UFRR

UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

NATÁLIA RIBEIRO DE ALMADA

Detecção Inteligente de Quedas em Idosos: Uma Abordagem Comparativa com Modelos YOLO

Boa Vista - RR
20 de março de 2026

Detecção Inteligente de Quedas em Idosos: Uma Abordagem Comparativa com Modelos YOLO

NATÁLIA RIBEIRO DE ALMADA

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação do Departamento de Ciência da Computação da Universidade Federal de Roraima, em cumprimento às exigências legais como requisito à obtenção do título Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Herbert Oliveira Rocha.

Boa Vista - RR
20 de março de 2026

AGRADECIMENTOS

À Universidade Federal de Roraima (UFRR), pela oportunidade de formação no curso de Ciência da Computação.

Em especial, ao Professor Dr. Herbert de Oliveira Rocha, pela paciência, dedicação e ajuda ao longo do curso e na realização deste trabalho.

Às Professoras Dra. Marcelle Alencar Urquiza, MSc. Thais Oliveira Almeida e ao Dr Marcelo Henrique Oliveira Henklain por aceitarem participar da banca de defesa e por suas relevantes contribuições ao documento.

A quase todos os professores do curso de Ciência da Computação, por seus ensinamentos e conselhos.

Aos meus pais, José Carlos Santos de Almada e Marina Benigna Ferreira de Almada, pelo amor, incentivo, apoio e conselhos que me ajudaram a prosseguir.

Aos meus familiares que de alguma forma contribuíram para o meu crescimento.

Aos meus amigos pessoais, Ohara, Guilherme, Amanda, John, Camila, Bárbara e Luigi, pela amizade e paciência. E aos colegas que fiz ao longo da graduação, pela convivência e aprendizado conjunto.

Em especial, a José Lucas Queiroz Lucena, por me aturar nesses últimos quatro anos, pelos inúmeros cafés da madrugada e pelas muitas horas de vídeo-cacetadas assistidas.

A todos que contribuíram, direta ou indiretamente, o meu obrigada!

E... a mim mesma por não ter desistido, mesmo quando parecia não ter fim!

Boa Vista - RR
20 de março de 2026

“Além disso, não é da minha natureza cair sem lutar (...)”

— Suzanne Collins

[PÁGINA PARA INSERÇÃO DA FICHA CATALOGRÁFICA]

Este documento será um pdf disponibilizado pela biblioteca da ufrr ao final do seu tcc 2. Para isso, você precisará enviar o seu tcc 2 em pdf para o e-mail processamento.tecnico@ufrr.br para mais informações, consulte https://ufrr.br/bibliotecas/elaboracao-de-ficha-catalografica . O pdf recebido será uma página. você poderá converter em imagem e inserir aqui ou, tendo o seu tcc 2 em pdf, é só inserir essa página adicional a partir de um software editor de pdf.

[PÁGINA PARA INSERÇÃO DA ATA DE APROVAÇÃO DO TCC 2]

Este documento será um pdf de uma página disponibilizado pelo seu orientador ao final do TCC 2. Ao receber o arquivo, basta inserí-lo aqui.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Detecção Inteligente de Quedas em Idosos: Uma Abordagem Comparativa com Modelos YOLO

NATÁLIA RIBEIRO DE ALMADA

Curso de Bacharelado em Ciência da Computação
Departamento de Ciência da Computação
Universidade Federal de Roraima (UFRR)
Boa Vista – RR – Brasil

{almada.compsci}@gmail.com

Resumo. *O envelhecimento da população traz consigo um aumento significativo de risco de quedas de idosos, que trazem consequências adversas, tais como lesões, hospitalizações e declínio funcional. Sendo assim, se faz necessária a intervenção imediata nas situações de queda, visando melhorar a segurança e bem-estar da população idosa, e para isso, modalidades de Tecnologias Assistivas seguem sendo aperfeiçoadas. Esta pesquisa propõe a aplicação e comparação de modelos de visão computacional YOLO para detecção de quedas de pessoas idosas a partir de vídeos, e empregando técnicas de classificação para identificar padrões que são característicos de quedas. Através dessa abordagem, visamos minimizar os efeitos negativos desses eventos, e propondo a notificação caso haja evento de queda detectado em vídeo, obtendo resultados promissores.*

Palavras-chave*: Visão Computacional , Idosos , Monitoramento de Quedas.*

Abstract. *The aging of the population brings with it a significant increase in the risk of falls among the elderly, which leads to adverse consequences, such as injuries, hospitalizations, and functional decline. Therefore, immediate intervention in fall situations is necessary in order to promote the safety and well-being of the elderly population. To this end, various modalities of Assistive Technologies have been continuously improved. This research proposes the application and comparison of YOLO computer vision models for detecting falls in elderly individuals using video data, employing classification techniques to identify patterns characteristic of such events. Through this approach, the aim is to minimize the negative impacts of falls and enable the generation of notifications when a fall is detected in video, obtaining promising results.*

Keywords*: Computer Vision, Elderly, Monitoring Falls.*

1. Introdução

O aumento expressivo da população idosa a níveis globais não é novidade e tem sido objeto de estudo e preocupação governamental, no que se refere a prover qualidade de vida adequada para esse público (ONU, 2023). No Brasil, foi registado o aumento de 54% da população sênior nas últimas décadas, totalizando assim mais de 22 milhões de pessoas com idade acima de 65 anos (GOMES; BRITTO, 2023), e diversas causas são atribuídas para que esse envelhecimento populacional ocorra, o mais relevante tem se mostrado a transição demográfica que implica na redução das taxas de fertilidade e de mortalidade, essa última estando diretamente ligada aos avanços da medicina.

Constatando essa problemática, as Tecnologias Assistivas (TAs) se mostram grandes aliadas para o cuidado e manutenção da saúde de idosos, pois apresentam soluções tecnológicas viáveis para resolver os problemas de ordem física e mental. Já ultrapassam um bilhão de pessoas que necessitam nos dias atuais de tecnologias assistivas (World Intellectual Property Organization, 2021) e auxiliar pessoas idosas, que por padrão fisiológico, têm dificuldade para se locomover e problemas de memória recorrentes é primordial, pois nem sempre o idoso pode contar com rede de apoio, e muitas vezes acabam sendo institucionalizados e não recebem os cuidados adequados (ANDRADE; PEREIRA, 2009).

É perceptível que devido aos percalços que acompanham o avanço da idade, esse público alvo detém uma taxa alta de casos de acidentes domésticos, e se percebem desamparadas, feridas e até mesmo chegam a óbito, visto que do período de 2010 a 2018 foram registradas 7.719 declarações de óbitos de idosos no Brasil, tendo como a causa principal, as quedas em ambiente doméstico, dado que pode ser entendido como situação crítica e urgente de saúde pública (DATASUS, 2024). A partir desse cenário, analisar um sistema para detecção de quedas de idosos utilizando visão computacional se faz necessário para buscar soluções aplicáveis para a realidade do cuidado com a saúde da comunidade de pessoas acima dos 60 anos.

A exemplo de pesquisa como a de Ximenes e Ziza (2022) que desenvolveram sistema autônomo de monitoramento utilizando visão computacional e sensores de temperatura e gás ligados a *bluetooth*, ou de Miguel et al. (2017) que utilizaram KNN e câmera para classificar queda em tempo real, ou até mesmo da empresa Tecnosenior, que oferece dispositivos como sensor de queda e pulseira de emergência para o público sênior (ROSA, 2022), observamos que a vulnerabilidade do público mais velho têm sido cada vez mais amparadas. Contudo, mesmo estas sendo soluções viáveis e já comerciais, ainda têm em comum uma vulnerabilidade primordial em seu funcionamento, pois necessitam que o usuário esteja em posse do objeto para que seja detectada a ocorrência do tombo.

Diante dessa realidade, define-se a seguinte questão de pesquisa: como contribuir para a detecção de quedas de idosos e notificação para um responsável? Tal questão ilustra o foco deste projeto. Visando contribuir para os avanços nessa área, visa-se avaliar modelos de aprendizagem de máquina como ferramenta de detecção de queda de idosos através de vídeos, para auxiliar no amparo de idosos vulneráveis, notificando alguém responsável por ele, possibilitando a intervenção mais rápida de atendimento hospitalar em caso de emergência confirmada. Um dos modelos mais populares e eficazes para detecção de objetos e eventos com base em *Deep Learning* é o YOLO (*You Only Look Once*), capaz de identificar múltiplos objetos em imagens ou vídeos (SOHAN et al., 2024).

Seguindo essa mesma linha de modelos da família YOLO, destacam-se dois modelos: YOLOv8, que será integrado à biblioteca de visão computacional OpenCV — amplamente utilizada para o processamento de imagens e vídeos. Esse processamento será vinculado à análise

de vídeos(*frames*) por meio de redes neurais convolucionais (CNNs), permitindo a detecção de possíveis quedas, com o objetivo de identificar e classificar os eventos registrados. E o YOLO-NAS, que adota uma abordagem semelhante, mas com arquiteturas otimizadas automaticamente por meio de NAS *(Neural Architecture Search*), mantendo a mesma base conceitual da linha YOLO, porém com foco em maior eficiência e adaptação a diferentes contextos computacionais(TERVEN et al., 2023).

2. Fundamentação Teórica

Nesta seção, apresentamos a teoria e os conceitos que fundamentam este projeto de pesquisa e elucidamos os termos técnicos que estão relacionados ao desenvolvimento do projeto, tais como: *Deep Learning*, Visão Computacional, e *frameworks* comumente utilizadas nela. Esses conceitos ajudam na compreensão do fluxo lógico que orienta a solução proposta.

2.1. Deep Learning

Deep Learning é definida por Patterson e Gibson (2017) como uma subcategoria específica do Aprendizado de Máquina, onde se concentram redes neurais profundas, caracterizadas por terem múltiplas camadas ocultas entre a entrada e a saída dos dados, permitindo modelagem mais complexas e abstratas dos dados hierarquicamente. É também boa em fazer previsões e generalizações a partir de dados que ainda não tenham sido vistos após um treinamento substancial. Consegue selecionar bem as melhores características sem intervenção humana. Um ótimo exemplo dessa aplicação é o uso de CNNs para detecção de tumores treinando a partir de 2892 imagens de do *dataset* BraTS 2020 de ressonância magnética (CHATTOPADHYAY; MAITRA, 2022)

2.1.1. Redes Neurais Convolucionais (CNNs)

As redes neurais convolucionais (*Convolutional Neural Networks*) são um tipo de rede neural que processa dados que têm estrutura em grade, tais como séries temporais (1D) ou imagens (2D). Elas utilizam a operação matemática chamada convolução, uma média ponderada onde valores recentes podem ter mais importância que valores antigos, em vez da multiplicação matricial tradicional usada em redes neurais densas. A convolução é uma operação linear que, no contexto de CNNs, é aplicada entre: Um *input* (entrada), como imagens, e um *kernel* (filtro), que é um conjunto de parâmetros aprendidos pela rede (GOODFELLOW et al., 2016).

2.2. Visão Computacional

A visão computacional é entendida como um campo que explora como sistemas computacionais podem ser projetados para interpretar e representar informações visuais extraídas de imagens, sendo elas estáticas ou sequenciais, buscando automatizar tarefas que o sentido visual humano pode fazer. Focando na modelagem computacional da percepção visual, abrangendo tarefas como aquisição, processamento, análise e compreensão estrutural e semântica de imagens. É capaz de gerar representações visuais que capturem a geometria, a textura e a profundidade do ambiente, permitindo que o sistema compreenda o mundo tridimensional a partir de projeções bidimensionais. Dessa forma, a visão computacional não se limita ao reconhecimento de objetos isolados, mas busca construir uma representação integrada e significativa da cena (SZELISKI, 2022).

2.2.1. You Only Look Once

É um modelo de detecção de objetos em tempo real criado por Redmon et al. (2015), e foi publicado em 2015 pela primeira vez. O modelo se baseia na simplicidade, onde aplica uma única rede neural diretamente nas imagens completas, e a partir daí divide as imagens em regiões simultaneamente, classificando essa imagem em uma única avaliação. A Figura 1 apresenta a arquitetura do *framework*, um modelo de detecção com 24 camadas convolucionais (extraem características da imagem, como bordas, texturas e etc), sequenciadas por 2 camadas totalmente conectadas (sintetizam todas as informações aprendidas pelas camadas convolucionais e produzem a previsão final). As camadas convolucionais alternadas de 1 x 1 vão reduzir o espaço da dimensionalidade da imagem de entrada, mantendo a essência das características mais importantes, e isso acelera o processo da detecção.

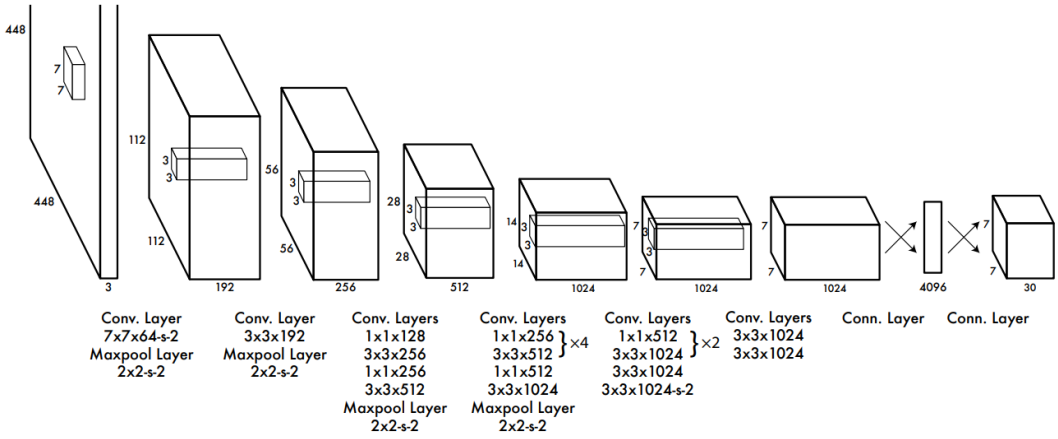

Figura 1. Arquitetura YOLO: mostra o esqueleto do modelo inicial (REDMON *et al*, 2016)

2.2.2. YOLOv8

O YOLOv8 é um modelo desenvolvido pela empresa *Ultralytics* em 2023, construído sobre uma CNN que executa o processo de detecção de objetos end-to-end, passando uma única vez a imagem (*You Only Look Once*). A CNN trabalha extraindo e refinando e interpretando as características visuais diretamente a partir da imagem original.

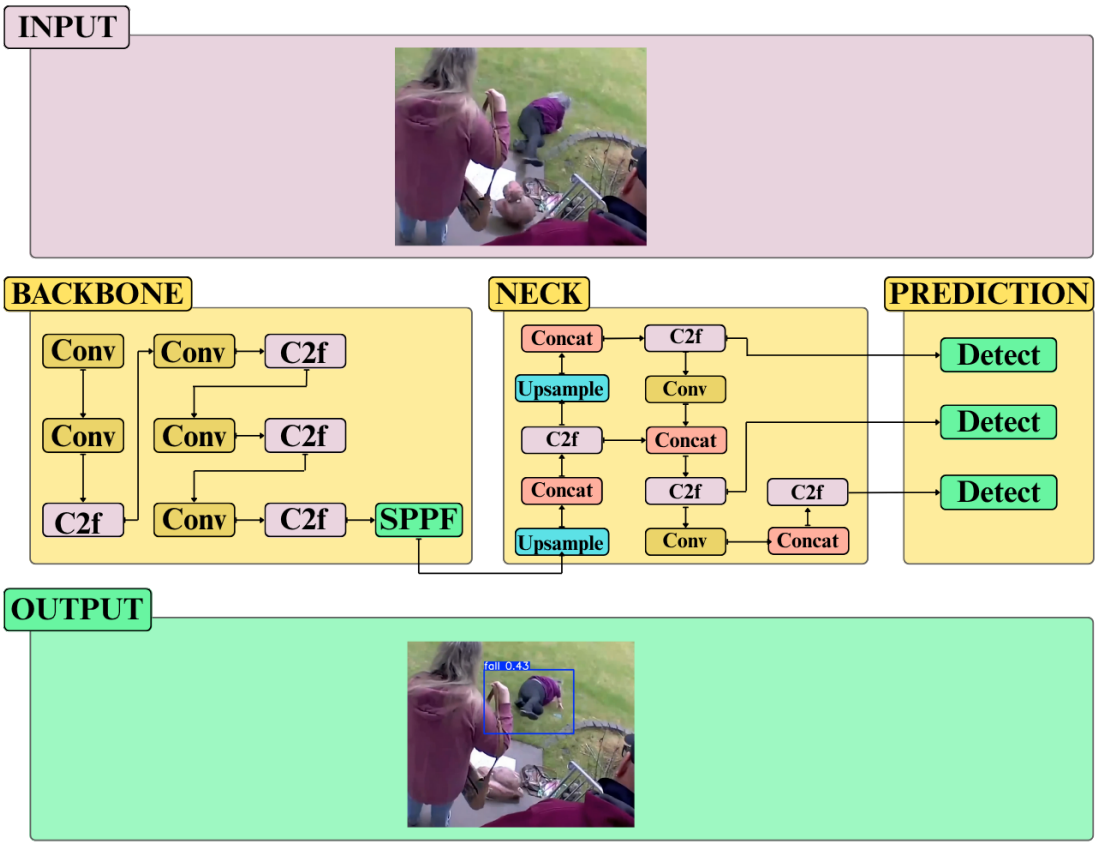

Figura 2. Arquitetura YOLOv8: Mostra o esqueleto do funcionamento do sistema. Autoria Própria, 2025.

O *backbone* é divido (ver Figura 2) em extração de características, *Cross Stage Partial Network*, diminuição de redundância computacional, mapas de características hierárquicas; *Neck*(fusão multiescala, que funde características extraídas em diferentes resoluções, melhorando detecção de objetos dentro da imagem; *Feature Pyramid Network + Path Aggregation Network; Prediction* (predição final onde a detecção acontece, trocando *anchor boxes* por *anchor free*, mais preciso, e prevê das caixas delimitadoras, rótulos e pontuações de confiança (YASEEN, 2024).

2.2.3. YOLO-NAS

O YOLO-NAS(*Neural Architecture Search*) é um modelo desenvolvido pela empresa Deci AI em 2023, inspirado nos modelos YOLOv8 e YOLOv6, voltado para detecção de pequenos objetos, com alta precisão de localização e eficiência computacional, sendo indicado para aplicações em dispositivos de borda em tempo real. Possui arquitetura de código aberto que está disponível para uso em pesquisas, e conta com módulos com reconhecimento de quantização, denominados QSP e QCI, que utilizam reparametrização para quantização em 8 bits, tentando minimizar a perda de acurácia durante o processo de quantização pós-treinamento. Foi construído automaticamente pela tecnologia *AutoNAC*, um sistema de busca neural adaptável a diferentes dados, ambientes e metas de desempenho**(Deci, 2023)**

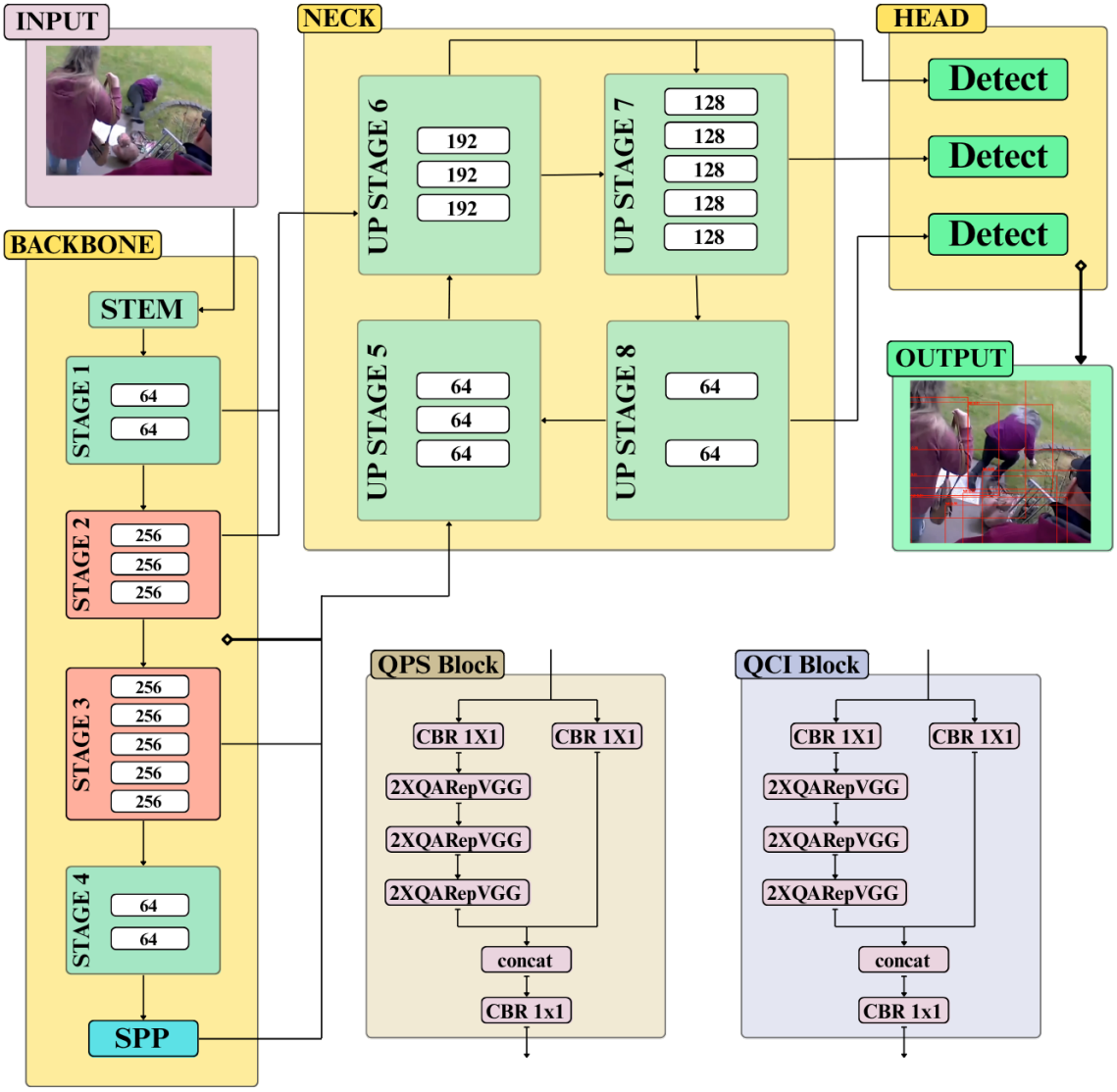

Figura 3. Diagrama de Sequência: descreve como os objetos interagem entre si em um sistema, em ordem cronológica. Autoria Própria, 2025.

Entre suas inovações, destacam-se os módulos QSP e QCI, projetados para quantização em 8 bits com mínima perda de acurácia, e o uso de quantização híbrida, que aplica a técnica seletivamente nas camadas do modelo. Durante o processo, foram integrados blocos RepVGG para compatibilidade com quantização pós-treinamento (PTQ). O modelo (ver Figura 3) passou por um pré-treinamento robusto, com dados rotulados automaticamente, *self-distillation* e grandes conjuntos de dados. Foram geradas três variantes: YOLO-NASS (*Small*), YOLO-NASM (*Medium*) e YOLO-NASL (*Large*), diferenciadas pela profundidade e pela disposição dos blocos especializados (TERVEN et al., 2023).

2.2.4. Ultralitics

O *Ultralitics* é uma biblioteca *open source* na linguagem *Python* voltada para visão computacional, que fica na execução de modelos YOLO de forma eficiente. Foi desenvolvida para suportar tarefas como detecção de objetos, classificação de imagens, segmentação de instâncias e esti-

mativas de pose, sendo uma biblioteca consolidada como de alta performance, facilidade de uso e compatibilidade com outros *frameworks* modernos - a exemplo do *PyTorch*. Dotada de arquitetura otimizada para funcionamento em tempo real e escalabilidade, é amplamente utilizada nos campos de pesquisa e indústria. Seguem uma estrutura unificada de *Backbone* (extração de características da imagem), *neck* (combinação de características multiescala) e *head* - predição final do *anchor free* mais precisa (JOCHER et al., 2023).

2.2.5. SuperGradient

A *SuperGradient* é um *framework* de código aberto para treinar e avaliar modelos de deep learnig, desenvolvido pela empresa DECI AI. Essencialmente voltada para visão computacional permitindo implementação de última geração, ficando na eficiência computacional, facilidade de reprodução de experimentos e otimização para implantar em produção. A arquitetura modular integra-se ao *PyTorch* e oferece suporte direto ao modelo YOLO-NAS e alguns outros. Arquitetura dividida em Modelos – implementação pré treinadas, *Recipes* - configurações de hiperparametros, e *Pipelines* – fluxos de trabalhos de treino, validação e exportação (AHARON et al., 2021).

2.3. Métricas de Precisão e Desempenho

Para avaliar a qualidade de modelos, são geralmente escolhidas algumas métricas para definir o desempenho do modelo a partir de diferentes aspectos. A iniciar pela Equação 1 - acurácia, que indica o percentual das previsões acertadas, Equação 2 - precisão, que calcula a proporção dos eventos que foram corretamente identificados como quedas em relação a todas as identificações de quedas, incluindo FP, Equação 3 - *F1-Score* é a média harmônica do *recall* e da precisão, que são inversamente proporcionais e informa a precisão do classificador e sua robustez, ou seja, quantas instâncias são classificadas de forma correta (THARWAT, 2021). Na Equação 4, AUC-ROC, onde AUC (Área sob a curva) indica a probabilidade de o modelo classificar corretamente um exemplo positivo acima de um negativo, escolhido de forma aleatória. e ROC (*Receiver Operating Characteristic*) é uma representação visual do desempenho de um modelo de classificação em diferentes limiares mostrando a relação entre a taxa de verdadeiros positivos (TP) e a taxa de falsos positivos (FP)(LI, 2024). E finalmente o *Log Loss* (Perda de Entropia Cruzada - Equação 5), métrica padrão usada para problemas de classificação, recomendado para modelos que produzem probabilidades como saída. Avalia não só o rótulo previsto, mas também a confiança do modelo, penaliza previsões incorretas com alta confiança, gera gradientes suaves (VOVK, 2015).

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{2}$$

$$F_1 = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{3}$$

$$\text{AUC-ROC} = \int_0^1 TPR(FPR)\, d(\text{FPR}) \tag{4}$$

$$\text{Log Loss} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)] \tag{5}$$

3. Trabalhos Relacionados

A pesquisa de Ximenes e Ziza (2022) aborda o desenvolvimento de um sistema residencial autônomo de monitoramento de acidentes domésticos de idosos, utilizando técnicas de visão computacional, sensores de temperatura e gás para detectar fogo e quedas, integrados a um microcontrolador com tecnologia *bluetooth* que envia as informações para outro dispositivo. Este segundo dispositivo possui uma câmera que faz a confirmação do acidente acontecido e notifica os responsáveis pelo idoso através de servidor web. Após a realização dos testes, o sistema constatou a precisão de 93,% para quedas reais e 100% para quedas falsas, entretanto, ainda devem ser superados alguns desafios, como a presença de animais de estimação nos cômodos e quando o idoso se deita intencionalmente. A visão computacional se mostrou eficaz e promissora para a detecção de quedas envolvendo pessoas idosas.

Em uma linha de trabalho similar, Miguel et al. (2017) propõem um sistema de detecção de quedas com visão computacional, baseado em câmera, onde a classificação de imagens é responsável por identificar, em tempo real, se o indivíduo monitorado sofreu uma queda. O sistema trabalha com variáveis extraídas da silhueta da pessoa detectada. As principais características utilizadas são o ângulo da elipse que envolve o corpo (inclinação), a razão entre altura e largura da figura envolvente (postura) e a derivada dessa razão (velocidade de mudança da postura ao longo do tempo). Essas variáveis são processadas por um filtro de *Kalman*, que reduz ruídos e oscilações naturais do corpo. Os dados são encaminhados para um classificador baseado em *K-Nearest Neighbors* (KNN), que compara as características atuais com um banco de dados previamente treinado. O algoritmo avalia os três vizinhos mais próximos no espaço de características para determinar se o evento corresponde a uma queda real. O treinamento foi com vídeos simulando quedas em diferentes direções, além de ações de não-queda, para garantir a robustez da classificação. Esse modelo permitiu alcançar um *recall* de 96%, uma alta taxa de detecção correta das quedas, mas ainda precisa de melhorias em situações de oclusão.

Seguindo a mesma temática dos trabalhos anteriores, o estudo de Zin et al. (2021) propõe um sistema de reconhecimento de ações em tempo real para pessoas idosas usando uma câmera estéreo de profundidade. O sistema trabalha com localização de pessoas por extração de regiões de interesse (ROI) a partir de mapas de disparidade UV, utilizando sequências de quadros de profundidade fornecidas pela câmera, dessa forma podem localizar e identificar ações realizadas pelos idosos. Os recursos espaciais-temporais de duas representações de ação

(aparência de movimento de profundidade (DMA) e histórico de movimento de profundidade (DMH) com um descritor de histograma de gradientes orientados (HOG)) são usados em combinação com recursos baseados em distância e fusionados com um método de arredondamento automático para o reconhecimento de ações em sequências contínuas de quadros longos. Os resultados experimentais demonstram que o sistema proposto pode detectar várias ações em tempo real com taxas de reconhecimento razoáveis, independentemente do comprimento das sequências de imagem.

Por fim, o estudo conduzido por Chen et al. (2022) propõe um método de detecção de queda de idosos em tempo real aprimorando a arquitetura do YOLOv5s e *TensorFlow*. Ao substituir a Convolução básica pelo módulo *Asymmetrical Convolutional Blocks* (ACB), os autores melhoraram a capacidade de extração de recurso na rede, fazendo com que a detecção da queda fosse mais eficaz. Também introduziram um modelo de atenção especial à estrutura residual da rede, o que melhorou a capacidade de localização a partir de concentração de localização de recursos. Ainda mencionando melhorias, na camada de recurso, os autores removeram as camadas de 76x76, focando nas de 19x19 e 38x38 para detecção de quedas. O ajuste detecta duas situações: queda e não-queda, o que reduz a complexidade. Os resultados mostraram melhorias de 3,5% na média de precisão, se comparado à YOLOv5s e *TensorFlow*, sugerindo que o aprimoramento é eficaz para solucionar o problema

4. Requisitos da Solução Proposta

Pautados nas pesquisas que destacam a importância de detecção rápida e incisiva na ocorrência de quedas de idosos, a subseção a seguir descreve o quadro geral do funcionamento dos sistemas projetados. A solução computacional visa oferecer uma resposta eficiente, comparando os modelos YOLOv8 e YOLO-NAS e notificando as quedas que forem detectadas em vídeo durante a execução dos modelos.

4.1. Solução Computacional

A solução desenvolvida é um sistema semi-automatizado de detecção de quedas e idosos em vídeos. Todas as etapas foram separadas em células, e após o treinamento, o sistema recebe vídeos e detecta a ocorrência da queda em tempo de inferência, *frame* por *frame*, identifica quais vídeos representam queda e os separa, para facilitar o envio de mensagens, simulando um sistema de notificação remota. A Figura 4 a seguir mostra um diagrama de fluxo da solução.

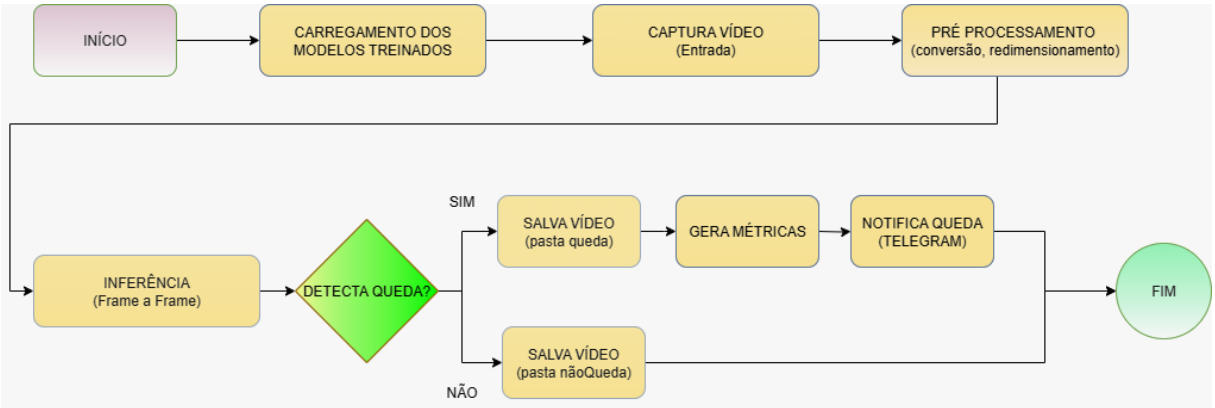

Figura 4. Diagrama de Fluxo: mostra as etapas, decisões e fluxos do projeto. Autoria Própria, 2025.

De acordo com o diagrama de fluxo descrito pela Figura 4, o retângulo rosa indica o início do funcionamento do sistema, os retângulos amarelos representam os fluxos dos proces-

sos, enquanto o verde se refere a etapa de funcionamento do sistema com tomada de decisão. O círculo verde indica o final da execução total. Os elementos são ligados por setas, que indicam relação de dependência, onde cada etapa obtém o resultado da etapa anterior (SOMMERVILLE, 2011).

Os modelos são treinados com o *dataset* autoral, e após finalizados, iniciam no carregamento dos modelos treinados com YOLOv8 e YOLO-NAS, segue para a alimentação de mídia, com vídeos disponibilizados para teste sem terem passado por nenhuma etapa do treinamento. Em seguida, a cada *frame* os vídeos são analisados, passando para o próximo passo, da detecção da queda do idoso. A partir da detecção, caso haja confirmação da queda, o salva o vídeo em uma pasta **queda**, gera métricas e notifica a queda. Caso não haja detecção de queda, o vídeo é apenas salvo em uma pasta **naoQueda**.

4.2. Descrição Geral dos Sistemas

Os sistemas propostos são soluções computacionais voltadas para detecção automática de quedas em vídeos utilizando CNNs, mais especificamente as arquiteturas YOLOv8 e YOLO-NAS. Ambos foram projetados como protótipos funcionais de um sistema inteligente de monitoramento assistivo para detectar quedas de idosos, capazes de classificar eventos em vídeos e enviá-los, junto com notificação textual de alerta automaticamente em caso de detecção de queda. Utiliza um *dataset* majoritariamente elaborado com vídeos públicos de idosos em situações cotidianas, que é exatamente como as quedas ocorrem. Aplicando Inteligência artificial para o monitoramento automatizado faz com que o sistema atue como suporte de segurança e possa ser usado para eventualmente melhorar o tempo de resposta aos incidentes de queda.

4.3. Requisitos

Para Sommerville (2011), o requisito funcional é definido pela necessidade de descrever as funções e serviços que um sistema deve oferecer, tratando do comportamento dele em cenários diversos e como responde às entradas. Já os requisitos não funcionais são referidos como a parte de como o sistema realiza suas funções, sendo restrições sobre os serviços dele, sendo críticos para o sucesso do sistema. Os requisitos demarcam o que o sistema deve fazer para ser funcional, cobrindo operações específicas, interação do sistema com o usuário e o processamento dos dados, definindo a capacidade esperada do sistema. A seguir estão descritos os requisitos Funcionais e Requisitos não funcionais dos sistemas em análise:

4.3.1. Requisitos Funcionais

ID do Requisito	RF001
Nome	Captura e leitura de vídeos de entrada
Descrição	Neste requisito, o sistema deve captar os vídeos armazenados localmente.
Prioridade	[X] ESSENCIAL

Tabela 1. Requisito funcional 1. Fonte: Autoria Própria, 2025.

ID do Requisito	RF002
Nome	Inferência de detecção de queda
Descrição	Os sistemas devem utilizar YOLOv8 e YOLO-NAS para processar o vídeo e realizar a detecção das classes *fall* e *person, frame* a *frame* nos vídeos.
Prioridade	[X] ESSENCIAL

Tabela 2. Requisito funcional 2. Fonte: Autoria Própria, 2025.

ID do Requisito	RF003
Nome	Classificação de vídeos
Descrição	O sistema deve classificar os vídeos como **queda** ou **naoQueda**, com base nas detecções realizadas.
Prioridade	[X] ESSENCIAL

Tabela 3. Requisito funcional 3. Fonte: Autoria Própria, 2025.

ID do Requisito	RF004
Nome	Organização automática dos vídeos
Descrição	Os vídeos processados devem ser automaticamente movidos para pastas separadas: **queda**, para vídeos com queda detectada e **naoQueda** para vídeos sem queda detectada.
Prioridade	[X] ESSENCIAL

Tabela 4. Requisito funcional 4. Fonte: Autoria Própria, 2025.

ID do Requisito	RF005
Nome	Notificação via Telegram
Descrição	Se uma queda for confirmada, o sistema deve enviar mensagem por meio do Telegram, com o vídeo correspondente à queda detectada, não excedendo 3 minutos entre a detecção e o recebimento da mensagem pelo destinatário.
Prioridade	[X] ESSENCIAL

Tabela 5. Requisito funcional 5. Fonte: Autoria Própria, 2025.

4.3.2. Requisitos Não Funcionais

Para garantir o desenvolvimento eficiente e a implementação adequada dos modelos YOLO propostos para detecção de quedas em idosos, foram estabelecidos requisitos não funcionais que orientam as características técnicas e operacionais do sistema. Estes requisitos, abrangem aspectos fundamentais como compatibilidade com o ambiente *Python* 3.10, portabilidade para sistemas *Windows* com suporte opcional a CUDA, otimização de desempenho através do uso prioritário de GPU quando disponível, e facilidade de uso por meio de notebooks organizados em células estruturadas.

ID do Requisito	RN001
Nome	Compatibilidade
Descrição	Os sistemas devem funcionar no ambiente virtual *Python* 3.10 com bibliotecas especificadas.
Prioridade	[X] ESSENCIAL

Tabela 6. Requisito não funcional 1. Fonte: Autoria Própria, 2025.

ID do Requisito	RN002
Nome	Portabilidade
Descrição	O sistema deve ser capaz de ser executado em qualquer máquina *Windows* (incluindo, no caso do NAS, o *WSL2*) com suporte CUDA (se disponível) e *Python* 3 e superiores.
Prioridade	[X] ESSENCIAL

Tabela 7. Requisito não funcional 2. Fonte: Autoria Própria, 2025.

ID do Requisito	RN003
Nome	Desempenho
Descrição	O uso da GPU (se disponível) deve ser priorizado para acelerar a inferência.
Prioridade	[X] ESSENCIAL

Tabela 8. Requisito não funcional 3. Fonte: Autoria Própria, 2025.

ID do Requisito	RN004
Nome	Usabilidade
Descrição	Os sistemas devem ser facilmente executáveis através dos *notebooks* com células organizadas.
Prioridade	[X] IMPORTANTE

Tabela 9. Requisito não funcional 4. Fonte: Autoria Própria, 2025.

ID do Requisito	RN005
Nome	Segurança
Descrição	Os dados de acesso ao Telegram devem ser armazenados de forma segura.
Prioridade	[X] IMPORTANTE

Tabela 10. Requisito não funcional 5. Fonte: Autoria Própria, 2025.

ID do Requisito	RN006
Nome	Modularidade
Descrição	O código deve ser modular, permitindo reutilização e facilidade de manutenção e inserção de trechos novos que não prejudique o todo, como *notebooks* divididos por etapas.
Prioridade	[X] IMPORTANTE

Tabela 11. Requisito não funcional 6. Fonte: Autoria Própria, 2025.

ID do Requisito	RN007
Nome	Desempenho
Descrição	O sistema deve ser capaz de processar vídeos curtos ($\tilde{1}$ min) em tempo razoável (não exceder 3 minutos), ou seja, com menos tempo de inferência.
Prioridade	[X] IMPORTANTE

Tabela 12. Requisito não funcional 7. Fonte: Autoria Própria, 2025.

4.4. Fluxo de Execução

Em um plano demonstrado pelo diagrama de sequência da figura a seguir, podemos visualizar o esquema de funcionamento proposto do sistema. Em um diagrama de sequência, os principais elementos representam objetos ou instâncias de classe que interagem entre si, criando um fluxo de comunicação, como nos objetos treinamento e captura de video, por exemplo, que interagem através das leitura de conteúdo da pasta. No caso apresentado, os objetos Captura de vídeo e pré-processamento, são entidades ativas, desempenhando funções muito específicas no sistema. As mensagens/chamadas trocadas entre esses objetos são representadas pelas setas, que indicam o extração de dados de vídeo com YOLO, por exemplo. Esse fluxo de sistema de decisão inclui lógica condicional, onde é avaliado o resultado proveniente do YOLO pela detecção de objeto e verificação de queda, e, caso haja confirmação da queda, aciona o bot do Telegram.

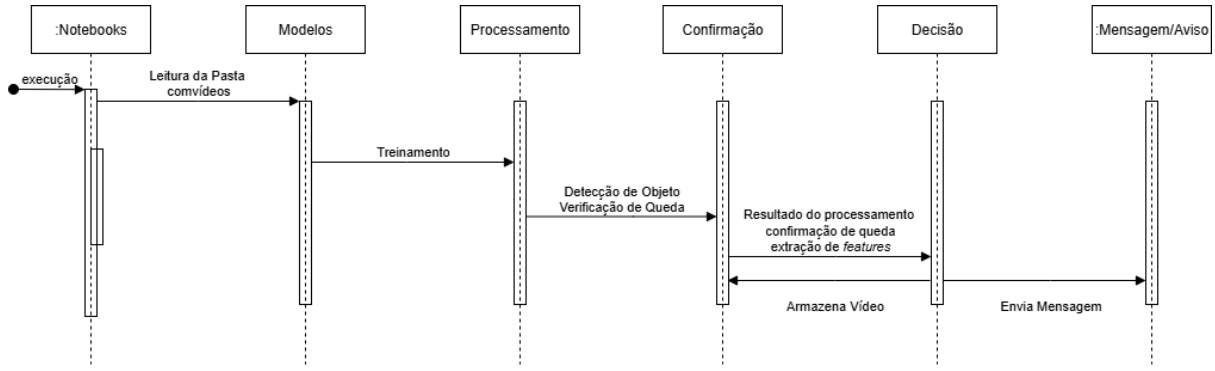

Figura 5. Diagrama de Sequência: descreve como os objetos interagem entre si em um sistema, em ordem cronológica. Fonte: Autoria Própria, 2025.

4.5. Realização do Experimento

Foram implementados dois modelos baseados na arquitetura YOLO: o YOLO-NAS e o YOLOv8. Para o YOLOv8, foi utilizado o modelo disponibilizado pelo Nanor (2024) em TechWatt,

enquanto para o YOLO-NAS, foi baseado o modelo desenvolvido por Stpeteishii (2023).O Segundo modelo dispôs originalmente de um conjunto de dados genérico *Fall Detection - Eldercare Robot, de Elwaly e Ibrahim (2023)*. Posteriormente, foi realizada a criação de um conjunto de dados próprio, adquiridos através de vídeos públicos de idosos em diversos cenário de **queda** e de **não-queda**, a fim de tornar os testes mais específicos e alinhados ao objetivo do projeto. Os testes foram realizados com os dois modelos, visando avaliar sua eficácia na detecção de quedas. No caso do YOLO-NAS, o pré-treinamento baseado no conjunto COCO foi removido, utilizando como base apenas biblioteca *SuperGradients*. Já para o YOLOv8, foi empregada a biblioteca *Ultralytics*. As análises de desempenho dos sistema seguiram métricas frequentemente calculadas para fornecer uma visão holística da performance do sistema, etapa crucial para entender o desempenho do modelo nas tarefas de classificação (THARWAT, 2021).

4.6. Coletas e Análises

Conforme diagramado na *Big Picture* (Figura 6), os sistemas coletam os vídeos de teste, e o passam pelo modelo de visão computacional, analisando cada *frame*. Analisam se o vídeo contém ou não uma queda de idoso. Caso não haja, apenas salvam o vídeo em uma pasta de não-queda, mas caso haja, extrai as *features* e manda o vídeo para uma pasta de queda, e o envia com mensagem de alerta através do *bot* do Telegram.

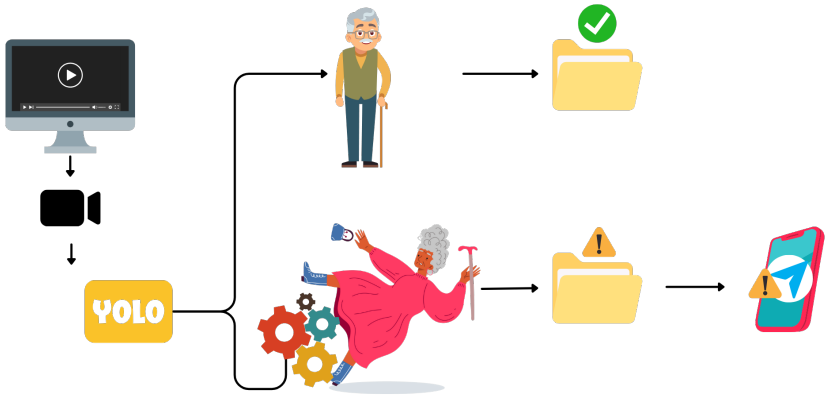

Figura 6. Big Picture: Visão geral de uma situação de queda no contexto do projeto. Fonte: Autoria Própria, 2025.

Para avaliar o funcionamento do sistema, foram replicados os modelos Fall Detection YOLO-NAS Train& Predict (STPETEISHII, 2023) e executado o *script Fall-Detection* (NANOR, 2024), com o YOLOv8. Após isso, ambos os modelos foram modificados para comportar o *dataset* autoral e a leitura de vídeos e saída de notificação condicional da queda. O modelo YOLOv8, mais simples, apenas com uma entrada de vídeo de detecção de queda para testar o funcionamento do modelo. A Análise do experimento prevê a entrada de vídeos públicos de ocorrências de quedas reais de idosos para representar de forma mais realista um ambiente doméstico, alguns sendo com possíveis obstáculos, poucos com imagens noturnas e utilizando métricas já estabelecidas.

4.7. Ambiente de Desenvolvimento e Implementação da Solução

A solução proposta tem dois sistemas adaptados para verificação de queda por vídeo. O projeto baseado no YOLO v8 foi executado em ambiente virtual do *Windows 11* com *Jupyter Notebook*. Já o YOLO-NAS precisou ser executado em ambiente virtual *Linux* com

UBUNTU, através do *WSL2*, utilizando ambiente virtual *Miniconda* 3 para ser executado com *Jupyter Notebook*. Ambos os projetos foram desenvolvidos com Linguagem *Python*. Todos os scripts desenvolvidos para viabilizar a execução dos projetos estão disponíveis em: `https://github.com/nataliaalmada/projects/tree/main/visionfall`.

5. Avaliação Experimental

Esta seção descreve o planejamento e resultados da execução da avaliação experimental da solução proposta, incluindo: o planejamento e projeto para a execução de um estudo experimental para avaliar o método proposto neste trabalho.

5.1. Criação do Dataset

A construção do *dataset* autoral foi uma etapa fundamental para o desenvolvimento do sistema de detecção de quedas de pessoas idosas com os modelos YOLO. Este processo envolveu diversas fases, que exigiram cuidado com a curadoria, organização e anotação dos dados.

A. Coleta dos Vídeos: A base de dados foi formada a partir de 166 vídeos próprios e públicos, coletados do *YouTube*, contendo 113 vídeos de situações reais de quedas de idosos e 53 vídeos de idosos realizando atividades comuns como realizando exercício físico, caminhando, sentando, levantando ou permanecendo parados, com apenas 1 vídeo de imagens noturnas, e o restante com imagens coloridas. Os vídeos foram organizados em cenas com duração curta, de 2 a 6 segundos, capturando momentos antes, durante e após a queda. Foram separados 10 vídeos de queda para não entrarem em nenhuma das etapas de treinamento, com apenas 1 vídeo de imagens noturnas, e o restante com imagens coloridas.
B. Padronização de FPS: Todos os vídeos tiveram seus quadros por segundo padronizados para 30, visando uma quantidade razoável de quadros para buscar sempre o quadro com melhor qualidade para ser incluído no *dataset* e evitar *frames* com poses muito semelhantes.
C. Extração de *Frames*: para transformar os vídeos em imagens anotáveis, foi realizada a extração de *frames* a uma taxa de amostragem de 30 fps, adequada para manter a continuidade do movimento. Isso garantiu que o *dataset* capturasse: A fase anterior à queda; O momento exato da queda; O estado posterior ao impacto.
D. Anotação Manual com *LabelImg*: As imagens extraídas foram rotuladas manualmente com a ferramenta *LabelImg*, respeitando o formato YOLO: Cada imagem recebeu uma ou mais caixas delimitadoras (bounding *boxes*); As classes utilizadas foram: 0: *queda*, anotado apenas no momento ou pós impacto e 1: *person* anotado na maioria dos *frames* onde houvesse pessoa visível.
E. Organização do *Dataset*: Após a anotação, as imagens e arquivos .txt com as *labels* foram divididos nas pastas `train/`, de treinamento, `val/` de validação, `test/` de teste, com uma divisão aleatória de proporção próxima de 70% (treinamento), 20% (validação) e 10% (teste), garantindo que os conjuntos fossem equilibrados em termos de vídeos com e sem quedas. Totalizando assim 452 *frames* de treinamento, 129 *frames* de validação e 66 de testes, para isso, foi usado um *script* em *Python*.
F. Preparação para o YOLOv8 e YOLO- NAS: Por fim, o arquivo `data.yaml` foi gerado com a estrutura exigida pelas bibliotecas *Ultralytics* e *SuperGradients*, indicando os caminhos para os diretórios e as classes do problema.

5.2. Projeto da Avaliação Experimental

Os experimentos foram conduzidos em um computador *Intel*(R) *Core*(TM) i5-14600K (3.50 GHz), 32 GB RAM, com *Windows 11 Pro* 24H2 x64. Placa de vídeo NVIDIA RTX 4070 TI de 12 GB VRAM com 7680 CUDA cores.

A solução proposta são dois sistemas adaptados para verificação de queda por vídeo. O projeto baseado no YOLO v8 foi executado em ambiente virtual do *Windows* com *Jupyter Notebook*. Já o YOLO-NAS precisou ser executado em<WSL2(GNU/Linux6.6.87.2-microsoft -standard-WSL2x86_64)UBUNTU(24.04.2LTS)>, utilizando ambiente virtual *Miniconda* para ser executado com *Jupyter Notebook*. Ambos os projetos foram desenvolvidos com Linguagem *Python*(3.10.11). O desenvolvimento dos sistemas foi estruturado nas seguintes etapas:

- **Etapa 1:** Execução dos sistemas já existentes (YOLO-NAS e YOLOv8), com pequenas correções relacionadas às versões de bibliotecas e dependências para basear os modelos propostos.
- **Etapa 2:** Curadoria de vídeos de idosos em situações de queda e em outras atividades, obtidos a partir de fontes públicas como o *YouTube*.
- **Etapa 3:** Construção de um *dataset* próprio a partir dos vídeos selecionados na etapa anterior.
- **Etapa 4:** Modificação dos sistemas de detecção para adaptar a entrada ao novo *dataset* autoral.
- **Etapa 5:** Treinamento dos modelos com base no novo conjunto de dados, ajustados para 50 épocas, com o modelo YOLO-NAS de confiança 0.1 na validação, enquanto YOLOv8 manteve-se em 0.4.
- **Etapa 6:** Adição de funcionalidades para leitura e exportação de vídeos processados.
- **Etapa 7:** Integração com a API do Telegram para envio automático de mensagens de alerta, juntamente com o vídeo correspondente à queda detectada.

5.3. Análise dos Resultados

A análise comparativa entre os modelos YOLO-NAS e YOLOv8 que foram adaptados ao *dataset* contendo exclusivamente idosos em contexto de **queda** e **não-queda**, com base nos gráficos de métricas extraídos durante o treinamento e avaliação. As comparações são feitas por métricas equivalentes, permitindo observar diferenças de comportamento e desempenho entre as arquiteturas.YOLOv8 representa o modelo original, assim como o YOLO-NAS. Já YOLOv8-A e YOLO-NAS-A representam os modelos propostos.

Métrica	YOLOv8	YOLOv8-A	YOLO-NAS	YOLO-NAS-A
Acurácia	1.0	0.977	0.841	0.387
Precisão	1.0	1.0	0.960	1.0
F1-Score	1.0	1.0	0.859	0.048
AUC-ROC	1.0	1.0	0.883	0.798
Log Loss	0.325	0.026	-	0.897

Tabela 13. Comparativo de métricas dos modelos YOLO. Fonte: Autoria própria, 2025.

O YOLOv8 apresenta valores significativos para Acurácia, precisão, *F1-Score* e AUC-ROC, que são indicativos de desempenho ideal na detecção correta e capacidade de evitar falsos positivos e negativos, o *Log Loss* baixo reforça a segurança nas previsões do modelo. Entretanto, esse desempenho também deve ser observado em caso de *overfitting*, pois a base de teste não foi suficientemente diversa, com apenas 1 vídeo de teste. Já a versão adaptada mantém as métricas

altas, quase se equiparando a modelo puro, divergindo em acurácia, de 0.97 e *Log Loss* também baixo, indicando previsões corretas. Essa queda de Acurácia sugere alguns erros em relação ao modelo puro, mas se mantém com bom desempenho.

Já o modelo YOLO-NAS apresenta métricas relativamente boas, se tratando de um modelo alternativo. Valores de Precisão e AUC-ROC indicam uma boa capacidade de classificação correta dos positivos. o *F1-Score* parece equilibrado (precisão e *recall*), mas a acurácia e o *Log Loss* mais altos deixam a desejar, pois sugerem previsões menos confiantes. O modelo Adaptado apresenta uma precisão boa, evita falsos positivos, mas o *F1-Score* é quase nulo, indicando severa dificuldade ao detectar corretamente os positivos. AUC-ROC de 0.798 mostra pouca discriminação, a o *Log Loss* altíssimo reforça que o modelo erra com alta confiança em casos incorretos. Em seguida, uma análise mais detalhada de gráficos gerados após a execução dos modelos YOLOv8-A e YOLO-NAS-A:

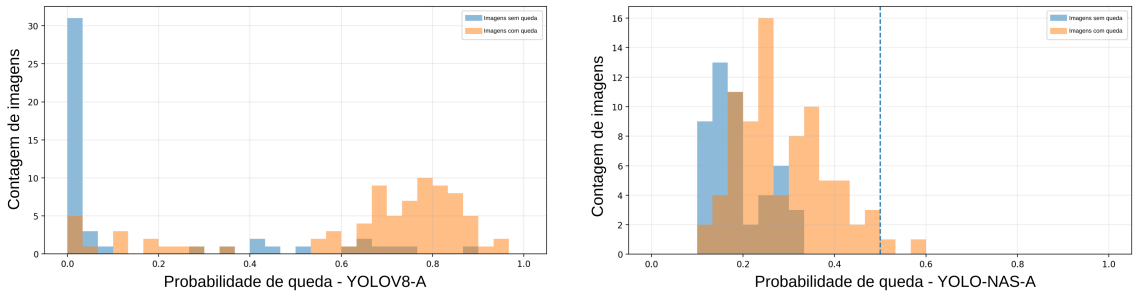

Figura 7. Gráfico de Probabilidade de Queda: distribuição da probabilidade de queda predita pelo modelo, separada entre imagens com queda e sem queda. A separação entre as curvas indica a capacidade do modelo em diferenciar os casos. A linha tracejada representa o limiar de decisão usado para classificar as imagens. Fonte: Autoria própria, 2025.

A (Figura 7) exibe a distribuição da probabilidade dada pelo modelo para a classe **queda**, para o YOLOv8-A, as probabilidades de classe estão bem separadas, poucas sobreposições, os **quedas** tendem para acima de 0.6, e **não quedas** próximos de 0, o que facilita a definição dos limiares de decisão mais assertivos. Já o do YOLO-NAS-A, o **queda** e o **não queda** são consideravelmente sobrepostos, o que dificulta uma separação clara entre classes, aumentando também as chances de falsos positivos e negativos. A separação mais nítida do YOLOv8-A reduz as ambiguidades durante a classificação.

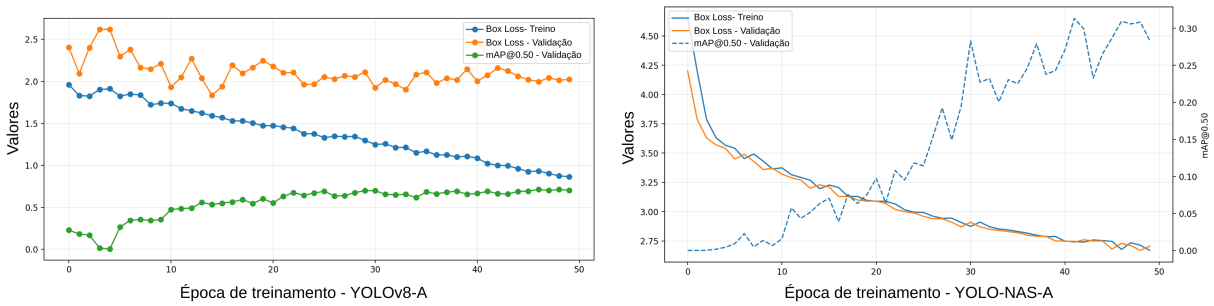

Figura 8. Gráfico de Evolução por Época (*Loss* e mAP): apresentam a evolução do erro (*loss*) no treino e validação, junto com a métrica mAP@0.50 ao longo das épocas. Fonte: Autoria própria, 2025.

É possível averiguar *Mean Average Precision* (mAP), ou média da precisão em diferentes níveis de confiança, o IoU sendo a a probabilidade atribuída pelo modelo de que a detecção

realmente corresponde à classe **queda**, avaliando se o modelo consegue detectar o objeto corretamente (IoU $\geq 50\%$) através da (Figura 8) que para o YOLO-NAS-A, o gráfico mostra uma queda bastante consistente a cada Época, sem sinais claros de *overfitting*, o mAP@0.50 parte de valores próximos a 0, e apresenta crescimento gradual. Mas mesmo com essa estabilidade no aprendizado, o desempenho na detecção ainda se mostra muito limitado. O *Box Loss* mede o erro na predição das caixas delimitadoras *(bounding boxes)*, e no YOLOv8-A, o *Box Loss* de Treinamento apresenta queda progressiva, enquanto o *Box Loss* Validação mantém oscilações, sugerindo maior dificuldade de generalização, e ainda assim, o mAP@0.50 cresce de forma rápida e atinge valores próximos de 0.7, mais que o dobro do valor obtido pelo YOLO-NAS-A. Enquanto o o NAS-A tem perda estável que não se traduz proporcionalmente em maior precisão, o v8-A mesmo sendo menos estável em perda, consegue extrair representações mais eficazes para detecção.

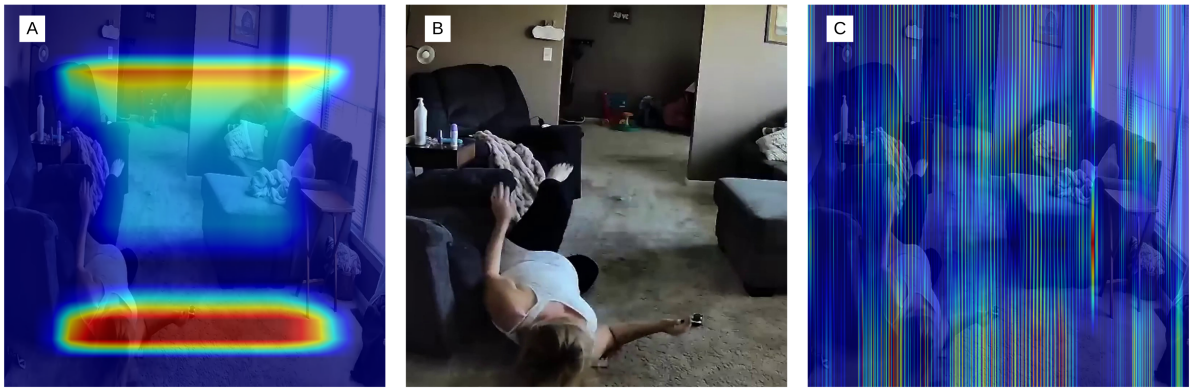

Figura 9. Grad-CAM: mapa de calor destaca as regiões da imagem que mais contribuíram para a decisão do modelo.as regiões mais intensas (vermelho) representam maior contribuição para a classificação, fornecendo interpretabilidade ao modelo. Em A imagem pelo modelo YOLO-NAS-A, e em B a imagem Original e C imagem pelo YOLOv8-A. Fonte: Autoria própria, 2025.

O mapa de ativação exibido na (Figura 9) do YOLO-NAS-A foca em duas faixas horizontais na imagem, com concentração na área inferior, onde há de fato uma pessoa caída, mas também inclui uma grande área irrelevante na área superior, o que pode acarretar em prejuízo na decisão final. Em contraponto, o mapa do YOLOv8-A é mais difuso, com detecções horizontais estreitas, que se concentram no idoso caído no chão, relevante para o evento da queda ser detectado. O YOLOv8-A demonstra maior foco na orientação do objeto de interesse, enquanto o YOLO-NAS-A dispersa parte da atenção.

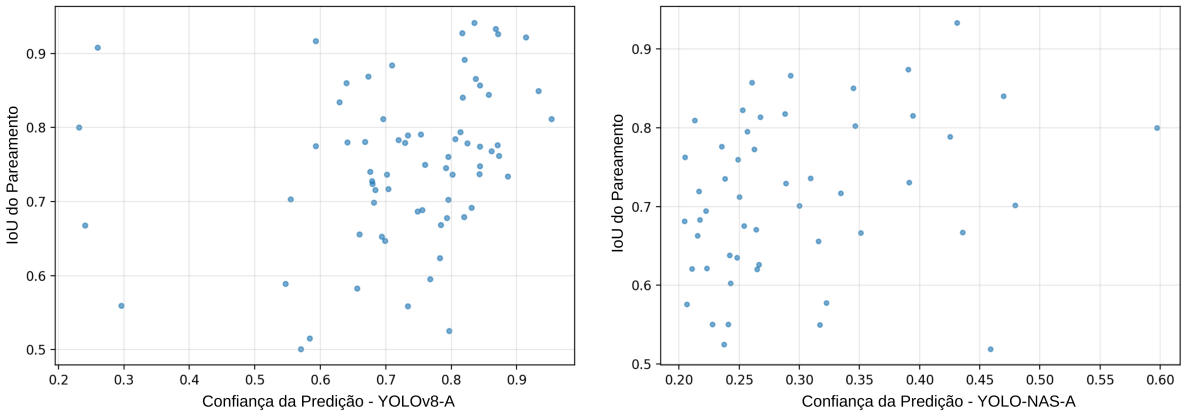

Figura 10. Gráfico de Dispersão (Confiança × IoU): relação entre a confiança das predições do modelo e a qualidade do ajuste (IoU) em relação ao ground truth. Cada ponto representa uma detecção. Fonte: Autoria própria, 2025.

O grafico de dispersão da (Figura 10) se refere à a probabilidade atribuída pelo modelo de que a detecção realmente corresponde à classe **queda**, e mostra padrão de concentração dos pontos entre 0,2 e 0,35 de confiança, com IoUs variando de moderados a altos, mostra que o modelo YOLO-NAS-A, mesmo ao acertar a localização (IoU elevado), tende a atribuir baixa confiança às predições, o que sugere problemas de calibragem ou indecisão na classificação, subestimando predições. Para o YOLOv8-A a distribuição se mostra significativamente melhor: grande parte das predições apresenta confiança acima de 0,7 e IoUs elevados (>0,7), ou seja, indica que o modelo localiza com precisão e demonstra segurança nas suas predições.

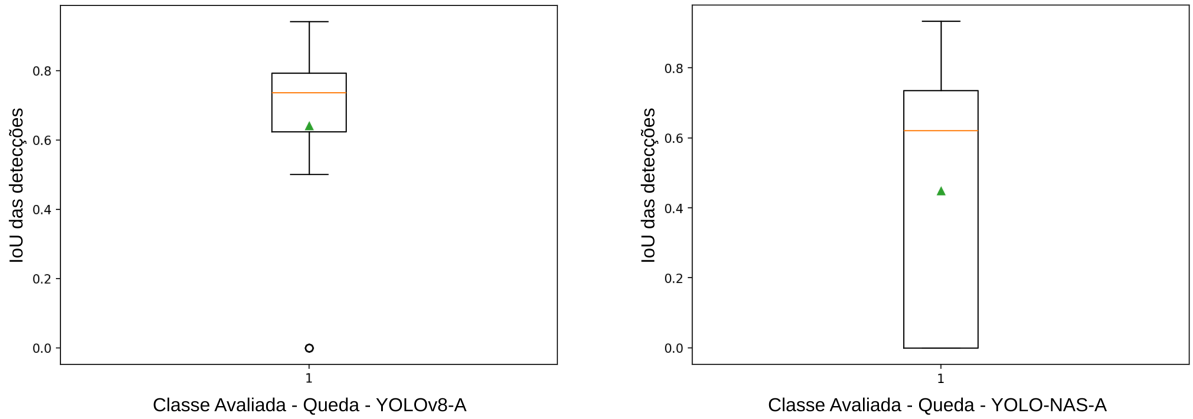

Figura 11. Gráfico Boxplot da Distribuição do IoU: distribuição do IoU das predições por classe. A mediana e os quartis indicam a qualidade típica das detecções, enquanto os pontos fora da caixa representam *outliers* (erros ou predições discrepantes). Fonte: Autoria própria, 2025.

Conforme podemos observar em (Figura 11), a mediana de IoU do YOLO-NAS-A fica em torno de 0,62, com ampla dispersão e presença de *outliers* significativos, o que sugere desempenho inconsistente, ou seja, algumas detecções são muito boas, mas outras falham de forma mais grave. Para o YOLOv8-A, a mediana próxima de 0,75, com intervalo interquartil menor e poucos *outliers*. Uma distribuição que indica que o modelo mantém qualidade mais uniforme nas predições. Ela também apresenta menor variabilidade, se tornando mais confiável em diferentes cenários.

5.4. Envio de Notificação

Ambos os modelos propostos conseguiram implementar com sucesso a função de envio de mensagem via Telegram para notificar um número de um responsável pelo idoso. Tal função tem importância significativa na evolução da notificação de acidentes de pessoas idosas e, consequentemente, na rapidez para a solução de possíveis problemas de saúde relacionados à queda.

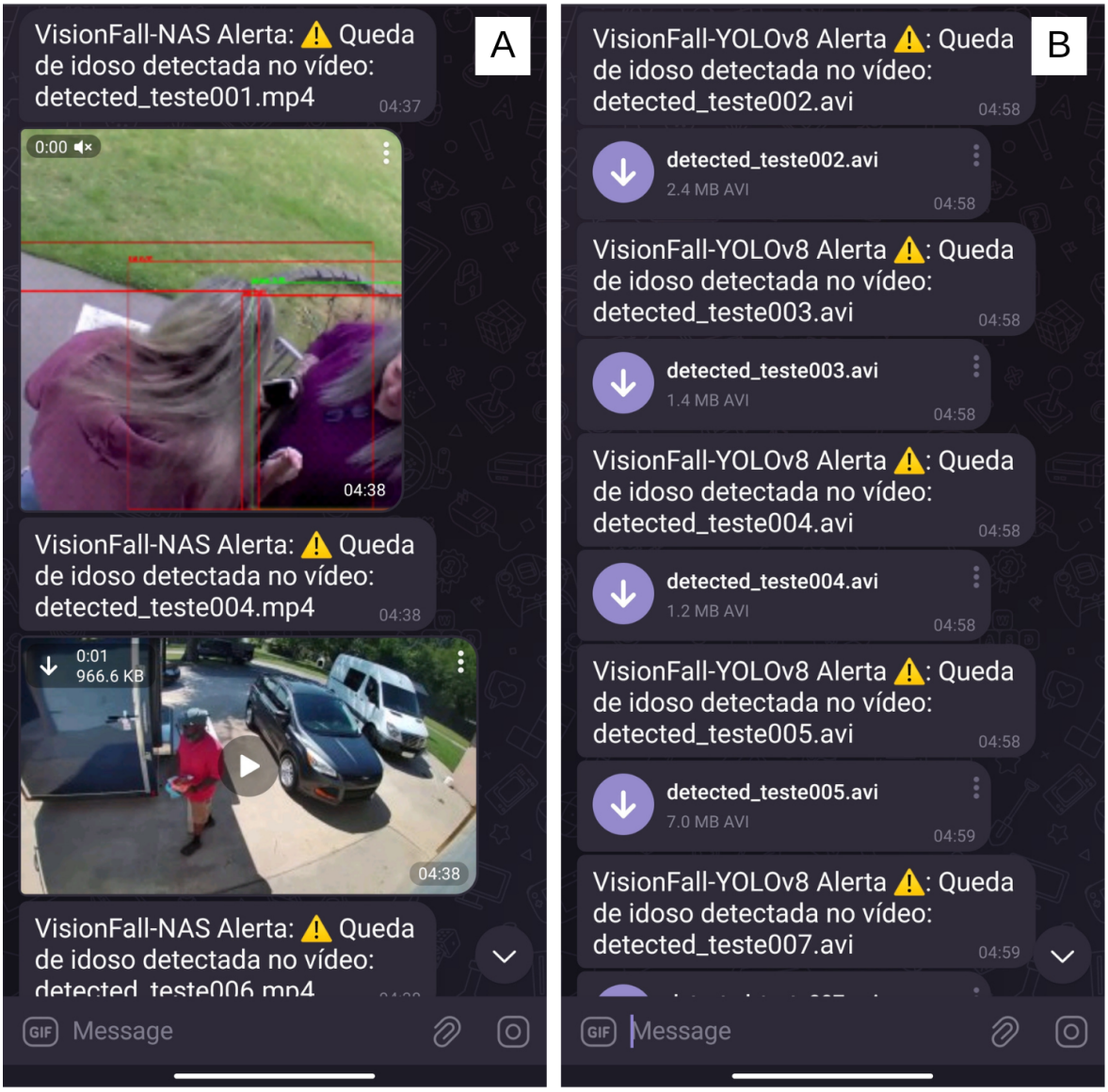

Figura 12. Captura de Tela de Notificações do Telegram. Mostra as mensagens de aviso juntamente ao envio dos vídeos onde foram detectadas as ocorrências de queda, em "A"temos as mensagens enviadas pelo modelo YOLO-NAS-A, e em "B"temos YOLOv8-A. Fonte: Autoria própria, 2025.

Na Figura 12, podemos observar a mensagem de ambos os modelos sendo recebidas, com uma notificação de alerta e o vídeo com a detecção em questão, para que o responsável averígue a veracidade da queda e possa assim tomar atitudes resolutivas, possibilitando evitar problemas mais graves. O modelo YOLO-NAS-A transfere o vídeo no formato `.mp4`, enquanto o YOLOv8-A envia no formato `.avi`, o que pode acarretar alguma dificuldade com a incompatibilidade de arquivo, a depender do dispositivo que recebe os vídeos.

6. Considerações Finais

O objetivo deste estudo foi utilizar ferramentas visão computacional para auxiliar na detecção de quedas de pessoas idosas, e assim buscar reduzir os problemas consequentes desses acidentes a essa população vulnerável. Com isso, utilizando modelos de detecção já funcionais, e os adaptando e treinando com um *dataset* específico de pessoas idosas, e adicionando um *bot* do Telegram para realizar notificação dessa queda, assim, buscamos atingir esse objetivo.

Verificamos que, ao realizar o treinamento com os dois modelos (YOLO-NAS-A e YOLOv8-A), ambos mostram padrões que se aproximam, em parte, do que a literatura reporta, mas também apresentam discrepâncias importantes, um deles se mostrou muito mais promissor para a detecção das quedas dos idosos, embora ainda com alguns ajustes a serem realizados. Em análise comparativa, o YOLOv8 se mostrou excepcional se mostrou tanto em métricas quantitativas quanto nas análises visuais de treinamento e predições, em relação ao modelo YOLO-NAS-A, atingindo valores máximos e algumas métricas como Precisão e *F1-score*, mantendo uma perda logarítmica relativamente baixa. As análises gráficas reforçaram sua consistência.

No caso do YOLOv8-A, a coerência entre as saídas visuais (mapas de ativação concentrados na pessoa caída) é bastante clara, a separação nítida das distribuições de probabilidade de queda. Esse conjunto de características evidencia que o modelo aprendeu bem a semântica da tarefa, sendo apto a localizar e classificar quedas com estabilidade. Tal performance se aproxima das melhores soluções relatadas nos artigos, como o YOLOv5s melhorado(Fulano), que alcançou mAP de 97,2% no URFD. A diferença está no fato de que, enquanto os artigos dificilmente superam 96–97% de F1-score ou acurácia, onde o YOLOv8-A chegou a valores próximos de perfeitos, o que sugere ou um conjunto de dados menos desafiador, ou até indícios de *overfitting* no protocolo de validação. Ainda assim, qualitativamente, o comportamento do YOLOv8-A está alinhado com os métodos de ponta que utilizam redes YOLO para detecção de quedas.

O YOLO-NAS-A, por outro lado, mostrou limitações severas na forma como foi adaptado. Os mapas de ativação (Figura 9) apresentam grande área de concentração fora de onde há queda, o boxplot (Figura 11) apresenta grande variância e valores baixos, e a dispersão (Figura 10) evidencia má calibração: as predições são incertas e pouco relacionadas com a qualidade da detecção. Os histogramas (Figura 8) reforçam esse cenário, já que há forte sobreposição entre classes, tornando a decisão binária extremamente instável. Esse resultado é inferior não apenas às abordagens baseadas em YOLO vistas na literatura, mas também a métodos mais antigos, como o sistema mais antigo implementado em *Raspberry Pi* de (Miguel, 2017), que já atingia acurácia de aproximadamente 96%. A performance do YOLO-NAS-A se assemelha mais aos cenários mais difíceis relatados no estudo com câmeras de profundidade em asilos (Zin 2021), nos quais a acurácia caía para 83% em sequências complexas, mas no YOLO-NAS-A é ainda mais crítico, com *F1-score* muito baixo e incapacidade de generalização.

Em um resultado imediato, o YOLOv8-A se tornou o modelo mais viável para desenvolver esse modelo detector de quedas de idosos, mas ainda requer alguns ajustes, confirmando tendência apontada de que as arquiteturas YOLO, quando bem ajustadas, podem superar métodos clássicos de visão e até sensores, alcançando níveis altos de precisão e *recall*, enquanto o YOLO-NAS-A destoa drasticamente em desempenho, até em relação às soluções mais antigas.

Ainda que o YOLO-v8-A tenha alcançado métricas quase perfeitas, algo que é incomum na literatura, o comportamento idica que é necessário investigar a confuguração experimental antes de afirmar uma superioridade absoluta. Para isso, alguns fatores devem ser considerados,

como aumentar a robustez do *dataset*, para treinar o modelo com dados mais balanceados e em maior quantidade, mantendo ainda a restrição de idade para ele. Para o YOLO-NAS-A, é possível que um treinamento com mais dados seja uma solução viável para a melhoria do modelo, assim como um melhor ajuste de hiperparâmetros, como das taxas de aprendizado.

Ainda, podem ser consideradas as combinações de arquiteturas, fazendo modelos híbridos que captem as melhores características dos modelos e as una, fazendo com isso uma ferramenta ainda mais eficaz, que seja capaz de identificar os idosos através das câmeras pelas de feições, postura e até caminhada. Sugerimos que estudos futuros avancem em relação a essa pesquisa buscando lidar com algumas limitações apresentadas, tais como avaliações em cenários reais, para verificar assim a qualidade do modelo, bem como o envio de mensagem automática e instantânea, que não dependa da execução do *Notebook* para ser enviada.

Referências

AHARON, S. et al. *Super-Gradients*. GitHub, 2021. Disponível em: <https://zenodo.org/record/7789328>.

ANDRADE, V. S.; PEREIRA, L. S. Influência da tecnologia assistiva no desempenho funcional e na qualidade de vida de idosos comunitários frágeis: uma revisão bibliográfica. *Revista Brasileira Geriatria Gerontologia*, 2009.

CHATTOPADHYAY, A.; MAITRA, M. Mri-based brain tumour image detection using cnn based deep learning method. *Neuroscience Informatics*, v. 2, n. 4, p. 100060, 2022. ISSN 2772-5286. Disponível em: <https://www.sciencedirect.com/science/article/pii/S27725286220 0022X>.

CHEN, T.; DING, Z.; LI, B. Elderly fall detection based on improved yolov5s network. *IEEE Access*, 2022.

DATASUS. *Filtro de Mortalidade - Brasil: Óbitos p/Residência por Faixa Etária segundo Local ocorrência*. 2024. Acesso em 25 de Maio de 2024. Disponível em: <http://tabnet.datasus.gov.br/cgi/tabcgi.exe?sim/cnv/obt10uf.def>.

ELWALY, A.; IBRAHIM, A. A. H. *Fall Detection - Eldercare Robot*. 2023. Dataset, versão publicada em 26 fev. 2023. Acessado em 13 jul. 2023. Disponível em: <https://www.kaggle.com/datasets/elwalyahmad/fall-detection>.

GOMES, I.; BRITTO, V. *Censo 2022: número de pessoas com 65 anos ou mais de idade cresceu 57,4% em 12 anos*. 2023. Acesso em 29 de Maio de 2024. Disponível em: <https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/38186-censo-2022-numero-de-pessoas-com-65-anos-ou-mais-de-idade-cresceu-57-4-em-12-anos>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

JOCHER, G.; QIU, J.; CHAURASIA, A. *Ultralytics YOLO*. 2023. <https://github.com/ultralytics/ultralytics>. If you use this software, please cite it using the metadata from this file. Disponível em: <https://ultralytics.com>.

LI, J. Area under the roc curve has the most consistent evaluation for binary classification. *PLOS ONE*, Public Library of Science, v. 19, n. 12, p. 1–28, 12 2024. Disponível em: <https://doi.org/10.1371/journal.pone.0316019>.

MIGUEL, K. D. et al. Home camera-based fall detection system for the elderly. *Sensors*, v. 17, n. 12, 2017. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/17/12/2864>.

NANOR, F. S. *Tech Watt – Fall Detection*. 2024. <https://github.com/Tech-Watt/Fall-Detection/tree/main>. Acesso em: 09 maio 2025.

ONU. *World Population Ageing 2023: Challenges and opportunities of population ageing in the least developed countries*. Department of Economic and Social Affairs, 2023. 74 p. Acesso em 29 de Maio de 2024. Disponível em: <https://desapublications.un.org/publications/world-population-ageing-2023-challenges-and-opportunities-population-ageing-least>.

PATTERSON, J.; GIBSON, A. *Deep Learning: A Practitioner's Approach*. 1. ed. [S.l.]: O'Reilly Media, 2017.

REDMON, J. et al. You only look once: Unified, real-time object detection. *arXiv*, 2015.

ROSA, B. *Empresa lucra com sensor de queda e chamada de emergência para terceira idade*. 2022. Acesso em 18 de Outubro de 2022. Disponível em: <https://oglobo.globo.com/blogs/pense-grande/post/2022/10/empresa-lucra-com-sensor-de-queda-e-chamada-de-emergencia-para-terceira-idade.ghtml>.

SOHAN, M.; RAM, T. S.; REDDY, C. V. A review on yolov8 and its advancements. In: *Data Intelligence and Cognitive Informatics*. [S.l.: s.n.], 2024.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. São Paulo, SP: Pearson Prentice Hall, 2011.

STPETEISHII. *Fall Detection YOLO-NAS Train & Predict (v2)*. 2023. <https://www.kaggle.com/code/stpeteishii/fall-detection-yolo-nas-train-predict>. Acesso em: 9 maio 2025.

SZELISKI, R. *Computer Vision: Algorithms and Applications*. Springer, 2022. Disponível em: <http://cv2.csie.ntu.edu.tw/CV2/2023/textbook.pdf>.

TERVEN, J.; CóRDOVA-ESPARZA, D.-M.; ROMERO-GONZáLEZ, J.-A. A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, MDPI AG, v. 5, n. 4, p. 1680–1716, nov. 2023. ISSN 2504-4990. Disponível em: <http://dx.doi.org/10.3390/make5040083>.

THARWAT, A. Applied computing and informatics. 2021.

VOVK, V. *The fundamental nature of the log loss function*. 2015. Disponível em: <https://arxiv.org/abs/1502.06254>.

World Intellectual Property Organization. *WIPO Technology Trends 2021: Assistive Technology*. 2021. Acesso em 29 de Maio de 2024. Disponível em: <https://www.wipo.int/publications/en/details.jsp?id=4541&plang=EN>.

XIMENES, T.; ZIZA, L. Desenvolvimento de um sistema para o monitoramento de acidentes domésticos de idosos utilizando visão computacional. *Revista Interdisciplinar de Pesquisa em Engenharia*, v. 8, n. 1, p. 36–47, 2022. Disponível em: <https://periodicos.unb.br/index.php/ripe/article/view/41819>.

YASEEN, M. *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*. 2024. Disponível em: <https://arxiv.org/abs/2408.15857>.

ZIN, T. T. et al. Real-time action recognition system for elderly people using stereo depth camera. *Sensors*, 2021.

ANEXOS

ANEXO 01 - TERMO DE RESPONSABILIDADE DO DISCENTE

Anexos são recursos que outro pesquisador produziu e que você usou no seu trabalho. Todos os alunos possuem, necessariamente, 03 anexos.

ANEXO 02 - DECLARAÇÃO DE AUTORIA

DECLARAÇÃO DE AUTORIA

Eu, **Natália Ribeiro de Almada** (código de matricula **2017009364**), autor da monografia/TCC (Trabalho de Conclusão de Curso) sob o título **Detecção Inteligente de Quedas em Idosos: Uma Abordagem Comparativa com Modelos YOLO**, declaro que o trabalho em referência é de minha total autoria e de minha inteira responsabilidade o texto apresentado. Declaro, ainda, que as citações e paráfrases dos autores estão indicadas com as respectivas obras e anos de publicação. Declaro, para os devidos fins que estou ciente:

- dos Artigos 297 a 299 do Código Penal, Decreto-Lei n. 2.848 de 7 de dezembro de 1940;
- da Lei n. 9.610, de 19 de fevereiro de 1998, sobre os Direitos Autorais; e
- que plágio consiste na reprodução de obra alheia e submissão da mesma como trabalho próprio ou na inclusão, em trabalho próprio, de ideias, textos, tabelas ou ilustrações (quadros, figuras, gráficos, fotografias, retratos, lâminas, desenhos, organogramas, fluxogramas, plantas, mapas e outros) transcritos de obras de terceiros sem a devida e correta citação da referência.

O corpo docente responsável pela avaliação deste trabalho poderá não aceitar o referido trabalho caso os pontos mencionados acima sejam descumpridos, por conseguinte, considerar-me reprovado.

_____

Assinatura do acadêmico(a)
Boa Vista - RR, data (por extenso).