset. This selected model was then deployed on a local server and seamlessly integrated with the application through asynchronous web requests. The application itself was developed using Swift, providing users with the capability to submit images captured by their cameras for server-based analysis. The server draws masks to identify instances of improper waste disposal and returns the modified image to the user through the application.

Keywords: hospital waste, urban environments, application, segmentation, Deep Learning, YOLO, Swift.

Lista de Figuras

Lista de Tabelas

Lista de Abreviatura

ANVISA Agência Nacional de Vigilância Sanitária

AP *Average Precision*

APP *Application*

CLIP *Contrastive Language-Image Pretraining*

CNN *Convolutional Neural Networks*

CPU *Central Processing Units*

FN *False Negative*

FP *False Positive*

GPU *Graphic Processing Units*

HTTP *Hypertext Transfer Protocol*

HTTPS *Hypertext Transfer Protocol Secure*

IHM Interface Humano-Máquina

iOS *iPhone Operational System*

IOT *Internet of Things*

IoU *Intersection over union*

mAP *Mean Average Precisio*

NAT *Network Address Translation*

NMS *Non-maximum Supression*

OPAS Organização Pan-Americana da Saúde

OpenCV *Open Source Computer Vision Library*

RNA Redes Neurais Artificiais

TN *True Negative*

TP *True Positive*

URL *Uniform Resource Locator*

VS Code *Visual Studio Code*

YOLO *You Only Live Once*

SUMÁRIO

1 INTRODUÇÃO

O descarte incorreto de lixos hospitalares é um grande risco para a vida urbana devido à presença de contaminações dos microrganismos infecciosos e substância radiotivas. Segundo a Organização Pan-Americana da Saúde, 30% das instalações de saúde não estão equipadas para lidar com as cargas de resíduos existentes. Isso se agrava nos países menos desenvolvidos, que compõe 60% desse número (OPAS, 2022). Esse cenário se agravou com o COVID-19, pois as autoridades consideram que toneladas de descartes extras tenham sido gerados sem compensação no gerenciamento de resíduos. Especificamente no Brasil, 90% dos materiais recicláveis vêm da interveção humana através dos catadores, visto que os espaços aos quais são enviados descartes não possuem separação ou tratamento adequado. (MNCR, 2017)

Com o cotidiano cada vez mais simplificado pelo uso dos *smartphones* por mais da metade da população mundial (ANALYTICS, 2021), a demanda por soluções e produtos que utilizam as funções trazidas pelos equipamentos também aumentou. As câmeras desses aparelhos foram ganhando mais espaço em atividades rotineiras, como pagamentos a partir de código de barras ou *QR Codes*, reconhecimento facial e videochamadas. Logo, soluções antes que precisavam de deslocamentos ou máquinas especializadas foram resumidas ao simples uso de um celular inteligente. Portanto, a integração de uma tecnologia de fácil acesso que auxilie na identificação dos descartes provenientes de ambientes hospitalares ao processo de separação de resíduos reutilizáveis proporcionará uma prevenção ao contato entre o usuário e itens contaminados, além de poder viabilizar uma diminuição de riscos biológicos ou uma contaminação urbana.

Além disso, os hardwares dos computares vêm apresentando evoluções nos últimos anos e, com isso, o *Deep Learning* ("Aprendizado Profundo") tornou-se solução para muitos tipos de problemas, como detecção e reconhecimento facial, diagnóstico médico, aplicações na agricultura, detecções de objetos, entre outros (VO et al., 2019). Logo, a utilização de algoritmos de Visão Computacional e arquiteturas de *Deep Learning* estão cada vez mais frequentes em abordagens como classificação, detecção e segmentação de lixos hospitalares (MYTHILI; ANBARASI, 2020).

Contudo, esse trabalho possui como objetivo final projetar, desenvolver e avaliar uma solução inovadora de visão computacional, fundamentada na segmentação de objetos, com o objetivo de capacitar os usuários de dispositivos celulares a identificar resíduos hospitalares de forma eficiente e precisa, mesmo em meio a diversos outros materiais. Essa tecnologia promissora visa aprimorar a segurança sanitária e contribuir significativamente para ambientes mais saudáveis.

1.1 PROBLEMA DE PESQUISA

O descarte vindo de ambientes hospitalares são portadores de microrganismo infecciosos e substâncias radioativas que podem prejudicar gravemente a vida de uma pessoa e, em alguns casos, também podem ser a causa da propagação de doenças infecciosas para toda a vida urbana.

Lixos hospitalares contêm rejeitos químicos, inflamáveis e biologicamente perigosos, por isso, seu despejo deve seguir a norma 307 da Agência Nacional de Vigilância Sanitária (ANVISA), para que sejam separados em grupos específicos para evitar qualquer perigo. Além disso, considerando que em grande parte do Brasil, a segregação do lixo é feita por catadores em aterros, qualquer erro na separação desse lixo pode ocasionar a impactos ao ambiente e aos neles envolvidos (SOUSA et al., 2016).

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

Projetar, desenvolver e avaliar uma solução inovadora de visão computacional, fundamentada na segmentação de objetos, com o objetivo de capacitar os usuários de dispositivos celulares a identificar resíduos hospitalares de forma eficiente e precisa, mesmo em meio a diversos outros materiais. Essa tecnologia promissora visa aprimorar a segurança sanitária e contribuir significativamente para ambientes mais saudáveis.

1.2.2 OBJETIVOS ESPECÍFICOS

- Organizar um conjunto de dados de exemplo de descartes hopitalares para criação de modelos inteligentes;
- Projetar e desenvolver aplicação de fácil uso capaz de auxiliar o descarte de resíduos hospitalares;
- Avaliar comparativamente diferentes estratégias tecnológicas para o reconhecimento de objetos de interesse por imagens.

1.3 JUSTIFICATIVA

O desenvolvimento nesse tema poderá contribuir ao simplificar o acesso automatizado mais barato a informações de quais são os descartes hospitalares, podendo assim evitar riscos ambientais e da saúde pública.

De acordo com o trabalho de Pozzetti e Monteverde (2017), o Brasil possui em sua legislação mecanismos jurídicos fortes de proteção ao meio ambiente e a saúde públicas

quando se trata de descarte incorreto de lixo hospitalar, mas é um país carente no tocante a fiscalização. Com isso, a aplicação da pesquisa enriquece o tema devido estar diretamente ligado à identificação de descartes irregulares que podem ter acontecido devido à falta de fiscalização.

Também levando em consideração as diretrizes aconselhadas pela ONU (2022), o trabalho age na intercessão dos objetivos de “Cidade e Comunidade Sustentáveis”, “Consumo e Produção Responsáveis”e “Vida Terreste”. Isso acontece, pois, o mesmo busca agregar à qualidade de vida, pensando no que diz respeito no final do ciclo de consumo de um produto, causando um impacto para tentar diminuir contaminações, ou até mesmo, poluições.

2 REFERENCIAL TEÓRICO

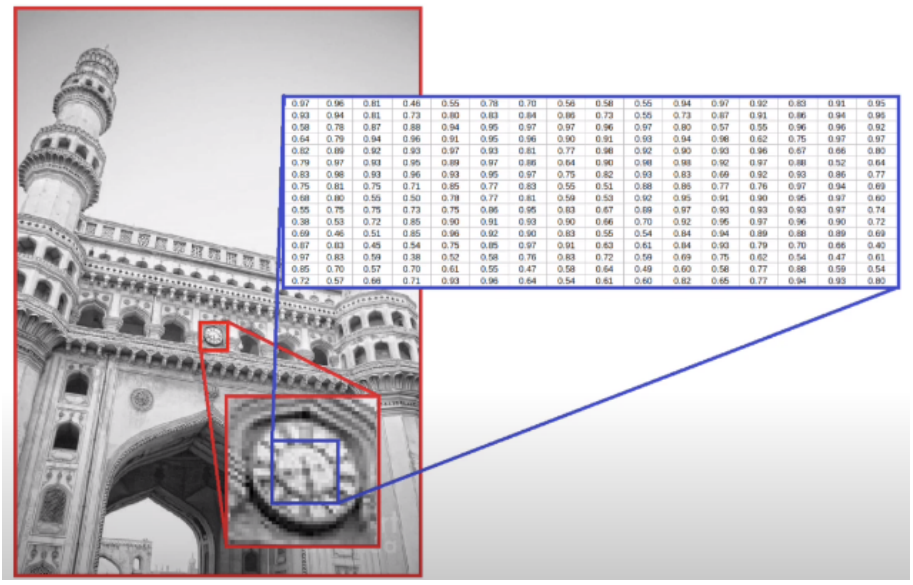
Neste capítulo serão apresentados os fundamentos teóricos e os conceitos que auxiliarão no desenvolvimento do trabalho proposto. Na Seção 2.1 é abordado os conceitos referentes às imagens e visão computacional. Na Seção 2.2 são apresentados todos os temas percorridos pelo projeto no que se refere ao aprendizado de máquina. Foi incluso logo após na sessão 2.3 um pouco sobre o que há de mais atual em base de dados sintéticas na comunidade ciêntifica. Já na Seção 2.4 é abordado todo o tema de construção do software, passando desde o servidor local até a linguagem utilizada para desenvolver o APP ("*Application*", ou Aplicativo). Também foram acrescentadas em cada tópico, noções sobre os softwares utilizados nesse estudo.

2.1 VISÃO COMPUTACIONAL

2.1.1 CONCEITOS

Imagens podem ser caracterizadas por uma função bidimensional $F(x, y)$, onde que x e y são as coordenadas espaciais e f é a amplitude em qualquer par de coordenadas, no qual é chamada de intensidade ou nível de cinza da imagem nesse ponto. Quando os valores de intensidade de f são quantidades finitas e discretas, podemos chamar de imagem digital. Portanto, quando é utilizado um computador para o processamento de imagens, refere-se ao Processamento Digital de Imagens (GONZALES; WOODS, 2010). A Figura 1 é um exemplo de como uma imagem na escala de cinza é interpretada pelo computador para o processamento.

Figura 1 – Representação de uma Imagem na escala de cinza



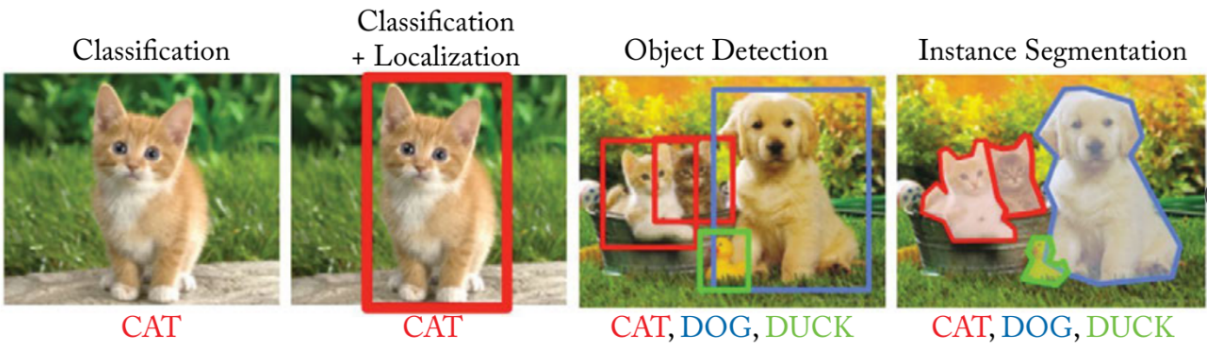
Fonte: Madras (2022)

Os seres humanos possuem um limitação de enxergar somente à banda visual do espectro eletromagnético, enquanto que os aparelhos de processamento de imagens podem cobrir quase todo o espectro. Isso ocorre variando ondas gama até ondas de rádio, podendo trabalhar com imagens feitas por fontes que os humanos não possuem costume de associar com imagens. Essas fontes incluem ultrassom, microscopia eletrônica e imagens geradas por computador (GONZALES; WOODS, 2010).

Ligado a isso, temos que a visão computacional é um ramo da inteligência artificial que possibilita equipamentos computadorizados interpretar informações vindas de imagens ou vídeos e proceder uma ação a partir disso. Esse processo é aplicado em técnicas focadas em automatizar tarefas que vão desde inspeção industrial até o entendimento de vídeo para guiar robôs autônomos para que possam interagir de maneira significativa e ao mesmo tempo segura (IBM, 2022).

Portanto, de acordo com Khan et al. (2018), visão computacional busca desenvolver métodos que sejam capazes de replicar uma das capacidades mais surpreendentes do sistema visual humano, ou seja, inferir características do mundo real 3D usando apenas a luz refletida para os olhos de vários objetos. Por exemplo, na Figura 2, é apresentado como geralmente é utilizado as aplicações de Visão Computacional retratando de aplicações como classificação de uma imagem e detectar ou segmentar um objeto especifico em uma imagem.

Figura 2 – Exemplo de Classificação, Deteção e Segmentação de Objetos



Fonte: Khan et al. (2018)

2.1.2 SEGMENTAÇÃO DE IMAGENS

A segmentação de imagens é a tarefa de encontrar grupos de pixels que representam um mesmo elemento. Em visão computacional, a segmentação de imagens é considerado um dos campos de estudo mais antigos e abordados, suas primeiras técnicas tendem a usar divisão ou fusão de regiões, que correspondem a algoritmos divisivos e aglomerativos na literatura de agrupamento. Os algoritmos mais recentes geralmente otimizam

alguns critérios como consistência intra-regional e comprimentos de limites entre regiões ou dissimilaridade (ALBANEZ; BATISTA; SILVA, 2016).

Quando imagens são convertidas a níveis de cinza, é possível realizar a segmentação de duas maneiras: por descontinuidade e por similaridade. Na primeira abordagem, o objetivo é particionar uma imagem bascando-se nas mudanças abruptas nos níveis de cinza. Como o interesse dentro dessa categoria é a detecção de pontos isolados, detecção de linhas e detecção de bordas numa imagem através de máscaras de convolução. E para segunda abordagem, a aplicação tem métodos baseados em limiarização ("*thresholding*"), crescimento de regiões ("*region growing*"), divisão e conquista ("*split and merge*") e aglomeração ("*clustering*") (ALBANEZ; BATISTA; SILVA, 2016).

Figura 3 – Exemplo de Segmentação de Instância em uma Imagem

Fonte: Khan et al. (2018)

2.1.3 OPENCV

OpenCV, abreviação de "*Open Source Computer Vision Library*", é uma biblioteca de código aberto desenvolvida para visão computacional e aprendizado de máquina. Compatível com sistemas operacionais como Windows, Linux, Android e macOS, seu objetivo é fornecer uma estrutura compartilhada para aplicações de visão computacional e impulsionar a incorporação de percepção de máquina em produtos comerciais. Com mais de 2.500 algoritmos otimizados, a biblioteca inclui tanto algoritmos clássicos quanto os mais recentes em visão computacional e aprendizado de máquina. O OpenCV oferece suporte a linguagens como C++, Python, Java e MATLAB, embora seja implementado em sua maioria em C++. (OPENCV, 2023)

Figura 4 – Logo do OpenCV

Fonte: OpenCV (2023)

2.2 DEEP LEARNING

2.2.1 CONCEITOS

Deep learning é um subcampo particular de metodologias de *Machine Learning* (ou Aprendizado de Máquina) utilizando Redes Neurais Artificiais (RNA) inspiradas na estrutura de neurônios localizados no cérebro humano (GULLI; PAL, 2017). De maneira específica, se trata de uma abordagem na qual o aprendizado se dá através de camadas sucessivas trazendo significado ao que deseja ser representado.

No seus usos mais recentes, o aprendizado profundo moderno geralmente envolve dezenas ou até centenas de camadas sucessivas de representações, com elas sendo aprendidas automaticamente com a exposição aos dados de treino. Além disso, o número de camadas que estão presentes no treinamento são chamadas de profundidade de um modelo e se esse número for até dois, considera-se a aplicação dessas camadas como de aprendizado superficial (CHOLLET, 2018).

É importante citar que atualmente estamos vivendo uma revolução em Deep Learning. Essa aceleração no desenvolvimento desse campo ocorre pela integração e utilização das Unidades de Processamento Gráfico (GPU), que agilizam o uso de larga escala do aprendizado profundo pois são ótimas para matriz paralela e álgebra vetorial (LIU, 2017).

2.2.2 REDES NEURAIS CONVOLUCIONAIS

As Redes Neurais Convolucionais (CNNs) operam de maneira muito semelhante com as redes neurais comuns. Sua principal diferença está na maneira como lida com suas camadas. Na aplicação desse método, cada unidade em um filtro de uma, ou mais, dimensões é concluído com a entrada dessa camada. Isso é essencial para casos em que é necessário aprender padrões de mídia de entrada de alta dimensão, como por exemplo, imagens e vídeos (KHAN et al., 2018).

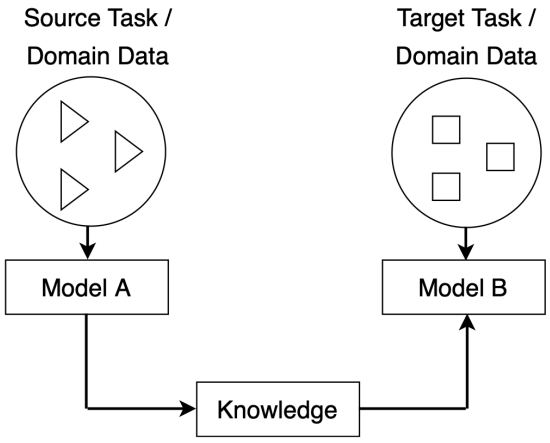
A rede convolucional aproveita o fato de usar arquivos de imagem em sua maioria

e restringe sensivelmente a arquitetura da rede profunda para que sejam reduzidos drasticamente o número de parâmetros no modelo analisado. Além disso, os neurônios em uma camada convolucional estão conectados apenas a uma pequena região local da camada anterior, de modo que é evitado o desperdício de neurônios totalmente conectados. A função de uma camada convolucional pode ser expressa de forma simples: ela processa um volume tridimensional de informação para produzir um novo volume tridimensional de informação (BUDUMA; LACASCIO, 2017).

2.2.3 TRANSFER LEARNING

O paradigma tradicional de aprendizado supervisionado se torna limitado quando não dispomos de dados rotulados suficientes para treinar um modelo confiável na tarefa ou domínio desejado. A transferência de aprendizado ("*Transfer Learning*") nos permite lidar com essa situação ao aproveitarmos os dados de uma tarefa ou domínio relacionado, conhecido como tarefa e domínio de origem. Transferimos o conhecimento adquirido ao resolver a tarefa de origem no domínio de origem para a tarefa e o domínio de destino, conforme ilustrado na figura 5. Especificamente para modelos baseados em redes neurais, que são usados em todo este trabalho, esse conhecimento está relacionado à representação aprendida. (JIN et al., 2020).

Figura 5 – Configuração do Transfer Learning



Fonte: Jin et al. (2020)

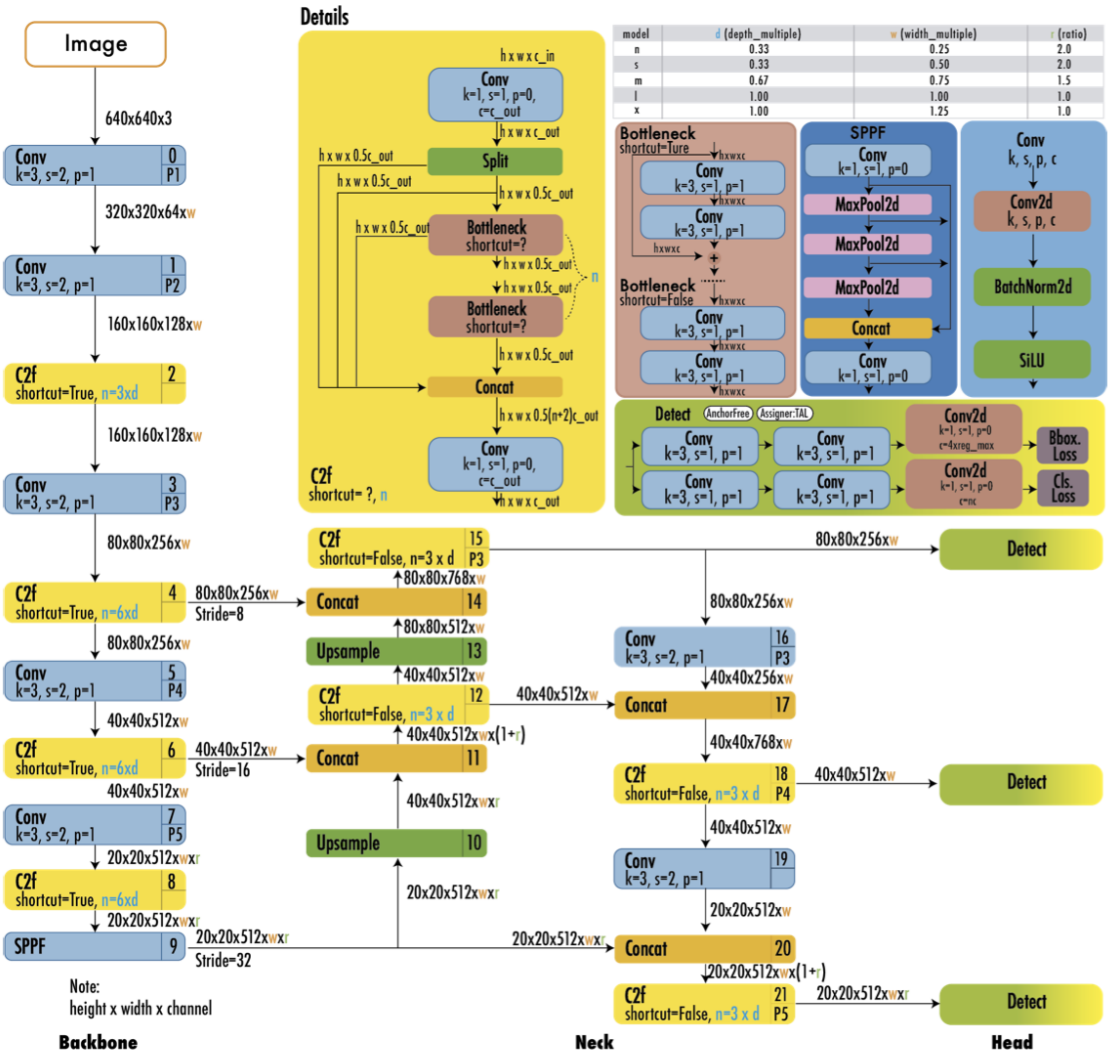
2.2.4 YOLOv8

O que diferencia as arquiteturas da família YOLO de outras versões de CNNs é que os algoritmos YOLO só necessita passar pela rede neural uma vez. Sua primeira versão foi lançada em 2016, moldando o básico da família. Ele envolve a divisão de

uma imagem em uma grade de células, gerando caixas delimitadoras com pontuações de confiança e probabilidade ao trabalhar com o *framework* Darknet. Em seguida, as Regiões de Interesse (Rols) são então encaminhadas para redes neurais, onde ocorre a extração de características, seguida pelo processo de regressão e previsão. (YU, 2020).

Evoluindo o YOLO desde sua quinta versão, a Ultralytics lançou o YOLOv8, por Jocher, Chaurasia e Qiu (2023), em janeiro de 2023. Nessa geração, o algoritmo não usa mais âncoras, prevê menos caixas e possui um processo de supressão que não é máxima, também chamada de Non-maximum Supression (NMS). Durante o treino, é utilizada a técnica de aumento chamada de "mosaic augmentation", mas que é desativada a partir das dez últimas épocas pois pode ser prejudicial ao resultado. É possível a estrutra da sua oitava versão na figura 6. (TERVEN; CORDOVA-ESPARZA, 2023).

Figura 6 – Estrutura da arquitetura do YOLOv8



Fonte: Terven e Cordova-Esparza (2023)

Pela figura 7 podem ser visto a comparações entre as cinco escalas do modelo para segmentação, sendo elas: YOLOv8n (nano), YOLOv8s (pequeno), YOLOv8m (médio), YOLOv8l (grande) e YOLOv8x (extra-grande).

Figura 7 – Comparação das versões do YOLOv8 baseada no dataset COCO val2017

Model	size (pixels)	mAP^{box} 50-95	mAP^{mask} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s-seg	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m-seg	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l-seg	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x-seg	640	53.4	43.4	712.1	4.02	71.8	344.1

Fonte: Ultralytics (2023)

2.2.5 MÉTRICAS DE VALIDAÇÃO DOS MODELOS

Para essa seção, serão exploradas métricas amplamente empregadas em cenários de segmentação. Elas estarão sendo empregadas para avaliar a eficácia dos modelos no contexto do projeto proposto. As métricas em questão são derivadas da matriz de confusão e incluem as seguintes: precisão (P), revocação (R), mAP50 e mAP50-95.

- **MATRIZ DE CONFUSÃO**

Se trata de uma técnica que utiliza uma matriz de quadrados cruzando as previsões feitas pelo modelo com os valores reais, possibilitando a análise das classificações corretas e incorretas obtidas pelo modelo. A avaliação de algoritmos em problemas de classificação recorre à matriz de confusão frequentemente por ser um método especialmente eficaz (MARSLAND, 2015).

Na tabela 1, é possível checar uma representação de uma matriz de confusão e logo abaixo o significado dos itens presentes.

Tabela 1 – Representação de uma Matriz de Confusão

	Previsto	
Real	Positivo	Negativo
Positivo	TP	FN
Negativo	FP	TN

a) *"True Positive"* (TP): Verdadeiro Positivo, indica a frequência em que o modelo previu corretamente uma condição como positiva, quando era de fato positiva;

b) *"True Negative"* (TN): Verdadeiro Negativo, refere-se ao número de vezes que o modelo acertadamente previu uma condição como negativa, quando era de fato negativa;

c) *"False Positive"* (FP): Falso Positivo, representa a ocorrência em que o modelo erroneamente previu uma condição como positiva, quando na realidade era negativa;

d) *"False Negative"* (FN): Falso Negativo, denota a situação em que o modelo equivocadamente previu uma condição como negativa, enquanto ela era positiva.

- **PRECISÃO**

A métrica da precisão é empregada para avaliar a proporção de verdadeiros positivos em relação ao total de instâncias que o modelo previu como positivas, que podem ser Verdadeiros Positivos e Falsos Positivos, segundo Marsland (2015). Na perspectiva matemática, sua definição é expressa:

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{1}$$

- **REVOCAÇÃO**

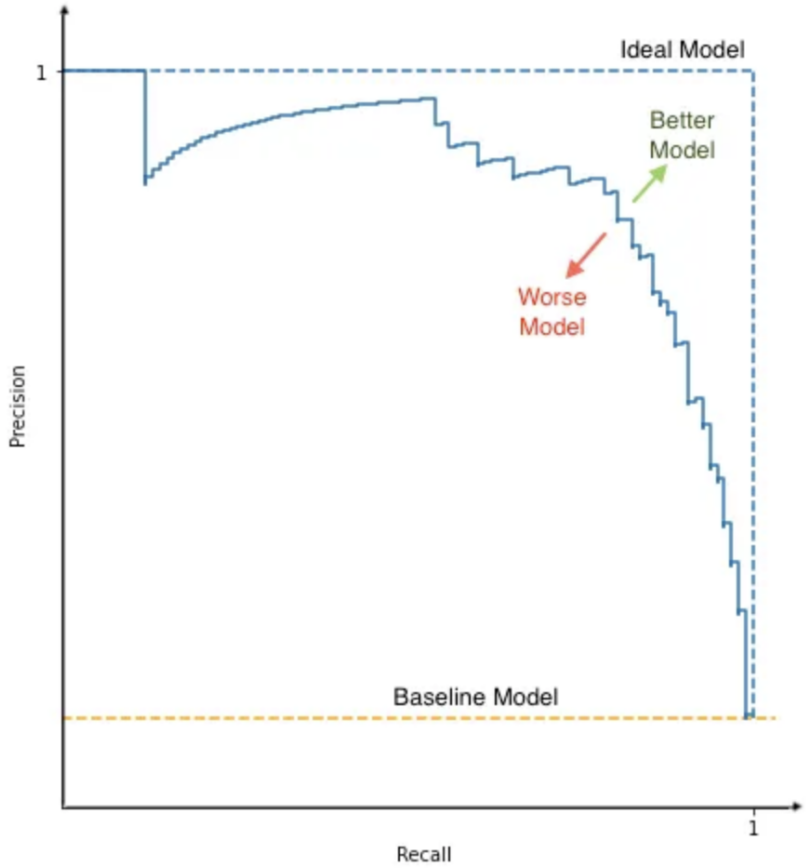
A métrica de "*recall*"(ou revocação) é empregada para avaliar a proporção de casos positivos identificados corretamente. Isso envolve a relação entre os verdadeiros positivos e a soma dos verdadeiros positivos com os falsos negativos (MARSLAND, 2015).

$$\text{Revocação} = \frac{TP}{TP + FN} \tag{2}$$

- **CURVA PRECISÃO-REVOCAÇÃO**

A curva precisão-revocação apresenta o equilíbrio entre a precisão e a revocação para diferentes limiares. Ela é frequentemente empregada em situações onde as classes estão desproporcionalmente distribuídas. Por exemplo, se uma observação é prevista como pertencente à classe positiva com uma probabilidade de 0,5, é rotulada como positiva. Contudo, é possível escolher qualquer limiar de probabilidade entre 0 e 1. A curva precisão-revocação auxilia na visualização de como o limiar afeta o desempenho do classificador. (SAXENA, 2022).

Figura 8 – Gráfico da Curva Previsão-Revocação



Fonte: Saxena (2022)

- **mAP50**

A Precisão Média do Valor Médio (mAP, do inglês "Mean Average Precision") é empregada para fazer a avaliação do desempenho de modelos de visão computacional. O mAP equivale à média da métrica de Precisão Média (*Average Precision*) em todas as classes de um modelo além de ser utilizado para comparar tanto diferentes modelos executando a mesma tarefa quanto diferentes versões do mesmo modelo.

Essa métrica é calculada em uma escala entre 0 e 1 e para compreender com mais profundidade a Precisão Média do Valor Médio. Seu cálculo envolve a escolha de um limiar de Interseção sobre União (IoU), que pode ser um valor único ou um intervalo. Um limiar de IoU mais alto estabelece critérios mais rigorosos, levando a uma diminuição do valor de mAP. O cálculo do AP começa traçando uma curva de Precisão-Revocação (PR) para um grupo de detecção, reduzindo gradualmente o limiar de confiança.

O AP é a área sob essa curva, ponderada pelo aumento de *recall* em relação ao limiar anterior. (SOLAWETZ, 2020). Logo o mAP é calculado encontrando a Precisão Média (AP) para cada classe e, em seguida, fazendo a média sobre um número de classes.

A equação 3 representa sua fórmula matemática:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} (AP_i) \tag{3}$$

Em seguida, as detecções são agrupadas com base na classe detectada, e a Precisão Média do Valor Médio (mAP) é calculada para cada grupo. O mAP resultante para um limiar de IoU específico é a média desses valores de AP. Na prática, o mAP é aplicado para comparar o desempenho de diferentes modelos, como mostrado em um experimento de detecção de células. (SOLAWETZ, 2020).

- **mAP50-95**

Os resultados mais recentes de pesquisas frequentemente apresentam métricas para o conjunto de dados COCO. Na métrica mAP do COCO, o AP@[.5:.95] representa a média do AP para valores de IoU entre 0,5 e 0,95, com um incremento de 0,05. Na competição COCO, o AP é a média de 10 níveis de IoU em 80 categorias (AP@[.50:.05:.95]: começa em 0,5 e chega a 0,95 com incremento de 0,05). (HUI, 2018)

Essa explicação pode ser reescrita em forma da seguinte equação:

$$mAP_{COCO} = \frac{mAP_{0.50} + mAP_{0.55} + \ldots + mAP_{0.95}}{10} \tag{4}$$

2.2.6 ROBOFLOW

O Roboflow é uma plataforma desenvolvida para simplificar e otimizar trabalhos de visão computacional que envolvam imagens. Ele possui um conjunto de ferramentas envolvendo todos os ciclos na preparação de um conjunto de imagens, capacitando o usuário a fazer pré-processamento, organização e otimização desses dados em um conjunto. Além disso, apresenta a capacidade de geração de mais imagens por meio de técnicas de aumento de imagens disponíveis durante seu uso.(ROBOFLOW, 2023)

Também destaca-se por possuir uma ferrameta de anotação e geração de máscaras para segmentação de imagens, facilitando o treinamento supervisionado de modelos. A plataforma Roboflow é capaz de fazer o armazenamento eficiente de dados de imagens e datasets, podendo deixar disponível para o uso da comunidade. Por fim, está conectada com os principais algoritmos para aprendizado de máquina ligado a imagens, disponibilizando a exportação dos dados nos padrões necessários. (ROBOFLOW, 2023)

2.3 BASE DE DADOS SINTÉTICAS

2.3.1 CONCEITOS

A geração de dados sintéticos tem se mostrado uma abordagem promissora para superar desafios em projetos de ciência de dados. É possível observar restrições na utilização de dados reais causadas por preocupações com privacidade, o que limita a colaboração externa. Nesse contexto, o uso dessa técnica pode ser uma solução viável por reproduzirem tanto as características estatísticas quanto estruturais dos dados originais. (PATKI; WEDGE; VEERAMACHANENI, 2016)

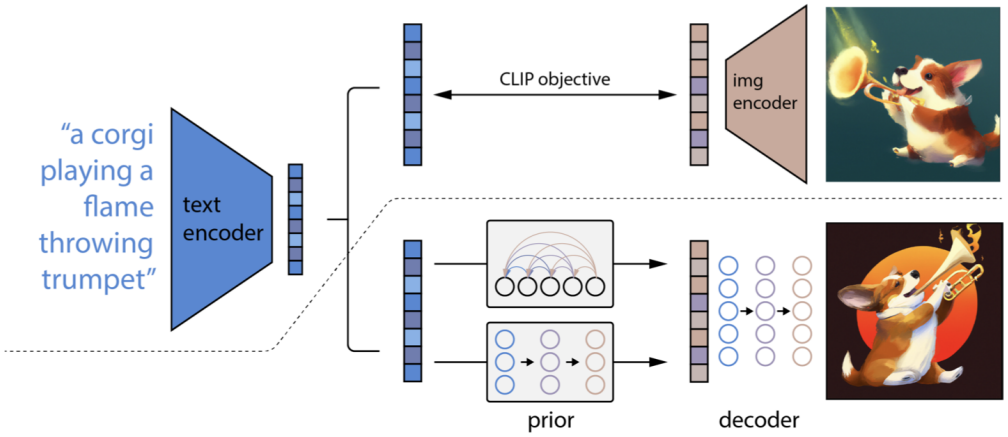
O estudo realizado por Patki, Wedge e Veeramachaneni (2016) também demonstra que a análise quantitativa mostra que a diferença de acurácia preditiva entre características desenvolvidas com dados sintéticos e aqueles criados com dados originais não é estatisticamente significativa em muitos casos. E ainda, a análise qualitativa ressalta a importância de esclarecimentos sobre as relações entre tabelas e o potencial para aprimorar a compreensão e a exploração dos dados sintéticos pelos cientistas de dados.

2.3.2 DALL-E

DALL-E é um software online que permite a geração de imagens sintéticas. Para construir essas imagens, ele utiliza um modelo de duas etapas: primeiro, é gerado uma representação de imagem CLIP (*Contrastive Language-Image Pretraining* ou Pré-treinamento Contrastivo de Linguagem e Imagem em tradução livre) a partir de uma legenda de texto usando um "*prior*"; em seguida, utiliza um "*decoder*"para gerar uma imagem condicionada a essa representação de imagem. Nesse caso, a abordagem de gerar explicitamente representações de imagem melhora a diversidade, mantendo apenas uma pequena perda referindo-se a termos de realismo fotográfico e similaridade com a legenda. (RAMESH et al., 2022).

Além disso, os decoders condicionados a representar imagens podem gerar algumas variações de uma imagem que preserva tanto a semântica quanto o estilo, e ao mesmo tempo em que variam os detalhes não essenciais ausentes na representação da imagem. O espaço de incorporação conjunto do CLIP também permite manipulações de imagem guiadas por linguagem de forma inovadora. O DALL-E utiliza modelos de difusão para o decoder e experimenta tanto com modelos autoregressivos quanto com modelos de difusão para o prior, concluindo que os últimos são mais eficientes computacionalmente e geram amostras de maior qualidade. (RAMESH et al., 2022).

Figura 9 – Funcionamento Estrutural do DALL-E



Fonte: Ramesh et al. (2022)

2.4 DESENVOLVIMENTO DE SOFTWARE

2.4.1 SERVIDOR LOCAL

Um servidor local é usar um dispositivo físico, geralmente um computador, disponível nas instalações para atender às solicitações de clientes na rede local. Esse servidor não cobra uma equipe para sua manutenção ou atualização. Suas vantagens incluem maior segurança de dados, controle direto sobre o acesso, rapidez na resposta a falhas devido à proximidade e conexões rápidas com bancos de dados internos. (SAFETEC, 2022).

Figura 10 – Comunicação de um servidor local

Fonte: Guilherme (2019)

Além disso, o servidor local é independente da conexão com a internet, o que evita interrupções na eficiência e produtividade em caso de problemas de conectividade. Ele também permite a realização de backups internos de forma mais ágil e segura. Portanto,

o servidor local apresenta diversas vantagens, desde a segurança dos dados até a independência em relação à internet. (SAFETEC, 2022).

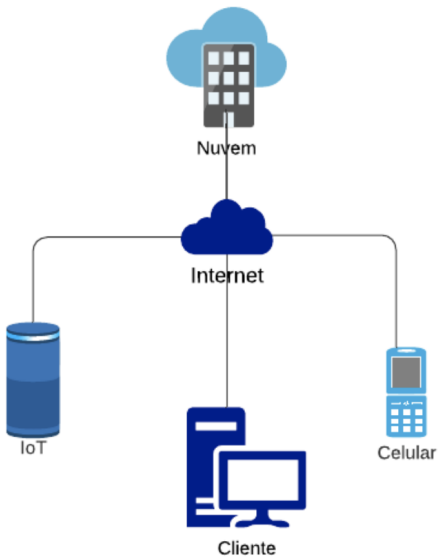
2.4.2 SERVIÇOS EM CLOUD

Os serviços em nuvem são infraestruturas, plataformas ou softwares hospedados por provedores terceirizados e disponibilizados aos usuários pela Internet. Esse tipo de fornecimento foi criado para facilitar o fluxo de informações com dispositivos diversos, tornando assim possível a democratização do acesso de determinado serviço para um simples equipamento computadorizado (REDHAT, 2022).

Quando ocorre a opção de utilizar uma infraestrutura em nuvem, é feita uma separação de recursos. Podem ser separados em: processamento das unidades de processamento central (CPUs), Memória ativa de chips de memória de acesso aleatório (RAM), Processamento gráfico das unidades de processamento gráfico (GPU), entre outras. Essa abstração geralmente é realizada por meio de virtualização e máquinas virtuais. Esse tipo de serviço em nuvem levou ao surgimento do armazenamento em nuvem, que armazena big data advindas da Internet das Coisas (IoT) (REDHAT, 2022).

Temos uma categorização do uso soluções em infraestrutura, plataformas, software e função como serviços, em que cada um tem sua tarefa específica em uso. Logo, toda a forma que o usuário interage com uma solução em que não exige-se downloads adicionais podem ser considerados serviços de computação (REDHAT, 2022).

Figura 11 – Funcionamento do Modelo em Nuvem



Fonte: Couto (2022)

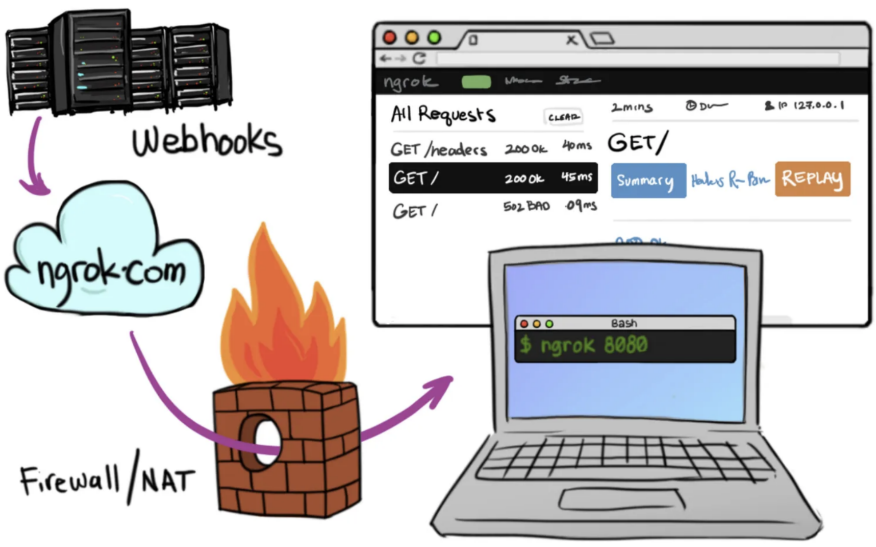
2.4.3 VS Code

O Visual Studio Code, conhecido pela sua abreviação “VS Code”, é uma ferramenta de desenvolvimento altamente popular e amplamente adotada pela comunidade de programadores. Desenvolvido pela Microsoft, o VS Code é um ambiente de desenvolvimento integrado (IDE) de código aberto, o que significa que é acessível gratuitamente para uma ampla gama de profissionais em todo o mundo. Uma das principais razões para sua popularidade é sua extensibilidade e flexibilidade incomparáveis, além de um rico ecossistema de extensões, o VS Code pode ser personalizado para atender às necessidades específicas de desenvolvimento de praticamente qualquer linguagem de programação. Uma das principais vantagens dessa IDE é sua interface de usuário que permite o ajuste e adaptação ao usuário. Além disso, é oferecido recursos avançados de edição de código, como conclusão de código inteligente, depuração integrada e um sistema de controle de versão Git integrado. (MICROSOFT, 2023)

2.4.4 NGROK

O Ngrok é uma ferramenta em linha de comando multiplataforma que auxilia na criação de um túnel seguro a partir de um ponto de extremidade público, como a internet, para uma rede local em execução. Ele possibilita a criação instantânea de um URL, *Uniform Resource Locator*, público HTTPS para um site em execução localmente em nossa máquina. Além disso, é possível configurar credenciais de autenticação HTTP para proteger o acesso ao túnel e aos dados compartilhados dentro dele. (SIBY; GS, 2020).

Figura 12 – Funcionamento do Modelo em Nuvem



Fonte: Paixão (2021)

Para a segurança, também utiliza NAT(*Network Address Translation*) e *Firewalls* (PAIXãO, 2021). O Ngrok funciona em qualquer lugar sem necessidade de alterações, mesmo quando um dispositivo muda de redes. É possível utilizar a interface de inspeção *web* do Ngrok para compreender o tráfego de solicitações e respostas HTTP por meio do túnel. (SIBY; GS, 2020).

2.4.5 XCODE

O Xcode 15 possibilita o desenvolvimento, teste e distribuição de aplicativos para todas as plataformas da Apple. ste ambiente de desenvolvimento oferece uma ampla gama de recursos e funcionalidades que permitem aos desenvolvedores conceber, testar e distribuir aplicativos de alta qualidade para dispositivos como iPhone, iPad, Mac e Apple Watch. Agiliza o processo de codificação e design dos seus aplicativos nativos iOS com aprimoramento na conclusão de código, visualizações interativas e animações em tempo real. Além disso, utiliza o sistema de preparação do Git para elaborar o seu próximo *commit* sem sair do ambiente de código. Também permite a exploração e analise dos resultados nos testes com relatórios redesenhados, incluindo gravações em vídeo. (APPLE, 2023)

O Xcode permite a integração com a plataforma Xcode *Cloud*, tornando o processo de implantação no TestFlight e na *App Store* mais simples e eficiente. Além disso, as atualizações frequentes e a comunidade de desenvolvedores ativa garantem que o Xcode esteja sempre atualizado e oferecendo suporte às tecnologias mais recentes da Apple. Portanto, o Xcode é uma peça fundamental no desenvolvimento de aplicativos para dispositivos Apple, facilitando a criação de experiências incríveis para os usuários. (APPLE, 2023)

2.4.6 SWIFT

De acordo com Reboucas (2016), após dezoito anos utilizando a linguagem Objective-C para o desenvolvimento nativo, a Apple anunciou no ano de 2014 uma nova linguagem de programação chamada Swift. Ela veio com o objetivo de ser mais confortável em sintaxe e para possibilitar o desenvolvimento para todas as plataformas da empresa: iOS, OS X, watchOS e tOS.

Swift se trata de uma linguagem multi-paradigma, de códigos compilados e estaticamente verificada. Foi desenvolvida para trabalhar em conjunto com o *framework* principal da Apple e para ser totalmente compatível com o antigo Objective-C, mas ao mesmo tempo sendo menos propensa a erros (por recurso como *Optionals*) e mais conciso. Além disso, a linguagem mais nova da Apple compartilha recursos vindos do Objective-C, aliados a conceitos mais modernos como programação com atributos genéricos, inferências de tipos e muitos outros (REBOUCAS, 2016).

A Apple também se refere ao Swift como uma “linguagem orientada a protocolos”, pois acrescentou o conceito de extensibilidade de protocolo, no qual tipos, structs e classes podem ser estendidos e modificados (REBOUCAS, 2016).

Figura 13 – Logo da linguagem Swift

Fonte: Swift.org (2022)

3 TRABALHOS RELACIONADOS

Nos últimos anos, a linha de pesquisa que explora a aplicação de *deep learning* em imagens de resíduos sólidos tem experimentado um notável enriquecimento por meio de diversas contribuições.

Um exemplo notável é o estudo de Geetha et al. (2022), que demonstra o desenvolvimento de um aplicativo móvel que permite a identificação e notificação de amontoados de lixo, a ativação de redes neurais para classificação e estimativa de volume, além da alocação eficiente de limpadores por meio de algoritmos genéticos. Essa abordagem amplamente explorada aprimora significativamente a gestão de resíduos, enquanto o potencial total do aprendizado profundo é investigado reforçadamente para enfrentar um problema em constante crescimento em múltiplas regiões. A pesquisa também abrangeu uma análise comparativa de diferentes algoritmos de classificação de imagem, com destaque para o uso da rede YoloV5, a qual demonstrou uma notável precisão de 95,6% no projeto.

O artigo de Mythili e Anbarasi (2020) realiza a segmentação de imagens contendo resíduos hospitalares já existentes. Nesse estudo, eles optaram pelo uso da DNN-TC e compararam com a EngSegNet-DNN-TCC em um conjunto de cem imagens. Surpreendentemente, a última abordagem apresentou uma acurácia superior e uma taxa de erro menor em comparação com a primeira. Além disso, é válido mencionar o trabalho de Minaee et al. (2020), que aprofundou significativamente o uso do deep learning em imagens, não apenas abordando as subdivisões do estado da arte, mas também detalhando os algoritmos empregados e suas respectivas métricas de validação, ampliando o conhecimento nesse campo.

Além dessas pesquisas específicas, uma linha de pesquisa relevante e mais moderna explorada no estudo de Patki, Wedge e Veeramachaneni (2016) concentrou-se em gerar dados sintéticos para fortalecer projetos de ciência de dados. O Synthetic Data Vault (SDV) constrói modelos generativos de bancos de dados relacionais, possibilitando a criação de dados sintéticos realistas que substituem dados reais em projetos. Isso foi demonstrado por um experimento no qual cientistas de dados obtiveram resultados comparáveis com dados sintéticos, usando modelagem estatística dos dados originais. Essa abordagem resolve preocupações de privacidade e acesso a dados. As pesquisas em deep learning para imagens de resíduos também progrediram, enquanto a criação de dados sintéticos, como no SDV, destaca-se como uma promissora estratégia para impulsionar a ciência de dados, podendo transformar a abordagem a dados e desafios em pesquisa e aplicações.

4 METODOLOGIA

Esse estudo se propôs a realizar uma pesquisa de cunho aplicado. Para atingir os objetivos delineados, uma abordagem que integra aspectos qualitativos e quantitativos foi adotada. A seguir são descritos os procedimentos metodológicos empregados juntamente com o material utilizado.

O projeto foi dividido em seis etapas:

a) Criação e preparação da base de dados, fase relacionada com a coleta e as anotações das imagens;

b) Treinamento dos modelos YOLO para a segmentação de objetos;

c) Avaliar e comparar os resultados dos modelos para selecionar o que possui melhor desempenho;

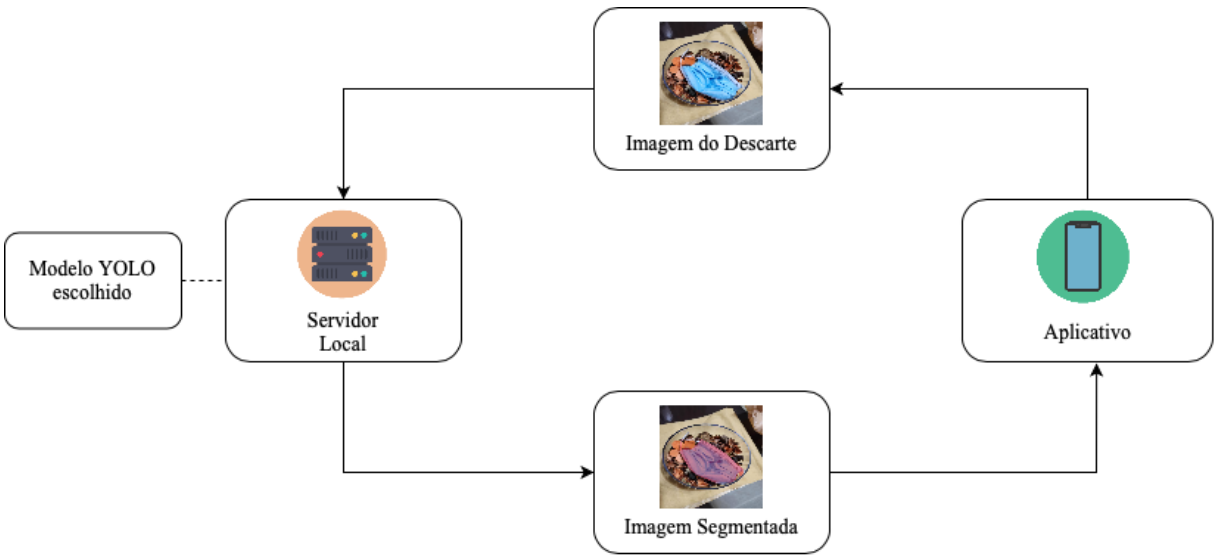
d) Desenvolvimento de um servidor web local e utilizar o NGROK para criar um túnel e, com isso, expor o servidor para uma comunicação externa;

e) *Deploy* do melhor modelo no servidor web;

f) Desenvolvimento do aplicativo utilizando a linguagem *Swift* e integrá-lo ao servidor.

O diagrama do funcionamento da aplicação final é mostrada na Figura 14, onde que o usuário captura uma imagem em um aterro sanitário com o *Smartphone* e a mesma é enviada para o servidor *web*, no qual o modelo de selecionado está em produção. A imagem resultante contém a segmentação do lixo hospitalar detectado pelo modelo e é retornado para o dispositivo do usuário.

Figura 14 – Diagrama de Comunicação Aplicativo - Servidor



Fonte: Própria

4.1 CRIAÇÃO E PREPARAÇÃO DA BASE DE DADOS

Para obter um resultado robusto e mais preciso, é fundamental realizar uma coleta e preparação do conjunto de dados de maneira adequada com a realidade do problema estipulado. A qualidade da representação, assim como os rótulos ou *labels*, tem impacto direto de realizar uma segmentação correspondente. Portanto, foram realizadas pesquisas de *datasets* já existentes para a problemática do trabalho, porém, até o presente momento, há uma escassez de base de dados que obtém esta abordagem de lixos hospitalares em aterros sanitários ou em ambientes abertos. Com isso, a criação de um *dataset* foi necessária.

O presente estudo focou nos principais itens observados durante as pesquisas de lixo hospitalar, tais como máscaras, seringas, ampolas e bolsas de sangue. Para a criação de uma base de dados estruturada para um modelo de segmentação de objetos como o YOLOv8, foi dividida essa etapa inicial em 5 subtarefas:

a) Coleta de imagens que possuem objetos hostilares em aterros sanitários ou em ambientes externos;

b) Coleta de imagens dos objetos selecionados para a segmentação;

c) Coleta de imagens de aterros sanitários;

d) Processo de sobrepor as imagens dos objetos nas imagens de aterros sanitários;

e) Segmentar e rotular os objetos presentes nas imagens.

4.1.1 COLETA DE IMAGENS

O primeiro passo na construção de um modelo para segmentação eficiente é a aquisição de imagens variadas com cenários que representam os desafios e condições que o modelo encontrará quando for utilizado. Esta seleção de imagens garante que o modelo abstraia padrões relevantes em diferentes contextos.

As imagens foram obtidas por diferentes fontes devido a escassez de fotografias sobre o tema. A fonte primária veio de pesquisas de termos relacionados aos objetos citados dentro do contexto de descarte hospitalar pelo *Google Imagens*, onde notícias e páginas sobre o conteúdo disponibilizavam as imagens ao público. Foram adquiridas 29 imagens extraídas diretamente do *Google* para compor a base de dados. Na Figura 15, apresenta uma amostra dessas imagens contendo máscaras, seringas, bolsas de sangue e ampolas.

Figura 15 – Imagens encontradas de lixos hospitalares

Fonte: Google Imagens

Outro recurso utilizado para adquirir foi por meio da edição de imagens, em que os objetos hospitalares definidos foram sobrepostos em imagens de aterros sanitários ou ambientes de descarte, como mostrado na Figura 16.

Figura 16 – Processo de sobrepor os objetos em imagens de aterros sanitários

Fonte: Própria

As imagens dos objetos e dos aterros sanitários foram extraídas também do *Google Imagens*. Foram utilizadas 16 imagens de diferentes ambientes de descarte para esse processo e diversos objetos hospitalares selecionados totalizando 198 imagens na base de

dados através dessa técnica. Uma amostra das imagens resultantes da edição é apresentada na Figura 17.

Figura 17 – Imagens com objetos hospitalares sobrepostos

Fonte: Própria

Por fim, uma outra fonte utilizada foi a geração de imagens por uma rede neural generativa do *OpenAI* chamada DALL-E[1] sendo responsável por gerar dados sintéticos, porém pouco explorada devido ao numero limitado de recursos adequados a necessidade. Teve como participação no conjunto de dados com 24 imagens e uma amostra das mesmas é apresentada na Figura 18.

[1]Site da ferramenta: https://openai.com/dall-e-2

Figura 18 – Imagens geradas pelo algoritmo DALL-E

Fonte: Própria

Um aspecto considerado durante a aquisição das imagens foi a variação de formatos, cores e tamanhos para cada material hospitalar definido. Essa pluralidade de imagens foi feita para evitar a especialização excessiva do modelo em um único padrão, fazendo com que assim, ele consiga atuar em cenários complexos e mais exigentes. No total, conseguiu-se 251 imagens para compor o conjunto de dados.

4.1.2 PROCESSO DE ANOTAÇÃO DAS IMAGENS

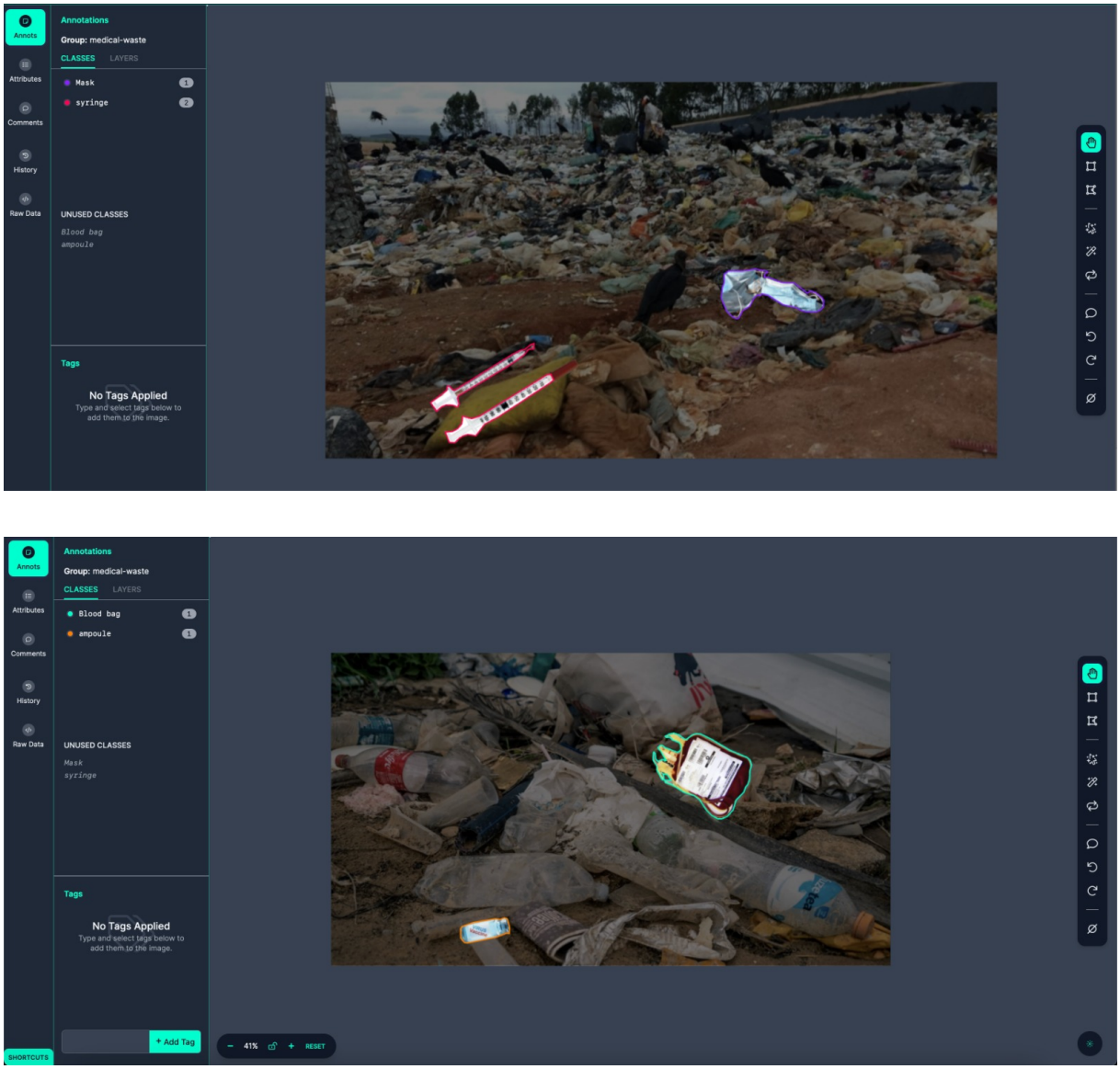
A próxima fase no processo de compilação do conjunto de dados envolve a elaboração de máscaras de segmentação para cada imagem previamente selecionada, de acordo com as classes predefinidas. Para realização desse procedimento, utilizou-se a plataforma de construção de *dataset* do *Roboflow* (Seção 2.2.6).

O *Roboflow* disponibiliza aos seus usuários diversas ferramentas para simplificar e agilizar a criação de *datasets*. Seu uso foi escolhido, pois, a plataforma já possui como um de seus padrões de organização a estrutura necessária de um *dataset* para modelos de YOLOV8 voltados para atividades de segmentação, além de suas capacidades de criar variações artificiais dos dados de treinamento para melhora de desempenho do modelo.

Foram construídas as máscaras nas regiões em que correspondiam aos descartes

hospitalares definidos de forma individual. Após a construção das máscaras, as mesmas eram rotuladas para identificação a que classe pertenciam. É válido se atentar que uma das limitações da plataforma caso fosse usado a ferramenta de geração de máscaras automáticas é o entendimento do objeto quando algo está abaixo ou sobreposto a ele, assim como da região completa ocupada pelo item quando o mesmo possui texturas irregulares. Para evitar esses casos, o delinear das áreas de segmentação foram construídas por interação humana. Na Figura 19, é visto como foi feita a segmentação dos objetos e classificando-os.

Figura 19 – Processo da segmentação e classificação dos objetos no *Roboflow*



Fonte: Própria

Uma amostra da base de dados com os objetos segmentados é apresentado na Figura 20.

Figura 20 – Objetos segmentados

Fonte: Própria

4.1.3 PREPARAÇÃO DO CONJUNTO DE DADOS

Obtendo as imagens com suas respectivas máscaras, a etapa seguinte foi adequar os dados brutos em um conjunto de dados para exportá-lo. Essa etapa envolveu a separação dos dados para uso no modelo e redimensionamento das imagens.

Primeiramente, foi realizada divisão apropriada do conjunto de dados em treinamento, validação e teste. O conjunto de treinamento é o que será utilizado para treinar o modelo, os dados agrupados em validação contribuem para otimizar hiperparâmetros e o teste é reservado para avaliar o desempenho final do modelo em dados não vistos. Para esse caso, a separação foi feita da seguinte forma:

a) 70% dos dados foram colocados no conjunto de treinamento totalizando 176 imagens;

b) 20% em validação, sendo 50 imagens;

c) 10% no grupo de teste com 25 imagens.

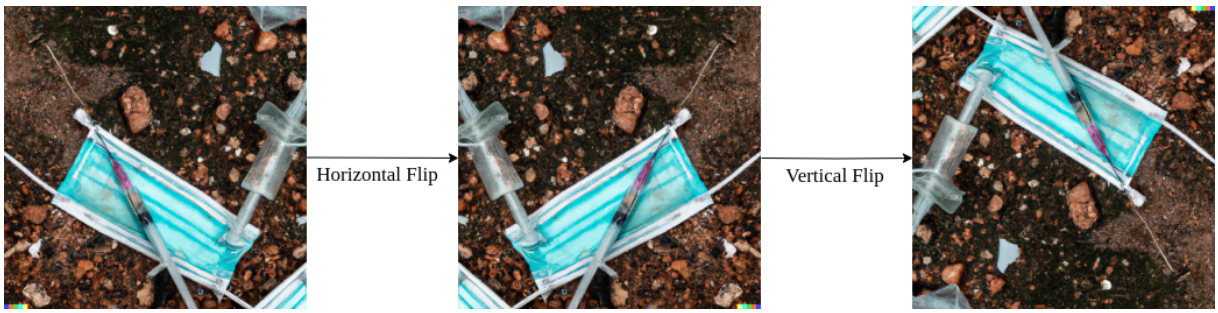
Com os dados separados, foi realizado o redimensionamento das imagens para o tamanho padronizado de 640 pixels de altura e largura. Isso é essencial pois trata-se de uma exigência comum das imagens no YOLOV8, além de contribuir para a economia de recursos computacionais durante o treino.

4.1.4 AUMENTO DO CONJUNTO DE DADOS DE TREINO

Além de realizar o treinamento do modelo com a base de dados com 251 imagens no total, também foi aplicado a técnica de aumento de dados ("*data augmentation*") para comparação e verificação dos resultados dos modelos para conseguir o melhor desempenho. Contudo, empregaram-se técnicas de *data augmentation* para ampliar o conjunto de treinamento utilizado no projeto, no qual envolve a aplicação de processamento de imagens ao conjunto, incorporando assim as novas imagens geradas. Neste estudo, foram utilizadas as modificações de espelhamento ("*Flip*") e rotação de 90º graus ("*90º Rotate*").

A primeira consiste em executar o espelhamento da imagem, sendo aplicada nesse contexto tanto na horizontal quanto na vertical, como é visto na Figura 21.

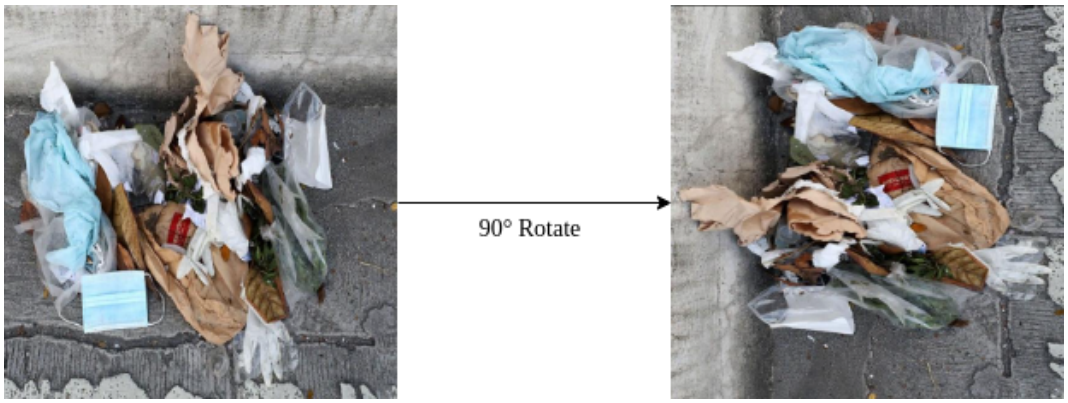
Figura 21 – Espelhamento horizontal e vertical da imagem



Fonte: Própria

A segunda se refere em realizar as rotações das imagens em 90° graus, aqui utilizada nos sentidos horário e anti-horário, um exemplo é apresentado na Figura 22.

Figura 22 – Rotação da imagem em 90º graus



Fonte: Própria

Por fim, registrou-se um notável crescimento, totalizando 545 imagens na base de dados, das quais 450 foram alocadas para treinamento, 60 para validação e 35 para o conjunto de teste. Essa diferença decorre de um processo de reequilíbrio na distribuição interna do conjunto de dados antes do processo de aumento de dados.

4.2 TREINAMENTO DOS MODELOS

Para a realização do estudo, foi escolhida a arquitetura YOLOv8. Essa escolha foi feita considerando o seu desempenho aplicado na solução de problemas em comparação com sua complexidade. Se trata da evolução da série YOLO de arquiteturas, que possuem destaque pela eficiência em detecção e segmentação de imagens. Os treinamentos foram realizados pelo *Google Colabory* utilizando o formato de ambiente de desenvolvimento, configurado com NVIDIA T4 Tensor Core GPUs, e a linguagem de programação *Python*. Para esse processo, foram testadas três variações do YOLOv8 aplicado: YOLOv8x, YOLOv8l e YOLOv8m visto que são os modelos com maiores métricas de mAP como mostra na Figura 7 da Seção 2.2.4.

Para o treinamento, utilizou-se o otimizador Adam, que é definido como um algoritmo que calcula não só as taxas de aprendizagem de maneira individual para cada parâmetro, como também do primeiro ao segundo momento dos gradientes. Ele é uma escolha mais eficiente para minimizar os ajustes de hiperparâmetros, que foram:

a) Tamanho do lote ("*Batch size*") = 8. O mesmo é um termo usado em aprendizado de máquina e refere-se ao número de exemplos de treinamento usados em uma iteração (ACADEMY, 2022). Neste hiperparâmetro, foi realizado uma tentativa com o valor 16, porém por conta da limitação de espaço de memória não foi possível finalizar o processo de treinamento limitando o valor do *batch size* sendo 8;

b) Épocas ("*Epochs*") = 100. É o número de vezes que os dados são passados pelo modelo. Porém, adicionamos o *early stopping* para o processo de treino encerrar quando o modelo não estiver mais aprendendo depois de 10 épocas;

c) Taxa de aprendizado ("*learning rate*") = 0,0001. É um hiperparâmetro que controla o quanto modificar o modelo em resposta ao erro estimado cada vez que os pesos do modelo são atualizados.

Outra técnica levada em consideração para alguns casos foi a transformação de multiclasses em uma única classe, fazendo assim o modelo ser focado em aprender todos os objetos de interesse como uma única classe e evitando os conflitos entre diferentes categorias. Portanto, todas as classes da base de dados, que são máscaras, bolsas de sangue, ampolas e seringas, foram tratadas como uma única classe sendo como lixos hospitalares.

Os modelos do YOLOv8 selecionados foram modelos pré-treinados no *dataset* COCO[2] e foi implementado a técnica de transferência de aprendizado ("*transfer learning*- Seção 2.2.3) para realizar o treinamento.

[2]Dataset: ⟨https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/datasets/coco.yaml⟩

Tabela 2 – Modelos de treinamento

Modelo	Aumento de Dados	Classe Única
YOLOv8m	Falso	Falso
	Falso	Verdadeiro
	Verdadeiro	Falso
	Verdadeiro	Verdadeiro
YOLOv8l	Falso	Falso
	Falso	Verdadeiro
	Verdadeiro	Falso
	Verdadeiro	Verdadeiro
YOLOv8x	Falso	Falso
	Falso	Verdadeiro
	Verdadeiro	Falso
	Verdadeiro	Verdadeiro

Fonte: Própria

Logo, resultou-se em 12 modelos treinados para análise dos resultados e selecionar o modelo com o melhor desempenho.

4.3 IMPLANTAÇÃO DO MODELO

Após a seleção do modelo, o próximo passo foi realizar o *deploy* do mesmo, no qual é a etapa de preparação do modelo para ser utilizado no dia a dia, ou seja, colocá-lo em produção. Logo, o mesmo foi exportado em um formato compátivel para ser utilizado em outros ambientes, além disso, um servidor local foi criado visando disponibilizar o uso do modelo em sistemas externos. No contexto dessa pesquisa, a máquina que hospedou o servidor foi um MacBook Pro de 2019, com um processador de 2.6 GHz 6-Core Intel Core i7 e para contexto gráfico possuiu a AMD Radeon Pro 5300M 4 GB e também a Intel UHD Graphics 630 1536 MB.

A implementação do servidor foi realizada empregando a linguagem *Python* e desenvolvido pelo VS Code, conforme ilustrado na figura 23 que demonstra seu início na porta 8082. O algoritmo foi projetado de maneira a facilitar sua interação com dispositivos móveis, permitindo a troca de dados simples e eficiente, resultando em uma economia

significativa de recursos.

Figura 23 – Servidor Local ao Iniciar

Fonte: Própria

A infraestrutura foi projetada para operar como um servidor *web* assíncrono, e isso foi viabilizado através da utilização da biblioteca *aiohhtp*[3]. Ela oferece uma estrutura para criar servidores e permite o processamento de múltiplas solicitações simultaneamente sem bloqueios. O código foi desenvolvido para lidar com solicitações do tipo HTTP POST, enviadas para um *endpoint* interno previamente definido. Inspirando-se no protocolo *WebSocket*, o código foi projetado de maneira a evitar a necessidade de respostas HTTP GET. Isso contribui significativamente para a fluidez e responsividade da aplicação, pois elimina a sobrecarga de solicitações adicionais para buscar informações, tornando a interação com a aplicação mais ágil e eficiente.

Foi criada uma classe para a realização das predições de segmentação de imagens de lixos hospitalares utilizando o YOLO. Porém, também foram acrescentados outros métodos para o contexto do estudo, considerando a otimização para dispositiveis móveis e eficiência de recursos. Nessa circunstância, foi adicionada uma função para redimensionamento de imagem para 640x640 *pixels*, um formato adaptado para ser usado no YOLOv8. Para realizar a marcação das máscaras de segmentação nas imagens eviadas após a tentativa de detecção, foram utilizadas técnicas específicas do OpenCV (Seção 2.1.3), tanto para a marcação dos objetos quanto para a coloração das máscaras.

De forma similar, houve a criação da classe principal do servidor para ser responsável por receber os dados enviados externamente. Nesse caso, a imagem do dispositivo móvel. O formato definido para a imagem ser enviada codificada em *base64*, no tipo *String*. Essa classe fica como responsável de decodificar o texto e conectar com o processo da classe de predição, e em seguida codifica-la em *base64* para enviar como uma resposta JSON.

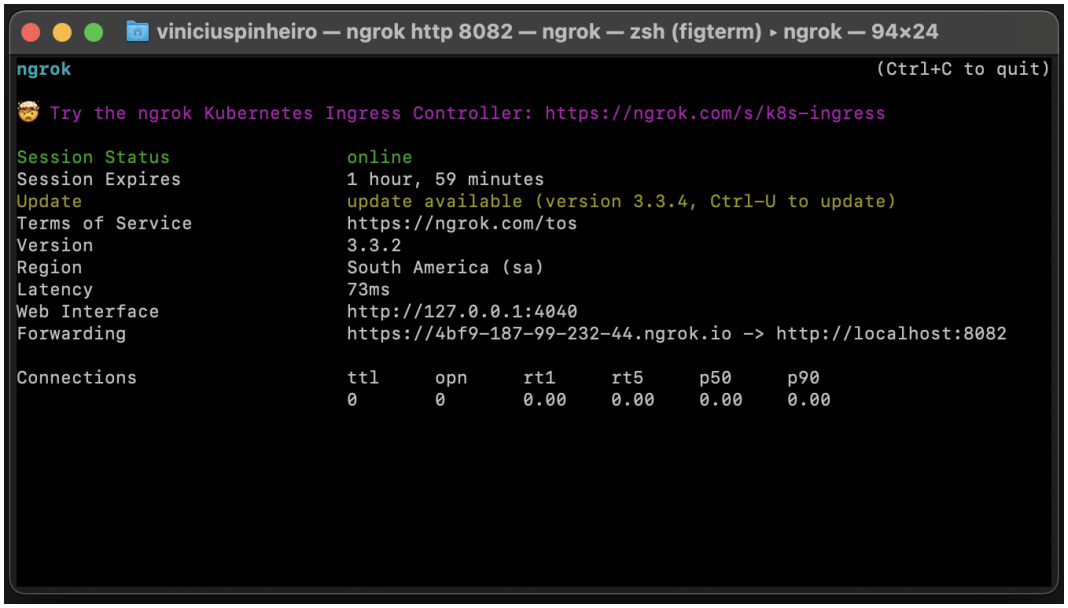
[3]Documentação da biblioteca: ⟨https://docs.aiohttp.org/en/stable/⟩

Em contextos que exigem velocidade e troca em tempo de real de informações, o protocolo *WebSocket* mantém uma conexão persistente bidirecional, sem precisar uma nova interação para que o servidor comunique a resposta. No escopo desse estudo, a aplicação real desse protocolo não fica necessária, mas aproveitando-se do conceito de uma comunicação bilateral entre chamada e resposta, adicionado ao fato da função de predição não estar atrelada a algum impeditivo, foi possível dispensar o uso da chamada de GET, deixando o resultado da predição atrelado a resposta assíncrona padrão já enviada pelo método POST.

Com essa estruturação de chamada e resposta do servidor, possibilitou a implementação mais simples de uma chamada de serviço no dispositivo móvel, além de por fim, agilizar a resposta ao aparelho que irá receber a imagem e o número de objetos.

Com o objetivo de disponibilizar o servidor local na internet de forma acessível e eficaz, optou-se por adotar a plataforma Ngrok, uma ferramenta de linha de comando (CLI) altamente versátil. O Ngrok desempenha um papel crucial ao criar um túnel seguro que atravessa barreiras como os NATs (Network Address Translators) e os *Firewalls*, tornando possível a exposição de serviços internos para a rede mundial. Essencialmente, ele age como um intermediário que estabelece uma conexão externa com o seu servidor local, permitindo que qualquer pessoa com o link de acesso acesse os serviços hospedados em sua máquina. Na Figura 24, é possível visualizar o Ngrok em ação, exibindo informações como o link de acesso direto ao seu servidor, o tempo restante antes da expiração do túnel e outros detalhes úteis.

Figura 24 – Ngrok em Funcionamento



Fonte: Própria

4.4 DESENVOLVIMENTO DO APLICATIVO MÓVEL

O aplicativo móvel foi desenvolvido com uma abordagem nativa de dispositivos iOS. O objetivo do *software* desenvolvido foi ser a ponte entre a imagem do descarte hospitalar e o servidor que realiza a predição, utilizando a camera disponível no momento em que o resíduo for encontrado.

Para realizar a elaboração do aplicativo, foi utilizada a IDE Xcode na versão 14.3.1 e seu código escrito em *Swift*. Durante a estruturação do projeto, sucedeu-se uma concorrência entre qual *Framework* de compontentes visuais seria o melhor para a aplicação: UIKit ou SwiftUI. Ambos possuem pontos fortes para agregar na execução do projeto, mas alguns fatores tem que ser levados em consideração.

Primeiramente, a abrangência de quantos equipamentos podem receber uma aplicação criada por algum deles. No caso, o UIKit por ser mais antigo, abrange uma vasta gama de dispositivos desde sua criação em 2008, enquanto o SwiftUI é parte do kit desenvolvimento mais moderno da *Apple*, criado em 2019 e com uma limitação de seus aplicativos exigirem como versão mínima o iOS 13, lançado no mesmo ano. Também considereando a questão de tempo, a documentação e materiais disponíveis do primeiro framework citado é bem maior que o do segundo. Logo, a escolha para uso foi o UIKit.

Outra escolha feita para melhor eficiência de tempo no projeto, foi a utilização de *Storyboard* no lugar de código visual exclusivamente escrito. O *Storyboard* é uma ferramenta visual que permite a construção das telas de maneira interativa, disponibilzando os componentes e recursos mais usados para agilizar a criação fiel das telas planejadas. Seu uso restringe personalizações mais complexas de componentes e cria dificuldades em uso extensivo de controle de versão. Ponderando as desvantagens descritas com o escopo do projeto, seu uso não conflita com a resoluçào aqui proposta.

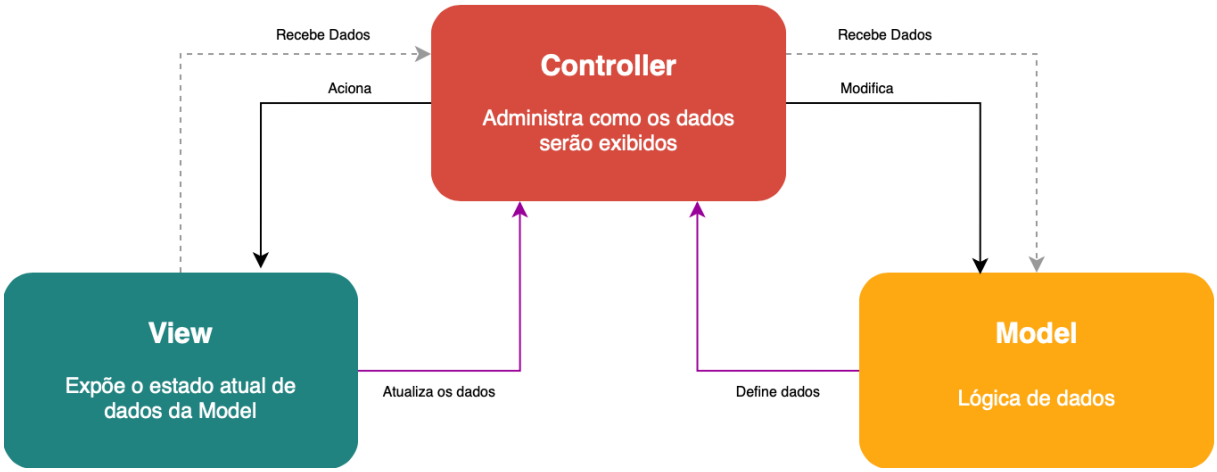
Considerando que esse é um estudo com requisitos moderados para exibição no *Front-end*, a arquitetura escolhida para o projeto foi o MVC, *Model-View-Controler*. Este é um padrão de arquitetura frenquentemente usado para desenvolvimento de softwares de baixo a médio grau. Assim, as classes do projeto foram dividas pelos seguintes diretórios:

1. **Model:** Na arquitetura, ele encapsula as representações de dados e funciona de maneira independentes da interface de usuário. Foi o diretório no qual a classe responsável pela conversão e desconversão de representações JSON para *Swift* foi mantido.
2. **View:** para essa parte da estrutura, são reservados os arquivos que renderizam a interface de usuário e não possui camada lógica. Neste local, foi dirigido o arquivo de *Storyboard*.
3. **Controller:** é camada que opera como um intermediário ligando os dados ao que é visto no aplicativo. Esse seguimento contém as classes de manipualção da interface

de usuário, coordenando as informações dispostas no modelo e refletindo visualmente o que a interação resultou. Somente uma classe de controle foi criada, ficando armazenada nesse diretório.

É valido ressaltar também que modificações e adequações podem ser feitas na arquitetura,seja para isolamento de interações específicas ou extensões aos frameworks disponibilizados pela *Apple* separados em arquivos. A aplicação aqui descrita buscou preservar a fidelidade ao conceito original da arquitetura demonstrado na Figura 25, mantendo sua separação de responsabilidades e reusabilidade das classes criadas.

Figura 25 – Diagrama da Arquitura MVC



Fonte: Própria

O *Storyboard* da aplicação empregou o uso da representação visual dos demais diretórios, em que foi colocado os componentes de exibição de imagens e outro para ser um botão que ligará seu toque a câmera do dispositivo. Otimizando o tamanho do componente que receberá a imagem para ocupar o maior espaço possível, foi adicionada uma barra de navegação para que o botão da câmera fique disposto ao usuário. Apesar disso, somente o botão e a componente da imagem foram referênciadas na camada controller considerando que os mesmos serão os únicos necessários de manipulação durante a interação do usuário com o *software*. Na figura 26 é possível checar como ficou a estrutura final da ferramenta.

Figura 26 – Uso dos recursos do Storyboard

Fonte: Própria

Guardando uma maior complexidade de desenvolvimento, a classe de controle foi a mais extensa durante a criação do aplicativo. Primeiramente, foram codificados os métodos de comunicação com o serviço, focando na construção de um algoritmo robusto para o melhor envio da imagem e identificação de erro nos pontos principais que deem margem a problemas com servidor. Na função principal de integração com o serviço externo é feita a conversão da imagem em uma *strings* que representa sua codificação em *base64*, logo após isso, uma solicitação HTTP POST é criada com a URL que irá receber a imagem. Prossegui-se configurando o cabeçalho e a tipagem de conteúdo e aceitação de JSON. Então a primeira tratativa é feita, onde verifica se existe algum problema na conversão dos dados para JSON quando os mesmos estão sendo adicionados ao corpo da solicitação para assim uma tarefa de sessão ser criada para enviar os dados ao servidor.

Dentro dessa tarefa, incluram-se mais algumas tratativas de erro. A menos profunda é caso aconteça um erro na solicitação POST e sua descrição. Outra de verificação de estado do HTTP, entendendo assim se o servidor está enviando uma resposta válida ou se está acontecendo algum erro por parte da comunicação. Verifica-se então se o serviço envia um retorno ou se comunicação responde vazio. Por fim, essa resposta é convertida de JSON para *Swift* e atualiza o componente de imagem e exibe um alerta contendo o número de objetos segmentados, e caso algo corrompido ou sem conversão seja recebido, é informado ao console do XCode.

Outras funções também foram criadas como a de conversão de *Strings* em *base64* para o formato de imagem proprietário da linguagem, além da função de atualização do componente de exibição. O método que faz a camera ser acionada está ligado ao botão, quando acionado realiza a abertura do *software* próprio da *Apple* para isso.

Para utilizar as camêras em equipamentos da marca, também fica necessário a permissão do usuário para seu uso. Nesse caso, é acrescentado uma critério adicional

nas configurações da construção do aplicativo, aqui sendo *"Privacy - Camera Usage Description"* ("Privacidade - Descrição do Uso de Câmera"em tradução livre) e, ao lado, a mensagem exibida ao usuário para informar sobre o uso. A parte modificada pode no arquivo info.plist, que contém esse parâmetro, pode ser vista na figura 27.

Figura 27 – Permissãos Adicionada no Arquivo info.plist

Fonte: Própria

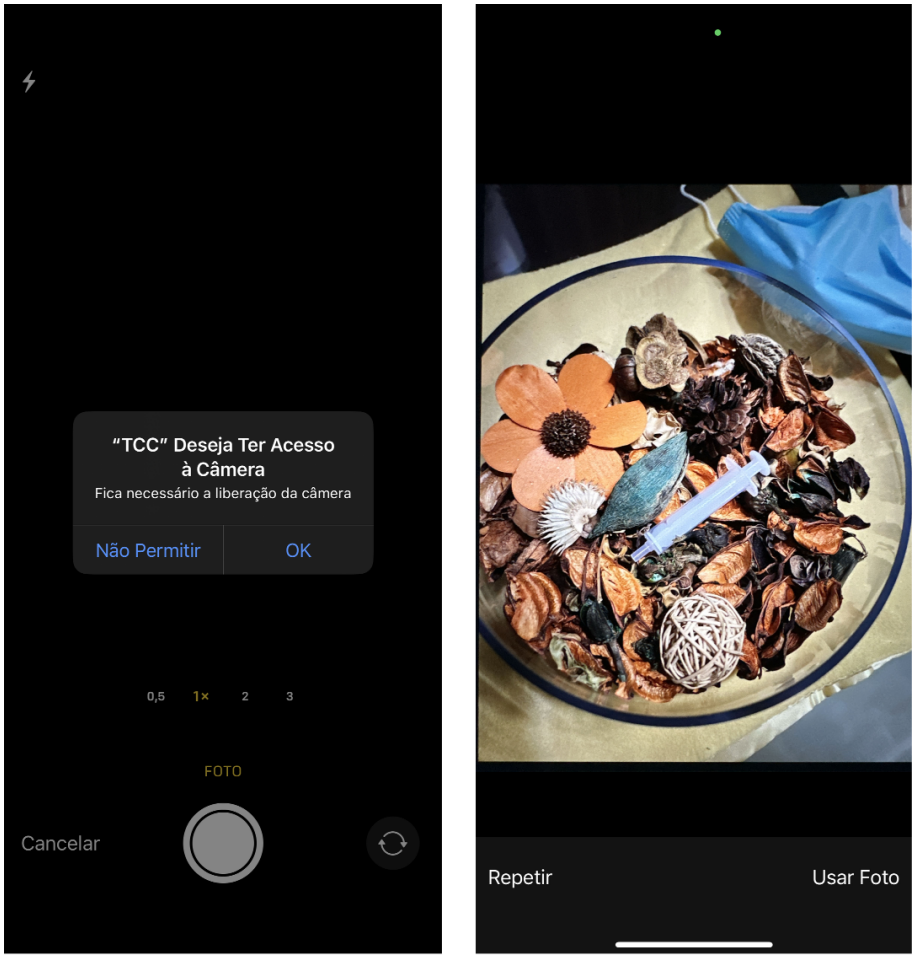
No instante em que o aplicativo é iniciado, o usuário é prontamente recebido com uma mensagem em tela, fazendo o pedido para que ele interaja por meio da ativação do botão da câmera. Esse engajamento é essencial para inaugurar o processo de verificação de resíduos, o qual, de maneira ilustrativa, encontra-se representado de maneira clara e visualmente explicativa, demonstrado na figura 28 em questão. Essa abordagem assegura que o usuário compreenda de forma intuitiva a ação necessária para iniciar a análise dos resíduos, estabelecendo uma conexão direta entre a ação de toque e o propósito funcional do aplicativo.

Figura 28 – Tela Inicial do Aplicativo

Fonte: Própria

A figura 29 demonstra o uso do módulo da câmera. Acionado quando o botão indicado é clicado, o componente é ativado no próprio do contexto do aplicativo, onde aparece uma notificação para a solicitação explícita de autorização ao uso da câmera, vista do lado esquerdo da imagem. Uma vez que a concessão é feita pelo usuário, e após a captura da imagem, sobrevém a etapa de validação, onde se demanda do usuário a confirmação da fotografia capturada, e a determinação se esta será enviada para a segmentação, demonstrado à direita.

Figura 29 – Uso do Módulo de Câmera no Aplicativo



Fonte: Própria

5 RESULTADOS E DISCUSSÕES

Foram obtidos um total de 12 modelos finais, distribuídos em conjuntos de quatro para cada variação pré-definida do YOLOv8. Para selecionar o modelo a ser empregado na resolução da problemática, foi realizada uma análise detalhada dos resultados obtidos. Para isso, utilizou-se as métricas de validação precisão, revocação, mAP50 e mAP50-95 que são referências para avaliação dos modelos com problemas de segmentação e detecção. Os modelos do YOLO calculam as métricas para o *bounding box*, que são as caixas que delimitam os objetos em interesse, e para as máscaras que delimitam as bordas dos objetos detectados pelo modelo. Neste trabalho, analisou-se as métricas das máscaras dos dados de validação da base de dados visto que é utilizado para problemas de segmentação de objetos como mencionado na Seção 2.1.2.

5.1 RESULTADOS DOS MODELOS

A partir da avaliação das métricas calculadas para cada modelo nas bases de validação, conforme apresentado na tabela 3, é possível identificar o modelo com melhor desempenho. Os melhores valores de cada métrica estão em destaques na cor verde, enquanto que os que estão em vermelho são menores valores para cada métrica. Os valores que estão sendo analisados são a média total juntando todas as 4 classes, isso por conta que o objetivo do trabalho é segmentar os lixos hospitalares de forma geral.

Tabela 3 – Métricas de Validação dos Modelos YOLOV8

Modelos	*Data Augmentation*	*Single Class*	P	R	mAP50	mAP50-95
YOLOv8m	False	False	0,65	0,60	0,60	0,35
	False	True	0,73	0,57	0,62	0,35
	True	False	0,70	0,57	0,62	0,40
	True	True	0,61	0,59	0,60	0,35
YOLOv8l	False	False	0,77	0,55	0,61	0,37
	False	True	0,67	0,60	0,63	0,37
	True	False	0,67	0,61	0,64	0,42
	True	True	0,81	0,58	0,65	0,40
YOLOv8x	False	False	0,65	0,54	0,58	0,36
	False	True	0,72	0,62	0,66	0,38
	True	False	0,76	0,58	0,67	0,43
	True	True	0,72	0,60	0,63	0,39

Fonte: Própria

Nota-se através da tabela 3 que os melhores resultados de revocação, mAP50 e mAP(50-95) se concentram nos modelos treinados com o YOLOv8x, o que era esperado visto que, pela Figura 7, o YOLOV8x é o modelo que obteve os melhores resultados de validação para segmentação com o *dataset* COCO. Enquanto que os valores mais baixos de precisão e mAP50-95 são do modelo YOLOv8m, que é o modelo com menos parâmetros e menores valores de métricas com o treinamento do *dataset* COCO dos 3 modelos pré-treinados selecionados de acordo com a Figura 7. Os modelos do YOLOV8l obtiveram os valores de métricas intermediários, porém ainda obteve o melhor resultado de precisão sendo 0,81 quando treinado com o aumento de dados e com a técnica de classe única ("*Single class*").

O modelo selecionado para implantar no aplicativo foi o do YOLOv8x treinado com o aumento de dados e sem a classe única, no qual obteve 0,76 de precisão, 0,58 de revocação, 0,67 de mAP50 e 0,43 de mAP50-95. O mesmo foi selecionado por conta dos melhores resultados de mAP50 e mAP50-95 que são métricas utilizadas para tarefas de detecção e segmentação apresentadas na Seção 2.2.5, além disso obteve valores de precisão e revocação aceitáveis.

5.1.1 ANÁLISE DO MODELO SELECIONADO

Verificado os resultados presentes na Tabela 3 e definido o modelo com um desempenho mais próximo do esperado, no qual foi o modelo YOLOV8x treinado com o aumento de dados e sem classe única, foi realizado uma análise do mesmo para verificar com melhor exatidão a qualidade do modelo treinado. Para isso, analisou-se as métricas de precisão, revocação, mAP50 e mAP50-95 para cada classe que foi segmentada, além de analisar as curvas Precisão-Revocação, Precisão-Confiança e Revocação-Confiança junto com a matriz de confusão do modelo.

- **Precisão, revocação, mAP50 e mAP50-95**

Na Tabela 4, é verificado os valores de precisão, revocação, mAP50 e mAP50-95 para cada classe (bolsas de sangue, máscaras, ampolas e seringas) na base de validação. Nota-se que o objeto que possui as melhores métricas são as bolsas de sangue, com 0.661 de mAP50-95 e 0.871 de mAP50, logo verifica-se que o modelo está conseguindo segmentar os objetos próximo do esperado, além disso a precisão e a revocação também estão com os melhores valores dentre as classes da base de dados com 0.885 e 0.884, respectivamente, ou seja, pode-se considerar que o modelo aprendeu padrões que diferencie a classe das outras.

Tabela 4 – Métricas de validação para cada classe do modelo selecionado

Classe	P	R	mAP50	mAP50-95
Bolsa de sangue (“*Blood bag*”)	0.885	0.884	0.871	0.661
Máscara (“*Mask*”)	0.731	0.692	0.776	0.56
Ampola (“*Ampoule*”)	0.505	0.456	0.467	0.254
Seringa (“*Syringe*”)	0.64	0.435	0.496	0.249
Todas as classes	0,76	0,58	0,67	0,43

Fonte: Própria

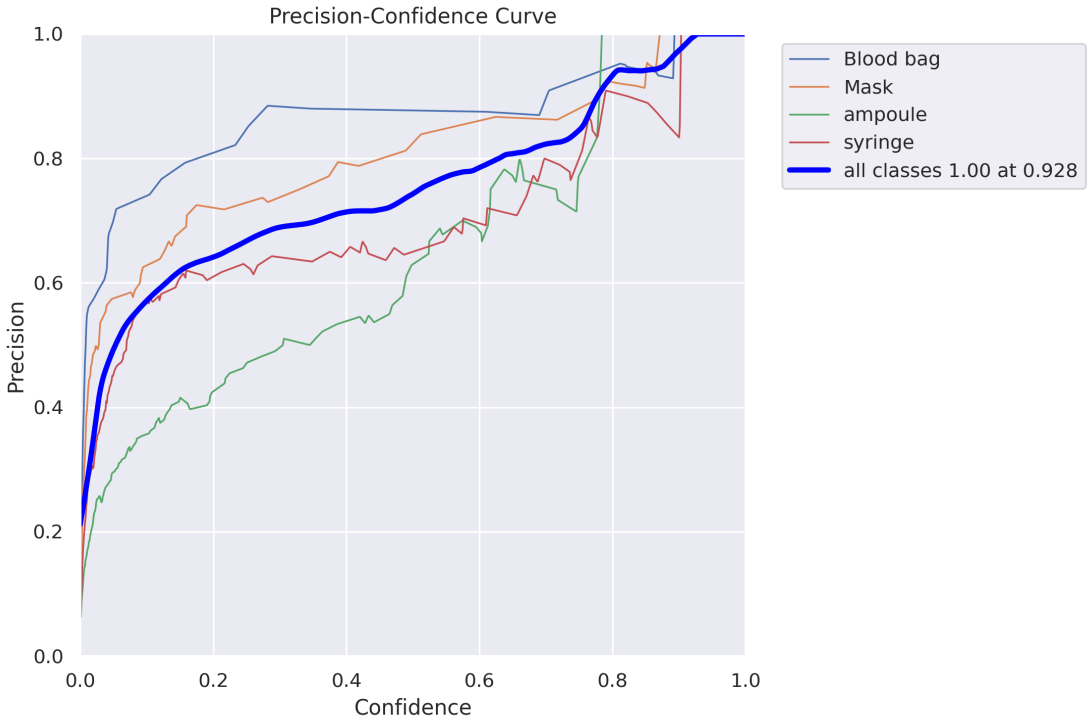
Todavia, observa-se que o modelo sentiu dificuldade para segmentar as classes da ampola e seringa visto que ambas obtiveram mAP50 próximos de 0.25 e mAP50-95 de 0.467 e 0.496, respectivamente. Além disso, a precisão e a revocação de ambas também foram as menores comparados com os valores das outras classes, o que mostra que o modelo pode não ter encontrados padrões suficientes para conseguir classificá-las e diferenciá-las das outras classes. Isso pode ser devido ao número de objetos presentes na base de validação ou também pelo formato dos objetos.

- **Curva Precisão - Confiança**

Um dos gráficos que o YOLO fornece com a finalização do treinamento é o gráfico da Curva de Precisão-Confiança do modelo apresentado na Figura 30. Primeiramente, a confiança de um modelo refere-se à medida de certeza nos resultados de previsão gerado pelo modelo, ou seja, é o quanto o modelo possui a certeza da sua predição. Portando, o gráfico apresentado na figura 30 mostra o valor de precisão de acordo com a confiança do modelo para cada classe e de forma geral.

Nota-se no gráfico que quanto maior é a confiança maior é a precisão, que é um comportamento esperado. Sendo este, uma vez que a elevada confiança do modelo indica maior certeza nos resultados consequentemente em menos falsos positivos e, por conta disso, um aumento no valor da precisão. Por exemplo, supondo que o modelo tem uma precisão de 90%. Isso significa que, em 90% dos casos, ele identifica corretamente um lixo hospitalar em uma imagem. Se o modelo tiver alta confiança, é mais provável que esteja correto nos 90% dos casos, caso contrário, é mais provável que esteja errado nos 10% dos casos.

Figura 30 – Curva de Precisão - Confiança



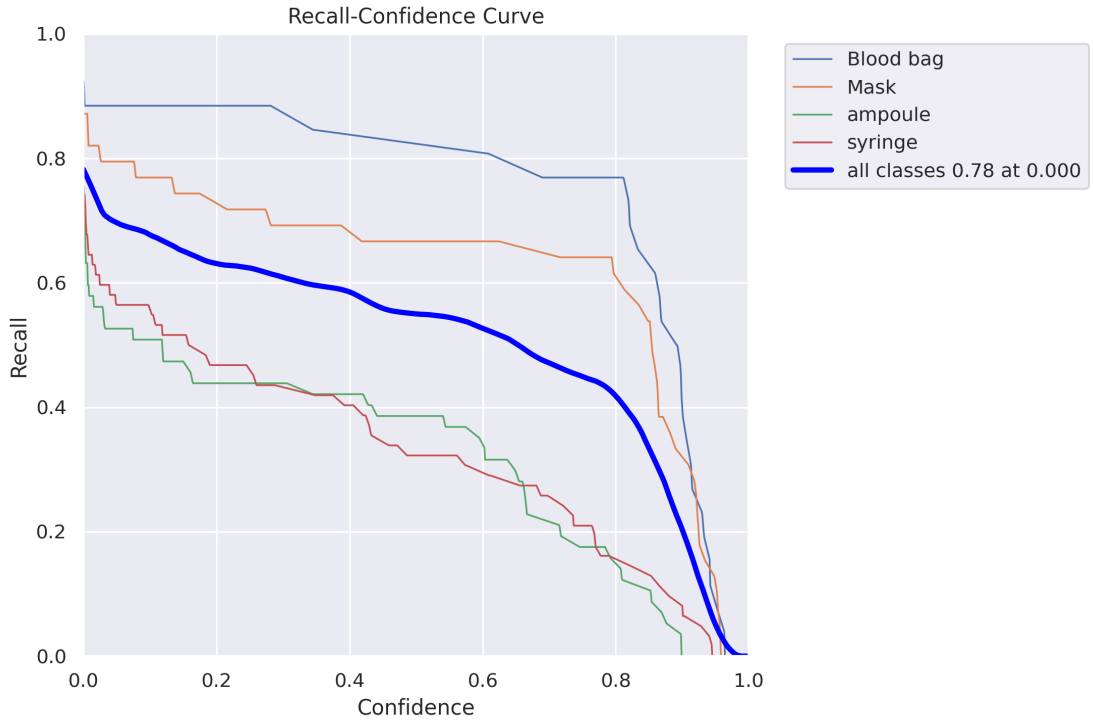
Fonte: Própria

- **Curva Revocação - Confiança**

Outro gráfico fornecido pelo YOLO ao fim da predição é o da Curva de Revocação-Confiança, ilustrado na Figura 31. A curva de revocação - confiança é um gráfico que mostra a relação entre a revocação e a confiança de um modelo de aprendizado de máquina. O gráfico ideal seria uma linha diagonal de cima para baixo. No entanto, na prática, as curvas não ficam dessa forma devido ao modelos não serem perfeitos.

Contudo, é um grafico para avaliar como o modelo se comporta em diferentes níveis de confiança e como a capacidade de recuperação se modifica conforme essa confiança varia. Por exemplo, ao analisar a curva, podemos observar que em confianças mais altas, a revocação tende a diminuir, indicando que o modelo é mais consistente em identificar corretamente os exemplos positivos, porém com uma confiança mais elevada, enquanto seu oposto também corresponde ao contrário.

Figura 31 – Curva de Revocação - Confiança



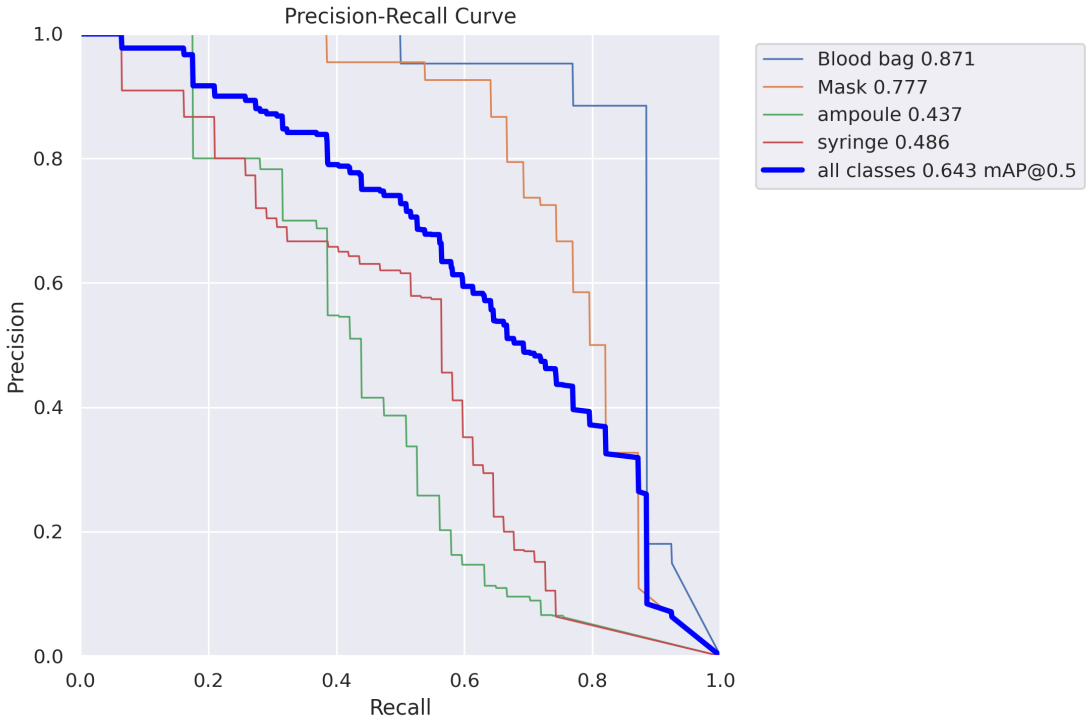
Fonte: Própria

- **Curva Precisão - Revocação**

A análise da curva precisão-revocação ilustra o desempenho do YOLO em vários limiares de confiança. Considerando o que foi explicado na sessão 2.2.5, é demonstrado que interseção ideal para os pontos dos dois parâmetros forma 90 graus. Isso ilustra como um cenário de equilíbrio pode ser visto entre detectar com precisão e capturar objetos variados.

Na análise deste estudo, fica evidente que o modelo se destaca na detecção de bolsas de sangue em comparação com outras classes de objetos. Isso é especialmente notável mesmo quando comparamos a curva correspondente para identificação das máscaras. Ambas as curvas que representam essas duas categorias superam a curva média, revelando a eficiência ao lidar com esses objetos específicos. Por outro lado, os resultados menos favoráveis estão relacionados às seringas e ampolas, que não apenas permanecem abaixo da curva média, mas também exibem instabilidade ao variar as métricas.

Figura 32 – Curva de Precisão - Revocação



Fonte: Própria

- **Matriz de Confusão**

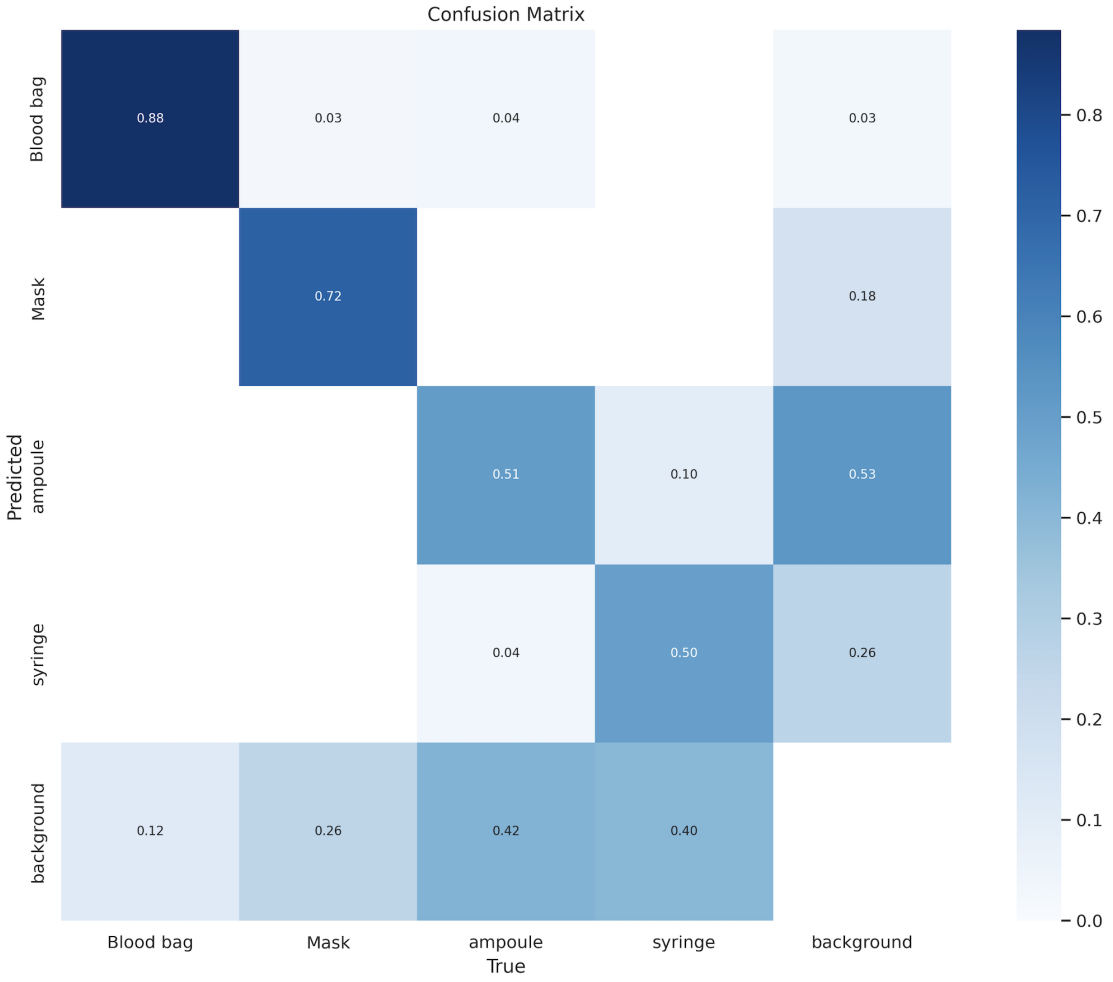
Ao analisar os resultados da matriz, nota-se que, no caso das bolsas de sangue, houve uma perda de 12% ao confundir com o plano de fundo. Isso quer dizer que essa porcentagem não foi identificada. No que diz respeito às máscaras, o modelo não as confundiu com ampolas ou seringas, porém teve uma confusão de 3% com as bolsas de sangue e uma perda significativa de 26% ao considerar o plano de fundo. Não apresentando tantos problemas, visto que foram os objetos de melhor desempenho do modelo.

Quando a atenção é voltada para as ampolas e seringas, identifica-se algumas inconsistências. No caso das ampolas, 51% foram identificadas corretamente, mas 42% foram erroneamente consideradas como plano de fundo, deixando de segmentá-las, enquanto 4% das ampolas foram classificadas como seringas e 4% como bolsas de sangue. Quanto às seringas, 50% foram corretamente identificadas, mas 40% não foram reconhecidas, e em 10% dos casos, o modelo as confundiu com ampolas.

Estes equívocos podem ser atribuídos às diferentes formas e tamanhos dos objetos, visto que as máscaras e bolsas de sangue são maiores, enquanto as seringas e ampolas são menores. Evidencia que o modelo enfrentou dificuldades na segmentação desses objetos menores, muitas vezes perdendo o objeto de interesse para o plano de fundo. Esse desafio poderia ser minimizado ao aprimorar tanto a quantidade quanto a qualidade das imagens na base de dados.

Apesar das limitações, é notável que, mesmo nas classes de desempenho menos satisfatório, a taxa de acerto se manteve acima de 50%.

Figura 33 – Matriz de Confusão



Fonte: Própria

- **Visualizando os resultados do modelo**

Feito a análise das métricas, o próximo passo foi verificar nas imagens de validação e nas imagens de teste como o modelo segmentou cada objeto. Isso auxilia para visualização se o modelo está segmentando e detectando os objetos.

A Figura 34 apresenta uma amostra fornecida pelo YOLO do resultado do modelo na base de validação. Nessa figura, é visto os objetos com suas respectivas máscaras, *"bounding box"* e a confiança que o modelo obteve na detecção e classificação do objeto. Pode-se notar que na segunda imagem da primeira linha, o modelo confundiu uma seringa com uma ampola, o que comprova a análise da matriz de confusão e das métricas anteriormente. Isso pode ter sido por conta do formato do objeto.

Figura 34 – Resultado do modelo para os dados de validação

Fonte: Própria

Já na Figura 35, é visualizado o resultado do modelo para uma amostra do conjunto de dados de teste, que são dados que o modelo não viu durando o processo de treinamento. Nota-se que o modelo conseguiu detectar e segmentar os objetos de interesse, além de classificá-los entre seringa, ampola, bolsa de sangue e máscara apesar dos resultados das classes de seringa e ampola obterem as menores métricas. Além disso, também tem a informação da confiança do modelo para cada objeto detectado e segmentado.

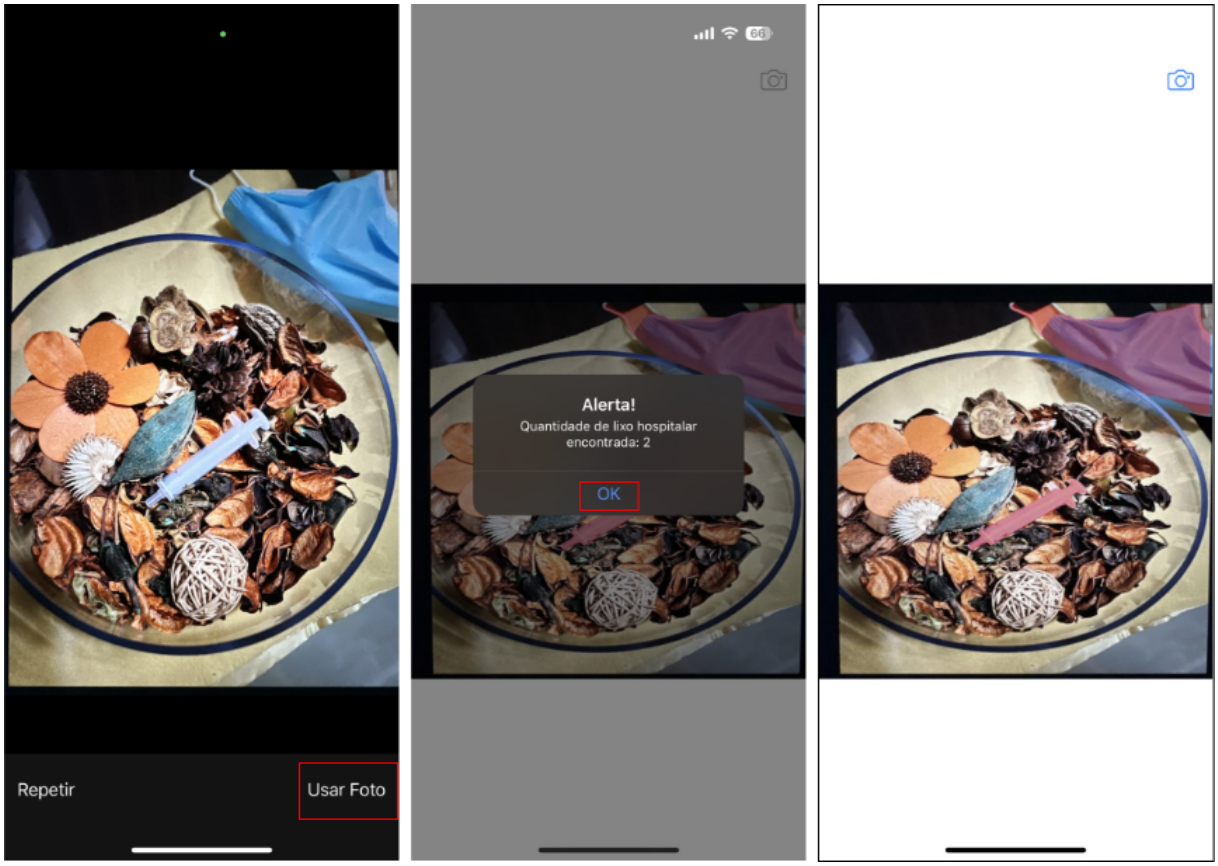
Figura 35 – Resultado do modelo para os dados de teste

Fonte: Própria

5.2 CASO DE USO - APLICATIVO

Utilizando a aplicação desenvolvida na seção 4.4, uma imagem capturada pelo iPhone é enviada para processamento na nuvem, onde um modelo escolhido de segmentação está aplicado. Conforme a imagem é transmitida, o aplicativo prontamente apresenta um alerta exibindo o número de resíduos hospitalares detectados na fotografia. Além disso, uma representação visual da foto, na qual os elementos de resíduos estão destacados por meio de segmentação, é exibida na página principal do aplicativo.

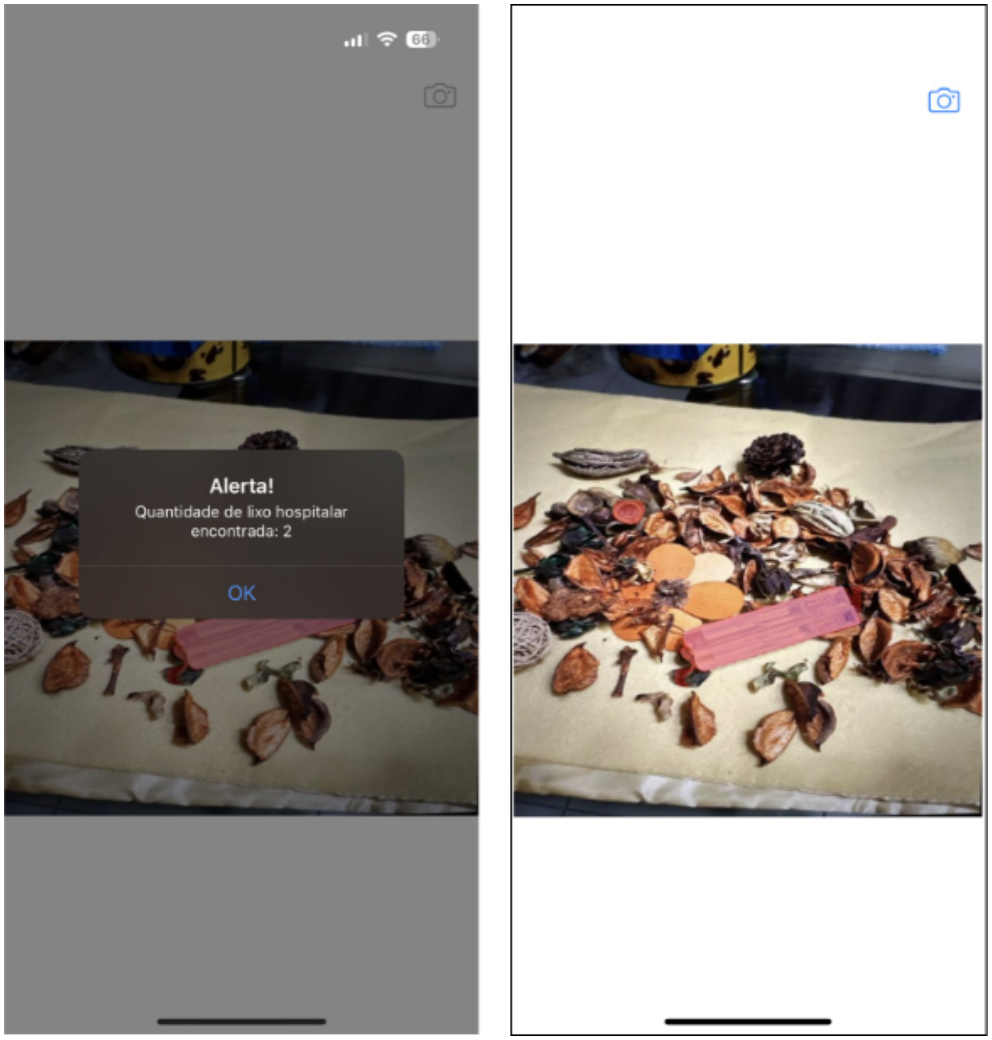
Figura 36 – Telas de Resultado do Modelo



Fonte: Própria

Através da Figura 36, é possível examinar um exemplo de saída do modelo e como o aplicativo interage com elas. O processo se inicia com a captura de uma fotografia que é enviada ao servidor para processamento clicando no botão “Usar Foto”. Uma vez que a predição é recebida do servidor, o aplicativo apresenta um alerta que ilustra a quantidade de objetos identificados pelo modelo. No exemplo ilustrado na figura 36, o alerta reporta a presença de dois itens distintos: uma máscara e uma seringa. Outro exemplo de saída do modelo e apresentar o resultado no aplicativo é visto na Figura 37, no qual o alerta identifica dois elementos, desta vez, duas ampolas.

Figura 37 – Exemplo de tela do Resultado do Modelo



Fonte: Própria

Após a notificação ser desativada, a última imagem segmentada fica disponível na tela principal para que o usuário possa realizar uma análise autônoma dos objetos que foram isolados pelo processo de segmentação. Essa abordagem permite que o usuário avalie com precisão os objetos delineados pela máscara de segmentação presentes na imagem.

6 CONSIDERAÇÕES FINAIS

A gestão inadequada dos resíduos hospitalares representa um perigo considerável aos ambientes urbanos devido à contaminação por microorganismos infecciosos e substâncias radioativas. Além disso, essa problemática foi agravada com a pandemia da COVID-19, que gerou uma sobrecarga nos sistemas de tratamento de resíduos devido ao acréscimo de materiais contaminados. Contudo, este trabalho teve o intuito de implementar um aplicativo que utiliza um modelo de aprendizado profundo para segmentar e identificar possíveis descartes irregulares de lixo hospitalar.

A metodologia aplicada consistiu em 4 etapas: criação e preparação da base de dados, treino do modelo, disponibilização do modelo em rede e, por último, a criação do aplicativo com a requisição do resultado do modelo utilizando a linguagem *Swift*.

Houve 5 procedimentos para a criação e preparação da base de dados, sendo eles: coletar as imagens por diferentes fontes devido a escassez de fotografias do tema específico; o processo de construção das máscaras de segmentação em cada imagem com sua rotulação identificando cada um dos objetos definidos; a preparação do conjunto de dados, onde foram adpatadas aos tamanho do modelo, como também separadas em treino, validação e teste; e, por último, o uso de técnicas para aumento de dados, fazendo as imagens contidas serem espelhadas e rotacionadas, criando um segundo dataset.

Na etapa de treinamento, foram realizados o uso de 12 modelos com uma técnica específica para melhorar o desempenho, sendo ela: a *single class*. O treinamento foi realizdo utilizando os modelos de segmentação de imagem disponíveis pela arquitetura YOLOv8 e tiveram o otimizador Adam na sua configuração. Para validar os modelos, utilizou-se as principais métricsa para problemas de segmentação, como a precisão, a revocação e o mAP50. As métricas foram disponibilizadas na tabela 3.

Fazendo a análise de forma geral o modelo com melhor desempenho foi o YOLOv8x, possuindo os maiores valores de métricas de aferição, destacando-se quando aplicado treino com o dataset em que foi realizado o aumento de dados por possuir valores balanceados quando comparado as demais variações feitas, sendo a opção escolhida para uso no projeto. Também destaca-se negativamente o YOLOv8m ter o pior resultado de precisão em métricas quando aplicada para simplificação de classes, mesmo com o aumento de dados.

Com o modelo definido, foi criado um servidor local no qual abrigou o uso do algoritmo de predição para ser interligado com a aplicação por meio de um método se solicitação web assíncrono. Nele, recebe-se uma imagem que será enviada pelo aplicativo codificada em *base64* e ocorre a conversão para PNG. O arquivo convertido é analisado pelo modelo e caso encontre-se um descarte, ocorre o desenho da máscara por meio de funções do OpenCV e é enviada novamente de maneira codificada para o software.

O aplicativo foi desenvolvido em *Swift* de forma nativa para os sistemas iOS. Para

o envio da imagem foi criado um botão interligado com a camêra seguindo de um método de serviço para quando o usuário desejar enviar a foto. A chamada de serviço executa um POST que recebe como resposta a própria imagem, que é decodificada, e o número de objeto segmentados. Esse número é exibido em um alerta e a fotografia segmentada é colocada em exibição na tela pelo componente de imagens proprietário.

No presente estudo foram utilizados datasets limitados com imagens vindas de edição e geradas artificialmente, o que pode ter influenciado o resultado dos modelos desenvolvidos. Sugere-se como mudança o crescimento do conjunto de dados, assim como, a aplicação de mais técnicas aos modelos visando a melhora de resultado das métricas escolhidas. Além disso, a concepção do aplicativo para operar em diversas plataformas e a hospedagem do algoritmo de previsão em um serviço de nuvem com maior capacidade de recursos podem ampliar o alcance do projeto para um público mais amplo.

REFERÊNCIAS BIBLIOGRÁFICAS

ACADEMY, D. S. ***Deep Learning book***. 2022. Disponível em: ⟨https://www.deeplearningbook.com.br/⟩. Acesso em: 20/08/2023.

ALBANEZ, D. de O.; BATISTA, M. A.; SILVA, S. F. da. O que é segmentação de imagens? 2016. Disponível em: ⟨https://files.cercomp.ufg.br/weby/up/615/o/O_que_e_segmentacao_de_imagens.pdf?1481557289⟩.

ANALYTICS, S. ***Strategy Analytics: Half the World Owns a Smartphone***. 2021. Disponível em: ⟨https://news.strategyanalytics.com/press-releases/press-release-details/2021/Strategy-Analytics-Half-the-World-Owns-a-Smartphone/default.aspx⟩. Acesso em: 24/09/2022.

APPLE. ***Xcode 15 - Apple Developer***. 2023. Disponível em: ⟨https://developer.apple.com/xcode/⟩. Acesso em: 01/09/2023.

BUDUMA, N.; LACASCIO, N. ***Fundamentals of Deep Learning***. [S.l.]: O'Reilly, 2017.

CHOLLET, F. ***Deep Learning with Python***. [S.l.]: Manning, 2018.

COUTO, L. M. ***Como iniciar a sua jornada na nuvem AWS — O blog da AWS***. 2022. Disponível em: ⟨https://aws.amazon.com/pt/blogs/aws-brasil/como-iniciar-a-sua-jornada-na-nuvem-aws/⟩. Acesso em: 15/09/2022.

GEETHA, S. et al. Design of waste management system using ensemble neural networks. ***MDPI***, 2022. Disponível em: ⟨https://www.mdpi.com/2411-9660/6/2/27⟩.

GONZALES, R. C.; WOODS, R. E. ***Processamento Digital de Imagens***. Third edition. [S.l.]: Pearson, 2010.

GUILHERME. ***Servidor Web Local: O que é, e como fazer um Agora Mesmo***. 2019. Disponível em: ⟨https://blog.dankicode.com/servidor-web-local/⟩.

GULLI, A.; PAL, S. ***Deep Learning with Keras***. [S.l.]: Packt, 2017.

HUI, J. ***mAP (mean Average Precision) for Object Detection)***. 2018. Disponível em: ⟨https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173⟩.

IBM, I. B. M. C. ***What is Computer Vision***. 2022. Disponível em: ⟨https://www.ibm.com/topics/computer-vision⟩. Acesso em: 15/09/2022.

JIN, D. et al. ***Transfer learning and robustness for natural language processing***. Tese (Doutorado) — Massachusetts Institute of Technology, 2020.

JOCHER, G.; CHAURASIA, A.; QIU, J. ***Ultralytics YOLOv8***. 2023. Disponível em: ⟨https://github.com/ultralytics/ultralytics⟩.

KHAN, S. et al. ***A Guide to Convolutional Neural Networks for Computer Vision***. [S.l.]: Morgan & Claypool, 2018.

LIU, Y. H. ***Python Machine Learning By Example***. [S.l.]: Packt, 2017.

MADRAS, I. ***Deep Learning for Computer Vision***. 2022. Disponível em: ⟨https://nptel.ac.in/courses/106/106/106106224/⟩. Acesso em: 05/10/2022.

MICROSOFT. ***Visual Studio Code - Code Editing. Redefined***. 2023. Disponível em: ⟨https://code.visualstudio.com⟩. Acesso em: 01/09/2023.

MINAEE, S. et al. Image segmentation using deep learning: A survey. IEEE, 2020. Disponível em: ⟨https://arxiv.org/pdf/2001.05566.pdf⟩.

MNCR, M. nacional dos Catadores de M. R. ***Quantos Catadores existem em atividade no Brasil?*** 2017. Disponível em: ⟨https://www.mncr.org.br/sobre-o-mncr/duvidas-frequentes/quantos-catadores-existem-em-atividade-no-brasil⟩. Acesso em: 21/09/2022.

MYTHILI, T.; ANBARASI, A. Enhanced segmentation network with deep learning for biomedical waste classification. ***INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY***, 2020. Disponível em: ⟨https://doi.org/10.17485/IJST/v14i2.2137⟩.

ONU, O. das N. U. ***Objetivos de Desenvolvimento Sustentável no Brasil***. 2022. Disponível em: ⟨https://brasil.un.org/pt-br/sdgs⟩. Acesso em: 15/09/2022.

OPAS, O. P.-A. da S. ***Toneladas de resíduos de serviços de saúde para COVID-19 expõem necessidade urgente de melhorar sistemas de gerenciamento de resíduos***. 2022. Disponível em: ⟨https://www.paho.org/pt/noticias/1-2-2022-toneladas-residuos-servicos-saude-para-covid-19-expoem-necessidade-urgente⟩. Acesso em: 21/09/2022.

OPENCV. ***About OpenCV***. 2023. Disponível em: ⟨https://opencv.org/about/⟩. Acesso em: 20/05/2023.

PAIXãO, J. R. da. ***Ngrok: do Localhost para o Mundo***. 2021. Disponível em: ⟨https://medium.com/desenvolvendo-com-paixao/ngrok-do-localhost-para-o-mundo-5445ad08419⟩. Acesso em: 15/07/2023.

PATKI, N.; WEDGE, R.; VEERAMACHANENI, K. The synthetic data vault. In: IEEE. ***2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)***. [S.l.], 2016. p. 399–410.

POZZETTI, V. C.; MONTEVERDE, J. F. S. Gerenciamento ambiental e descarte do lixo hospitalar. 2017. Disponível em: ⟨http://revista.domhelder.edu.br/index.php/veredas/article/view/949⟩.

RAMESH, A. et al. Hierarchical text-conditional image generation with clip latents. ***arXiv preprint arXiv:2204.06125***, v. 1, n. 2, p. 3, 2022.

REBOUCAS, M. de S. C.

An Empirical Study on the Usage of the SWIFT Programming Language, Janeiro 2016. Monografia (Bacharel em Ciência da Computação), UFPE (Universidade Federal de Pernambuco), Recife, Brasil.

REDHAT. ***What are cloud services?*** 2022. Disponível em: ⟨https://www.redhat.com/en/topics/cloud-computing/what-are-cloud-services⟩. Acesso em: 15/09/2022.

ROBOFLOW. ***Roboflow***. 2023. Disponível em: ⟨https://roboflow.com⟩. Acesso em: 15/07/2023.

SAFETEC. ***Servidor local ou servidor na nuvem: quais as principais diferenças?*** 2022. Disponível em: ⟨https://www.safetec.com.br/cloud-computing/servidor-local-ou-servidor-na-nuvem/⟩.

SAXENA, S. ***Precision-Recall Curve***. 2022. Disponível em: ⟨https://pub.towardsai.net/precision-recall-curve-26f9e7984add⟩. Acesso em: 15/07/2023.

SIBY, A.; GS, M. A. Android hacking using msfvenom: Integrating ngrok. 2020.

SOLAWETZ, J. ***Roboflow Blog***. 2020. Disponível em: ⟨https://blog.roboflow.com/mean-average-precision/⟩.

SOUSA, B. A. de et al. Lixo hospitalar e seus impactos no meio ambiente. 2016. Disponível em: ⟨http://www.faculdadealfredonasser.edu.br/files/Pesquisar_5/21-11-2016-21.46.06.pdf⟩.

SWIFT.ORG. ***Welcome to Swift.org***. 2022. Disponível em: ⟨https://www.swift.org⟩. Acesso em: 21/09/2022.

TERVEN, J.; CORDOVA-ESPARZA, D. A comprehensive review of yolo: From yolov1 to yolov8 and beyond. ***arXiv preprint arXiv:2304.00501***, 2023.

ULTRALYTICS. ***YOLOv8 - Ultralytics YOLOv8 Docs***. 2023. Disponível em: ⟨https://docs.ultralytics.com/models/yolov8⟩. Acesso em: 05/07/2023.

VO, A. H. et al. A novel framework for trash classification using deep transfer learning. 2019. Disponível em: ⟨https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8930948⟩.

YU, K. ***Multi-classification and Object Detection in Intelligent Manufacturing***. Dissertação (Dissertação de Mestrado) — Massachusetts Institute of Technology, 2020.