

UNIVERSIDADE DE LISBOA  
FACULDADE DE CIÊNCIAS  
DEPARTAMENTO DE INFORMÁTICA



## **Visão Computacional para Soluções de Segurança Adicional em Máquinas de Auto-Atendimento**

Vasco Miguel Raimão Maria

**Mestrado em Engenharia Informática**

Trabalho de Projeto orientado por:  
Prof. Doutor Nuno Cruz Garcia  
Eng. Dino Coutinho



## Agradecimentos

Quero começar por agradecer profundamente à minha mãe, Dina, pela presença incansável nos momentos em que mais precisei, pelo amor e apoio incondicional e pelas palavras de incentivo que tantas vezes me deram força para seguir em frente.

Estendo este agradecimento a toda a minha família, pelo apoio constante ao longo desta jornada, em especial aos meus avós, que sempre procuraram proporcionar-me o melhor dentro das suas possibilidades e cujo amor esteve sempre presente, bem como à minha namorada, pelo amor, paciência e compreensão ao longo deste percurso, pelo apoio constante e por me ajudar, de diversas formas, a manter o foco e a concretizar esta dissertação.

Quero igualmente expressar a minha gratidão ao Professor Nuno Garcia, meu orientador académico, pela orientação dedicada, pela disponibilidade e, em especial, pela ajuda na compreensão dos fundamentos de aprendizagem profunda, área com a qual não tinha tido contacto prévio. A sua orientação foi essencial para o desenvolvimento deste projeto e para o meu crescimento técnico ao longo da tese.

Agradeço também ao Engenheiro Dino Coutinho, meu orientador na empresa Innovation Makers Labs, pela forma positiva e motivadora com que recebeu o meu interesse em realizar esta tese e pela oportunidade de a desenvolver em contexto profissional. Um agradecimento especial ao Engenheiro Paulo Filipe, pelo envolvimento direto no desenvolvimento do trabalho e pela constante disponibilidade para esclarecer dúvidas, partilhar conhecimento e contribuir ativamente para a sua concretização.

Estendo ainda o meu agradecimento a toda a equipa da Innovation Makers Labs pelo acolhimento e pelo apoio prestado durante este percurso. Agradeço também ao projeto ATTRACT pela bolsa atribuída, que permitiu apoiar o desenvolvimento deste trabalho.

Por fim, agradeço a todos aqueles que, de forma direta ou indireta, contribuíram para a realização desta tese.

*Dedico aos meus pais, Dina e Rui, por todo o amor, apoio e educação.*

## Resumo

As máquinas de autoatendimento fazem parte do quotidiano e assumem um papel central na prestação de diversos serviços, permitindo a realização de operações de forma rápida, autónoma e acessível. Contudo, a sua utilização crescente levanta desafios significativos no domínio da segurança, exigindo mecanismos capazes de observar, interpretar e reagir a situações de risco em tempo real.

No âmbito de um estágio, foi desenvolvido um sistema de segurança para máquinas de autoatendimento com o objetivo de reforçar a proteção dos equipamentos e assegurar que as operações decorrem de forma segura e fiável, contribuindo também para a segurança dos utilizadores. A solução recorre a técnicas de visão computacional e utiliza a câmara incorporada na máquina para analisar continuamente o ambiente e as interações que nele ocorrem, identificando sinais visuais de risco e fornecendo informação que apoia decisões rápidas e fundamentadas.

A simplicidade de integração foi um princípio orientador no desenvolvimento, permitindo que o sistema possa ser instalado sem alterar a aplicação principal da máquina, preservando todas as funcionalidades originais. Esta abordagem reduz o impacto na operação, facilita a adoção em diferentes contextos e diminui os custos de implementação.

O sistema foi concebido com uma arquitetura flexível e evolutiva, possibilitando a integração eficiente de novos modelos e tecnologias de visão computacional de forma ágil, acompanhando os avanços da área e adaptando-se a necessidades futuras.

Os resultados obtidos demonstram que a solução é viável para utilização em máquinas de autoatendimento, sendo capaz de operar de forma contínua e de sinalizar eventos críticos com latência compatível com o contexto operacional. A validação realizada com dados representativos de diferentes cenários de utilização confirma um desempenho consistente, com um valor de recall de aproximadamente 0.77, evidenciando a capacidade do sistema para identificar eficazmente situações de risco sem interferir com o funcionamento normal da máquina.

**Palavras-chave:** Máquinas de autoatendimento, Visão computacional, Deteção em Tempo Real, Análise Comportamental, Monitorização Contínua



## Abstract

Self-service machines have become an integral component of modern daily life and play a central role in the delivery of various services. They enable operations to be carried out efficiently, autonomously, and accessibly, responding to the demand for solutions that save time and increase efficiency. However, their growing use also brings challenges in the field of security, requiring mechanisms capable of observing, interpreting, and reacting to risk situations in real time.

Within the scope of an internship, a security system for self-service machines was developed with the objective of reinforcing equipment protection and ensuring that operations are carried out in a safe and reliable manner, while also contributing to user security. The solution relies on computer vision techniques and uses the machine's built-in camera to continuously analyze the surrounding environment and the interactions taking place within it, identifying visual signs of risk and providing information that supports timely and well-informed decisions.

Ease of integration was adopted as a guiding principle during development, allowing the system to be deployed without modifying the machine's main application and preserving all original functionalities. This approach reduces operational impact, facilitates adoption across different contexts, and lowers implementation costs.

The system was designed with a flexible and evolutive architecture, enabling the efficient integration of new computer vision models and technologies in an agile manner, keeping pace with advances in the field and adapting to future requirements.

The results obtained demonstrate that the proposed solution is viable for deployment in self-service machines, being able to operate continuously and to signal critical events with latency compatible with the operational context. Validation performed using data representative of different usage scenarios confirms consistent performance, achieving a recall value of approximately 0.77, which highlights the system's ability to effectively identify risk situations without interfering with the normal operation of the machine.

**Keywords:** Self-service machines, Computer vision, Real-time detection, Behavioral analysis, Continuous monitoring



# Conteúdo

<b><a href="#">Lista de Figuras</a></b>	<b>xiv</b>
<b><a href="#">Lista de Tabelas</a></b>	<b>xv</b>
<b><a href="#">Siglas e Acrónimos</a></b>	<b>xviii</b>
<b>1 <a href="#">Introdução</a></b>	<b>1</b>
1.1 <a href="#">Contextualização</a> . . . . .	1
1.2 <a href="#">Motivação</a> . . . . .	2
1.3 <a href="#">Objectivos</a> . . . . .	2
1.4 <a href="#">Contribuições</a> . . . . .	3
1.5 <a href="#">Modelo de Ameaça e Pressupostos</a> . . . . .	4
1.6 <a href="#">Estrutura do documento</a> . . . . .	6
<b>2 <a href="#">Contexto</a></b>	<b>7</b>
2.1 <a href="#">Máquinas de Autoatendimento</a> . . . . .	7
2.2 <a href="#">Principais Ameaças</a> . . . . .	8
2.3 <a href="#">Projeto HEFESTO</a> . . . . .	9
2.4 <a href="#">Visão Computacional e Aprendizagem Profunda</a> . . . . .	11
2.5 <a href="#">Aprendizagem automática em Produção</a> . . . . .	12
2.5.1 <a href="#">Previsões em lote e Previsões em tempo real</a> . . . . .	12
2.5.2 <a href="#"><i>Edge Computing e Cloud Computing</i></a> . . . . .	12
2.6 <a href="#">Rastreamento e Detecção em tempo real</a> . . . . .	13
2.7 <a href="#">Análise comportamental</a> . . . . .	15
<b>3 <a href="#">Trabalho relacionado</a></b>	<b>19</b>
3.1 <a href="#">Abordagens Baseadas em Redes Neuronais para Detecção de Objetos</a> . . . . .	19
3.1.1 <a href="#">Limitações dos Métodos Tradicionais de Visão Computacional</a> . . . . .	19
3.1.2 <a href="#">Evolução das Arquiteturas com Redes Neuronais</a> . . . . .	19
3.1.3 <a href="#">Modelos de Uma Etapa para Detecção em Tempo Real</a> . . . . .	20
3.1.4 <a href="#">Distinção entre Modelos de Detecção e de Classificação</a> . . . . .	21
3.1.5 <a href="#">Arquiteturas com Transformadores</a> . . . . .	22
3.2 <a href="#">Modelos de Detecção em Tempo Real</a> . . . . .	23
3.2.1 <a href="#">Limitações dos Modelos com NMS</a> . . . . .	23

3.2.2 Arquiteturas <i>End-to-End</i> : DETR e RT-DETR . . . . .	23
3.2.3 Avanços com YOLOv10 . . . . .	23
3.2.4 Comparação de Desempenho . . . . .	23
3.3 Abordagens para Rastreamento de Múltiplos Objetos . . . . .	24
3.4 Fluxo para Detecção e Prevenção de Crimes em ATMs . . . . .	25
3.5 Reconhecimento de Emoções Faciais . . . . .	27
<b>4 Metodologia</b>	<b>29</b>
4.1 Exploração Inicial das Tecnologias de Visão Computacional . . . . .	29
4.2 Construção do Sistema de Segurança . . . . .	30
4.2.1 Objetivos Técnicos e Critérios de Conceção . . . . .	30
4.2.2 Arquitetura Modular e Estrutura Plug-and-Play . . . . .	30
4.2.3 Descrição da Arquitetura e dos seus Componentes . . . . .	31
4.2.4 Regras de Decisão . . . . .	38
4.3 Dinâmica Operacional do Sistema de Segurança . . . . .	42
4.3.1 Fluxo Interno de Processamento . . . . .	42
4.3.2 Fluxo Cíclico do Sistema . . . . .	44
4.3.3 Implantação Local, Integração com o HEFESTO e Considerações de Privacidade	47
<b>5 Análise</b>	<b>49</b>
5.1 Avaliação dos modelos de Computação Visual . . . . .	49
5.1.1 Pesquisa e Seleção de Datasets . . . . .	49
5.1.2 Procedimento de Avaliação dos Modelos . . . . .	51
5.1.3 Avaliação dos Modelos de Detecção de Armas . . . . .	52
5.1.4 Avaliação dos Modelos de Reconhecimento de Emoções . . . . .	54
5.1.5 Avaliação do Modelo de Detecção de Ambiente Violento	55
5.2 Validação do Sistema de Segurança . . . . .	56
5.2.1 Criação do Dataset de Validação . . . . .	56
5.2.2 Avaliação dos Modelos por Origem dos Dados . . . . .	60
5.2.3 Avaliação da Influência dos Modelos . . . . .	63
5.2.4 Ajuste dos Pesos no Ficheiro de Configuração . . . . .	66
5.2.5 Avaliação da métrica de proximidade facial	70
5.3 Análise de Desempenho e Otimizações . . . . .	73
5.3.1 Comparação entre YOLO com e sem NMS . . . . .	73
5.3.2 Avaliação do Processamento em Lote . . . . .	73
5.3.3 Benchmark na Máquina de Autoatendimento	75
<b>6 Conclusão</b>	<b>77</b>
6.1 Síntese Final . . . . .	77
6.2 Trabalho Futuro . . . . .	78
<b>Glossário</b>	<b>82</b>





# Listas de Figuras

2.1 Módulos configuráveis do HEFESTO. . . . .	10
2.2 Máquina de autoatendimento HEFESTO com configuração base. [1, 2] . . . . .	10
2.3 Ilustração de MOT em tempo real [3]. . . . .	13
2.4 Fluxo de processamento de um sistema de deteção e rastreamento em tempo real (elaboração própria). . . . .	14
2.5 Deteção automatizada de ação anómala, com as caixas verdes a representar o <i>ground truth</i> e as caixas vermelhas as previsões do sistema. [4]. . . . .	15
2.6 Visão geral do processo de deteção de comportamentos violentos baseado em estimativa de pose. Adaptado de [5]. . . . .	16
2.7 Exemplo da aplicação da estimativa de pose na análise comportamental. Adaptado de [5]. . . . .	17
3.1 Arquitetura do modelo R-CNN . . . . .	20
3.2 Arquitetura do modelo Faster R-CNN . . . . .	20
3.3 Arquitetura genérica do modelo YOLO. . . . .	21
3.4 Comparação entre classificação (esquerda) e deteção (direita). . . . .	22
3.5 Comparação de métricas entre YOLOv10 e outras arquiteturas, incluindo latência e número de parâmetros. Gráficos retirados de [6]. . . . .	24
3.6 Comparação da precisão entre rastreadores do estado da arte. Gráfico retirado de [7]. . . . .	25
3.7 Fluxo funcional de um sistema de segurança para ATMs baseado em deteção de objetos e reconhecimento de atividades suspeitas. [8] . . . . .	26
3.8 Pipeline geral de um sistema de reconhecimento de emoções faciais [9]. . . . .	28
4.1 Arquitetura geral do Sistema de Segurança, com destaque na ligação assinalada em (1) para os serviços dos modelos de computação visual. . . . .	31
4.2 Exemplo de utilização da métrica de áreas faciais para informar sobre situações de intrusão. . . . .	37
4.3 Fluxo global de processamento dos serviços de análise emocional (DeepFace e FER). . . . .	38
4.4 Diagrama dos níveis de decisão realizados pelo controlador no Sistema de Segurança. . . . .	39
4.5 Excerto descritivo do config.json, com indicação da função de cada parâmetro. . . . .	41
4.6 Fluxo interno de processamento no Sistema de Segurança, desde a receção dos frames até à decisão final. . . . .	43
4.7 Exemplo de ciclo de análise no Sistema de Segurança, com receção contínua de 5 FPS e envio em lote para os modelos de computação visual, sendo W a janela de decisão. . . . .	45
4.8 Arquitetura local do sistema de segurança e sua integração com o HEFESTO. . . . .	48

5.1 Matriz de confusão resultante da avaliação do modelo <i>ViolenceModel</i> no <i>dataset RWF-2000</i> [10]. . . . .	56
5.2 Distribuição do número de imagens do <i>dataset</i> de validação por fonte. . . . .	57
5.3 Distribuição do número de imagens do <i>dataset</i> de validação por classe. . . . .	58
5.4 Exemplos de imagens incluídas no <i>dataset</i> de validação, contendo casos de <i>alarme</i> (situações de risco) e de <i>não alarme</i> (utilização normal). . . . .	59
5.5 Distribuição do número de imagens por fonte após a deduplicação. . . . .	60
5.6 Importância média das <i>features</i> (modelos) segundo o Random Forest, com desvio padrão. . . . .	64
5.7 Curvas ROC obtidas para as categorias armas e emoções. O ponto azul representa o limiar ótimo segundo o índice de Youden, com o respetivo valor indicado. A linha diagonal corresponde ao desempenho esperado por acaso (AUC = 0.5). . . . .	69
5.8 Comparação das métricas do sistema antes e após o ajuste dos pesos, com base na execução sobre o <i>dataset</i> de validação. . . . .	69
5.9 Distribuição das imagens do dataset pelas classes negativas e positivas. . . . .	71
5.10 Distribuição da razão percentual entre a segunda maior face e a face principal, para as classes negativas e positivas. . . . .	72
5.11 Comparação dos tempos de execução do modelo YOLO com e sem NMS, com base em 50 imagens. . . . .	73

# Listas de Tabelas

1.1 Resumo dos objetivos do sistema de segurança proposto . . . . .	3
2.1 Principais ameaças associadas ao uso de ATMs . . . . .	8
2.2 Especificações de hardware do HEFESTO (Bmax) . . . . .	11
4.1 Exemplo de entradas agregadas por lote de 2 <i>frames</i> , onde cada modelo aparece uma vez e o campo <i>detections</i> inclui os resultados de ambos os <i>frames</i> . . . . .	42
5.1 Datasets de detecção de armas e respectivas métricas publicadas. . . . .	50
5.2 Resumo dos datasets organizados por categoria, utilizados na avaliação dos modelos. . . . .	51
5.3 Desempenho dos modelos na detecção de armas, avaliados sobre múltiplos datasets. . . . .	53
5.4 Desempenho dos modelos de reconhecimento de emoções nas classes <i>Angry</i> e <i>Fear</i> . . . . .	55
5.5 Desempenho do modelo <i>ViolenceModel</i> ao dataset RWF-2000 [10]. . . . .	56
5.6 Critérios de classificação utilizados nas imagens do dataset de validação. . . . .	58
5.7 Impacto da deduplicação no subconjunto de imagens de fontes públicas. . . . .	60
5.8 Métricas de classificação por modelo e subconjunto de validação ( <i>leave-source-out</i> ). . . . .	62
5.9 Resultados médios das métricas com IC95% (5×2 CV + bootstrap). . . . .	64
5.10 Valores de AUC dos modelos das categorias de armas e emoções no dataset de validação. . . . .	66
5.11 Correlação entre pares de modelos segundo os coeficientes de Pearson e Spearman. . . . .	67
5.12 Comparação dos tempos de processamento por lote em ambos os ambientes de teste. . . . .	74
5.13 Benchmarks dos modelos de visão computacional, do sistema de segurança e da aplicação nativa ( <i>kiosk</i> ). . . . .	75



# Siglas e Acrónimos

**API** Interface de Programação de Aplicações. Tradução de: *Application Programming Interface*. 30

**ATM** *Automated Teller Machine*. 1

**BoVW** Modelo de visão computacional baseado na quantização de descritores locais.. 19

**CCTV** *Closed-Circuit Television*. 2

**CLIP** Modelo multimodal de *deep learning* treinado pela OpenAI que projeta imagens e texto num mesmo espaço vetorial. Permite medir a semelhança entre representações visuais e textuais através de embeddings.. 59

**CNN** Redes Neuronais Convolucionais, Tradução de: *Convolutional Neural Networks*. 11

**COCO** Objetos Comuns em Contexto, base de dados de avaliação, Tradução de: *Common Objects in Context*. 24

**CPU** Unidade Central de Processamento, Tradução de: *Central Processing Unit*. 11

**DETR** Transformador de Detecção, Tradução de: *Detection Transformer*. 22

**DL** Ramo da Aprendizagem Automática que utiliza redes neurais artificiais com múltiplas camadas para modelar relações complexas em dados. Tradução de: *Deep Learning*. 11

**EAST** Associação Europeia para Transações Seguras, Tradução de: *European Association for Secure Transactions*. 1

**FER** Reconhecimento de Emoções Faciais, Tradução de: *Facial Expression Recognition*. 27, 81

**FIFO** Política de filas onde o primeiro elemento a entrar é o primeiro a sair.. 32

**FPS** *Frames* por Segundo, Tradução de: *Frames Per Second*. 25, 32, 78

**GPU** Unidade de Processamento Gráfico, Tradução de: *Graphics Processing Unit*. 11

**HOG** Histograma de Gradientes Orientados, Tradução de: *Histogram of Oriented Gradients*. 27

**HOTA** Precisão de Rastreamento de Ordem Superior, Tradução de: *Higher Order Tracking Accuracy*.

**HTTP** Protocolo de Transferência de Hipertexto. Tradução de: *Hypertext Transfer Protocol.* 31

**IA** Inteligência Artificial. 9

**IC** Intervalo de Confiança (ex.: IC 95% para 95%). 63, 64

**INM** *Innovation Makers Labs.* 1

**IoT** Internet das Coisas, Tradução de: *Internet of Things.* 11

**IoU** Interseção sobre União, métrica de avaliação, Tradução de: *Intersection over Union.* 24

**JSON** Notação de Objeto JavaScript. Tradução de: *JavaScript Object Notation.* 33

**mAP** Média da Precisão, Tradução de: *mean Average Precision.* 24

**ML** Aprendizagem Automática, Tradução de: *Machine Learning.* 12

**MOT** Rastreamento de Múltiplos Objetos, Tradução de: *Multiple Object Tracking.* 13

**NMS** Supressão Não-Máxima, Tradução de: *Non-Maximum Suppression.* 22, 23

**P95** 95.<sup>º</sup> percentil (valor abaixo do qual se encontram 95% das observações). 5, 74

**PIN** Personal Identification Number. 9

**R-CNN** Modelo de deteção de objetos que utiliza regiões propostas para aplicar uma rede convolucional.. 19

**RGB** *Red, Green, Blue.* 5

**RPN** Rede de Propostas de Regiões, Tradução de: *Region Proposal Network.* 20

**SaaS** Software como Serviço, Tradução de: *Software as a Service.* 12

**SORT** Algoritmo de rastreio em tempo real baseado em modelos de movimento linear e associação por IoU.. 24

**SSD** *Single Shot Multibox Detector.* 13

**SVM** Máquinas de Vetores de Suporte, Tradução de: *Support Vector Machines.* 27

**TbD** *Tracking-by-Detection*, técnica de rastreio em que a deteção é usada como base. 24

**VTM** Virtual Teller Machine. 9

**YOLO** *You Only Look Once.* 13

# Capítulo 1

## Introdução

As máquinas de autoatendimento têm-se consolidado como um elemento central nos processos de transformação digital em setores como serviços financeiros e seguradoras, de modo a oferecer conveniência e eficiência tanto para os utilizadores como para as empresas, permitindo que os clientes beneficiem de maior autonomia e disponibilidade temporal. No entanto, apesar da sua utilidade, estas máquinas continuam a enfrentar desafios significativos no que diz respeito à segurança, sobretudo na prevenção de ameaças físicas. Com o avanço das tecnologias de visão computacional e aprendizagem automática surgem novas oportunidades de desenvolver soluções de segurança que ajudem a mitigar alguns riscos, permitindo a deteção automática de potenciais ameaças em tempo real. Este trabalho foi desenvolvido durante um estágio na empresa *Innovation Makers Labs* (INM), no âmbito de implementar uma camada adicional de segurança num projeto interno denominado de HEFESTO.

### 1.1 Contextualização

Nos últimos anos, os incidentes relacionados com fraudes e ataques físicos em máquinas de autoatendimento tornaram-se uma preocupação crescente para instituições financeiras e para os próprios clientes. Assaltos, tentativas de coação e vandalismo são algumas das ameaças enfrentadas, tornando evidente a necessidade de reforçar a segurança destas máquinas. Segundo o relatório da Associação Europeia para Transações Seguras (EAST), os ataques físicos a *Automated Teller Machine* (ATM) na Europa registaram um aumento de 24% em 2023, totalizando 4637 incidentes [1]. Estes ataques resultaram em prejuízos superiores a 9 milhões de euros, evidenciando o impacto financeiro significativo associado a estas ocorrências. Alguns destes incidentes envolveram assaltos diretos aos utilizadores das máquinas, muitas vezes sob coação, além de vandalismo e fraudes eletrónicas. Estes números reforçam a necessidade de soluções mais avançadas que permitam a deteção de ameaças e uma resposta eficaz em tempo real. Atualmente, muitas soluções de segurança limitam-se a medidas reativas, como a análise posterior de imagens de videovigilância. No entanto, com os avanços em visão computacional e aprendizagem automática, através da própria câmara da máquina de autoatendimento é possível desenvolver sistemas de segurança que detetem ameaças de forma autónoma, nomeadamente através da deteção de armas, do reconhecimento de emoções associadas a situações de risco, como o medo, ou da identificação de ambientes violentos, permitindo sinalizar eventos críticos e contribuir para a proteção da integridade das operações, da segurança dos utilizadores e do próprio equipamento, reduzindo o impacto de incidentes e perdas associadas a situações de risco.

## 1.2 Motivação

Apesar da crescente necessidade de reforçar a segurança em máquinas de autoatendimento, continua a não existir uma solução que permita a integração direta de sistemas de segurança baseados em visão computacional nas máquinas de autoatendimento. Em particular, faltam abordagens do tipo "*plug and play*" capazes de detetar automaticamente, e em tempo real, comportamentos suspeitos ou ameaças físicas. A maioria das soluções de segurança baseadas em visão computacional encontra-se integrada em sistemas de videovigilância tradicionais, como *Closed-Circuit Television* (CCTV), os quais alguns já incorporam algoritmos de deteção [12]. No entanto, esses sistemas estão maioritariamente posicionados em ambientes externos às máquinas de autoatendimento. Um sistema de segurança integrado diretamente nas máquinas permitiria atuar de forma imediata, identificando ameaças no próprio ponto de interação, aumentando assim a segurança das operações e a proteção dos utilizadores. Além disso, a falta de uma solução modular e adaptável dificulta a adoção de novas tecnologias sem necessidade de grandes alterações na infraestrutura. Desta forma as máquinas de autoatendimento podem beneficiar de sistemas de segurança com uma abordagem mais integrada, onde a deteção de ameaças físicas em tempo real forneça informação útil para uma resposta mais eficaz. Ao atuar como uma camada adicional de proteção, o sistema desenvolvido neste trabalho permite que o cliente final disponha de informação necessária para acionar alarmes que permitam às entidades responsáveis, como instituições bancárias, decidir as ações mais adequadas, tais como bloquear as operações na máquina, acionar protocolos de segurança ou até alertar as autoridades, caso necessário.

## 1.3 Objectivos

O principal objetivo deste trabalho é desenvolver um sistema de segurança avançado, baseado em visão computacional e aprendizagem automática, capaz de detetar em tempo real situações de ameaças físicas nas máquinas de autoatendimento, atuando como uma camada adicional de segurança. Este sistema deverá monitorizar e analisar continuamente as interações dos utilizadores com a máquina e, sempre que sejam identificadas situações potencialmente perigosas, como a presença de armas ou comportamentos violentos, fornecer uma resposta analítica que permita à máquina de autoatendimento adotar ações preventivas ou corretivas. O intuito é disponibilizar uma camada adicional de informação útil, assegurando uma resposta rápida e eficaz que contribua para a mitigação imediata dos riscos e para o reforço da segurança global da máquina, apoiando tanto cenários em que já existam mecanismos de segurança de primeira linha, como a deteção de *spoofing* em sistemas de biometria por câmara ou leitores de cartões baseados em tecnologia de infravermelhos, como também contextos em que tais camadas de proteção não estão presentes, aumentando assim a resiliência global da infraestrutura de autoatendimento.

Para além deste objetivo principal, pretende-se ainda cumprir um conjunto de requisitos específicos que assegurem a aplicabilidade prática e a evolução contínua do sistema. Assim, o sistema deverá ser modular e adaptável, permitindo a sua integração em diferentes tipos de máquinas de autoatendimento, particularmente no setor bancário, sem que sejam necessárias alterações estruturais significativas. Deverá igualmente ser expansível, adotando uma abordagem "*plug and play*" que possibilite a sua atualização e melhoria ao longo do tempo, através da integração de novos modelos ou tecnologias, como modelos de deteção de armas ou de reconhecimento de emoções faciais. Por fim, o sistema deve contribuir de forma

efetiva para a proteção dos utilizadores e para a integridade das operações, oferecendo uma camada de segurança sólida, funcional e eficiente.

De forma a sintetizar estes aspectos, a Tabela 1.1 apresenta uma visão resumida do objetivo principal e dos requisitos específicos estabelecidos.

Tabela 1.1: Resumo dos objetivos do sistema de segurança proposto

<b>Tipo de objetivo</b>	<b>Descrição</b>
Objetivo principal	Desenvolver um sistema de segurança avançado, baseado em visão computacional e aprendizagem automática, capaz de detetar em tempo real situações de ameaças físicas nas máquinas de autoatendimento e atuar como uma camada adicional de proteção.
Complementaridade	Reforçar tanto máquinas que já possuem mecanismos de segurança de primeira linha (ex.: deteção de <i>spoofing</i> em biometria por câmera, leitores de cartões com tecnologia de infravermelhos) como também aquelas que não dispõem de tais medidas, aumentando a resiliência global da infraestrutura.
Modularidade	Assegurar que o sistema seja modular e adaptável, permitindo a integração em diferentes tipos de máquinas de autoatendimento, especialmente no setor bancário, sem alterações estruturais significativas.
Expansibilidade	Garantir uma abordagem “ <i>plug and play</i> ”, que possibilite atualizações futuras e a incorporação de novos modelos ou tecnologias, como deteção avançada de armas ou reconhecimento de emoções faciais.
Proteção	Contribuir para a proteção dos utilizadores e para a integridade das operações, fornecendo uma camada de segurança sólida, funcional e eficiente.

## 1.4 Contribuições

O trabalho desenvolvido nesta tese contribui para a área da segurança em máquinas de autoatendimento, abordando a ausência de soluções que recorram à visão computacional para detetar, em tempo real, ameaças físicas ou comportamentos suspeitos. Com base nas limitações identificadas na literatura consultada e nas oportunidades observadas ao longo da análise técnica, apresentam-se as seguintes contribuições:

- Proposta e implementação de um sistema de segurança baseado em visão computacional, capaz de detetar em tempo real comportamentos suspeitos e ameaças físicas, com foco na aplicabilidade em contextos reais de utilização.
- Desenvolvimento de uma arquitetura modular e extensível, permitindo a integração e substituição de modelos de computação visual de forma independente e sem necessidade de alterações estruturais na aplicação.
- Concepção de mecanismos de decisão capazes de agregar os resultados dos diferentes modelos de computação visual utilizados pelo sistema de segurança, permitindo uma avaliação do risco mais informada, consistente e ajustável ao contexto operacional da máquina de autoatendimento.

- Avaliação comparativa de modelos de computação visual, com análise de métricas como tempo de inferência, consumo de memória e precisão, com o objetivo de informar decisões futuras de integração e otimização.
- Utilização de processamento em lote de *frames*, contribuindo para reduzir o tempo médio por imagem e aumentar a eficiência do sistema sem comprometer a reatividade.
- Validação prática e análise interpretável, com testes em hardware limitado, uso de um *dataset* realista e aplicação de técnicas de análise para aferir a influência dos modelos na decisão final.

## 1.5 Modelo de Ameaça e Pressupostos

Esta secção apresenta os elementos de enquadramento que sustentam o trabalho proposto, descrevendo os atores envolvidos, o ambiente de operação, as capacidades e limitações do sistema, a definição de alarmes e alertas, as metas de desempenho, os aspetos fora de âmbito e os pressupostos considerados.

### Atores e comportamentos

O trabalho proposto considera dois tipos principais de atores: o utilizador legítimo da máquina de autoatendimento, que em situações normais interage de forma regular e não hostil mas que pode também manifestar emoções de medo quando sujeito a intimidação, e potenciais agressores que podem surgir como transeuntes isolados ou em grupo, adotando comportamentos suspeitos ou hostis. Estes comportamentos incluem a exibição de armas de fogo ou de armas brancas de pequenas dimensões, atitudes de agressividade física como aproximação ameaçadora, confrontos interpessoais com empurrões ou lutas, bem como a manifestação de expressões faciais de agressividade e intimidação.

### Ambiente

O ambiente de operação contempla tanto cenários interiores como exteriores, sendo a câmara embutida na própria máquina de autoatendimento e permanecendo em posição e ângulo fixos. O campo de visão cobre distâncias típicas de meio a dois metros e permite a observação da região superior do corpo dos indivíduos, incluindo de forma consistente as faces e os movimentos dos membros superiores, o que possibilita a análise de expressões faciais, gestos e interações físicas entre utilizador e potenciais transeuntes. Assume-se que, no enquadramento da câmara, podem surgir apenas o utilizador legítimo, o utilizador acompanhado de agressores ou, em alguns casos, apenas os próprios agressores, sendo ainda possível que armas ou objetos de interesse se encontrem parcial ou totalmente ocultos pela posição corporal ou pela interação entre os indivíduos, bem como que os agressores se apresentem com o rosto total ou parcialmente coberto, o que pode limitar a análise de expressões faciais. As condições de iluminação podem variar de acordo com o contexto, sendo consideradas apenas situações que garantam níveis mínimos de visibilidade adequados à análise, excluindo-se cenários de escuridão total ou de luminosidade extrema permanente. Em situações em que a câmara não disponha de condições adequadas, por exemplo devido a obstrução, pintura, colocação de objetos ou incidência direta de luz intensa, o sistema gera um alerta para assinalar a incapacidade de análise.

## Capacidades e limitações

A solução baseia-se na receção sequencial de *frames Red, Green, Blue* (RGB) fornecidos em tempo real pela câmara embutida na máquina de autoatendimento, que são processados localmente em lotes sem qualquer armazenamento persistente, assegurando o cumprimento das restrições de privacidade. Assume-se um *throughput* mínimo de aproximadamente um *frame* por segundo, considerado adequado para garantir funcionamento contínuo mesmo em *hardware* limitado, embora valores superiores possam ser alcançados em equipamentos mais capazes.

## Definição de alarmes e alertas

Um evento de alarme ocorre quando, a partir da combinação das três categorias de deteção, o sistema atinge o limiar definido pela regra de decisão configurada, aplicado sobre lotes de *frames* processados. As categorias consideradas são a presença de armas incluindo armas de fogo e armas brancas de pequenas dimensões a deteção de emoções negativas como medo ou agressividade no utilizador legítimo e a identificação de comportamentos classificados como violência. Situações de proximidade física excessiva ou degradação da qualidade da imagem são tratadas apenas como alertas, que não desencadeiam alarmes mas fornecem informação contextual que pode apoiar a interpretação do cenário.

## Metas de desempenho

No que respeita às metas de desempenho, procura-se maximizar o valor de *recall*, estabelecendo-se como objetivo mínimo um valor de 0.7 de forma a validar a viabilidade inicial do sistema. Foi também considerada como meta de referência uma latência *end-to-end* correspondente a um tempo médio de processamento por *frame* no lote inferior ou igual a um segundo 95.<sup>º</sup> percentil (P95). Esta métrica é entendida sobretudo como orientação para o desenho do sistema e não como requisito obrigatório. Não foi definido formalmente um teto de falsos alarmes por hora, embora se considere que o custo associado a não detetar uma ameaça real é superior ao custo de gerar falsos alarmes.

## Fora de âmbito

No âmbito desta dissertação não se inclui a deteção de objetos que não sejam armas, o uso de áudio, a integração multi-câmara ou o reconhecimento facial e biométrico. Também se considera fora de âmbito a decisão que ocorre após a emissão de um alerta ou de um alarme, isto é, as ações subsequentes que a máquina de autoatendimento ou a entidade bancária possam adotar. Embora possam ser apontados cenários de aplicação, como a interrupção da operação em curso ou a notificação automática das autoridades, a definição e implementação dessas medidas não fazem parte do presente trabalho.

## Pressupostos

Assume-se que a câmara embutida na máquina de autoatendimento mantém calibração e posição estáveis ao longo do tempo. Parte-se ainda do pressuposto de que não existem manipulações adversariais intencionais do campo visual, como padrões ou objetos especificamente concebidos para enganar os modelos de computação visual. Considera-se também que os modelos correm a uma taxa mínima de um *frame* por segundo, garantindo funcionamento contínuo. Todo o processamento bem como a emissão de alertas

e alarmes ocorrem localmente, pelo que a estabilidade de rede não é considerada um fator determinante para o desempenho do sistema.

Em síntese, esta secção estabelece o enquadramento necessário para a introdução do trabalho, clarificando os atores, os cenários de risco, as condições técnicas e os limites assumidos. Estes elementos constituem a base conceptual para a metodologia e para a avaliação apresentadas nos capítulos seguintes.

## 1.6 Estrutura do documento

O documento encontra-se estruturado em seis capítulos, complementados por um glossário que compila os principais termos técnicos utilizados ao longo do trabalho. A estrutura foi concebida para conduzir o leitor desde a contextualização inicial até à apresentação dos resultados e propostas de evolução futura, garantindo assim uma compreensão gradual e fundamentada do projeto.

O capítulo de **Introdução** apresenta o tema em estudo, a motivação que conduziu ao desenvolvimento do projeto, os objetivos definidos e as principais contribuições do trabalho. Este capítulo estabelece o enquadramento global e fornece ao leitor uma visão clara do propósito e da relevância do sistema de segurança proposto.

O capítulo de **Contexto** descreve em detalhe as máquinas de autoatendimento e os riscos de segurança a elas associados, analisando de que forma estas vulnerabilidades justificam a necessidade de mecanismos adicionais de proteção. São igualmente discutidos os fundamentos tecnológicos que suportam a proposta, nomeadamente a visão computacional e a aprendizagem automática, com foco na sua aplicação em cenários de tempo real.

O capítulo de **Trabalho Relacionado** reúne e discute estudos anteriores e soluções já existentes na literatura e no mercado, identificando modelos de deteção, arquiteturas recentes e diferentes abordagens ligadas à análise de vídeo e ao reconhecimento de padrões. Esta análise crítica permite destacar as limitações das soluções correntes e justificar as opções metodológicas adotadas neste trabalho.

O capítulo de **Metodologia**, é apresentada a abordagem seguida para o desenvolvimento do sistema de segurança, incluindo a arquitetura modular proposta, os critérios de conceção, os componentes implementados e a descrição do funcionamento em ambiente local. Este capítulo fornece uma visão detalhada sobre o processo de desenvolvimento e as decisões técnicas que suportam a solução.

O capítulo de **Análise** concentra-se na avaliação prática do sistema, abordando o desempenho dos modelos de inferência, a validação com dados realistas e a medição da eficiência em ambiente com os recursos computacionais limitados de uma máquina de autoatendimento. Esta análise empírica permite aferir a robustez da solução e a sua viabilidade em cenários reais.

Por fim, o capítulo de **Conclusão** sintetiza os principais resultados obtidos, reflete sobre a utilidade e aplicabilidade do sistema desenvolvido e aponta potenciais caminhos de evolução, sob a forma de perspetivas de trabalho futuro. Este encerramento assegura a ligação entre os objetivos inicialmente definidos e as contribuições efetivamente alcançadas.

A organização do documento, assim estruturada, visa não apenas apresentar os conteúdos de forma sequencial, mas também facilitar a compreensão e evidenciar a progressão lógica do raciocínio científico subjacente a todo o trabalho.

# **Capítulo 2**

## **Contexto**

### **2.1 Máquinas de Autoatendimento**

As máquinas de autoatendimento, como os ATMs, tornaram-se fundamentais em vários setores pela sua capacidade de otimizar processos, onde trouxeram conveniência e eficiência aos clientes. Em particular no setor bancário, estas máquinas permitem às instituições responder à crescente procura por serviços rápidos e seguros sem a necessidade de intervenção humana, melhorando assim a qualidade dos serviços oferecidos. Desde a década de 80, as máquinas de autoatendimento evoluíram de simples funções como levantamento de dinheiro, para máquinas que possibilitam uma vasta gama de serviços, como transferências, pagamentos de contas, consultas de saldo e outras operações financeiras complexas. Contudo, com esta evolução, surgiram novos desafios, como vulnerabilidades relacionadas com a segurança e falhas técnicas. Problemas como a clonagem de cartões, a falsificação de transações, a possibilidade de distribuição de notas falsas, eventos criminosos como assaltos diretos e tentativas de acesso ilegal aos compartimentos internos das máquinas são preocupações crescentes no contexto das máquinas de autoatendimento, principalmente no setor bancário. Além disso, falhas técnicas frequentes, como a indisponibilidade de dinheiro e falhas mecânicas durante os períodos de maior utilização ou em feriados, afetam diretamente a confiança dos consumidores. A fiabilidade das máquinas, incluindo a consistência das transações e a precisão dos registos de contas, também é fundamental para a confiança dos utilizadores [I3]. Existem diversos exemplos de implementações destas máquinas em diferentes setores de negócios. Em particular, os sistemas de autoatendimento têm sido amplamente adotados no setor dos transportes, como é o caso do terminal rodoviário de Larkin, na Malásia [I4]. Nesse local, foram instaladas máquinas de emissão de bilhetes eletrónicos (*e-ticketing*) com o objetivo de agilizar o processo de compra, sem necessidade de intervenção humana. Outro caso notável foi a implementação das máquinas interativas TOMI nas lojas do cidadão em Portugal [I5], que permitem aos utilizadores obter senhas virtuais e receber notificações sobre a sua vez, reduzindo assim filas e tempos de espera. Deste modo, a adoção das máquinas de autoatendimento tornou-se essencial não apenas no setor financeiro, mas também em diversos contextos operacionais e de prestação de serviços, contribuindo para a eficiência dos processos e satisfação dos utilizadores finais. No entanto, este crescimento torna também evidente a importância de considerar e resolver as vulnerabilidades associadas a estes sistemas, particularmente no que diz respeito à segurança, para garantir uma utilização segura e sustentável destas tecnologias.

## 2.2 Principais Ameaças

Os ATMs, embora convenientes, têm-se tornado alvos frequentes de diversos tipos de crimes, sobretudo físicos [11]. Estas ameaças podem ser classificadas em três categorias principais: fraude com cartões e numerário, ataques lógicos, que recorrem a dispositivos ou softwares maliciosos para manipular o funcionamento da máquina, e ataques físicos [16].

De entre estas categorias, é possível identificar um conjunto de ameaças recorrentes, que se encontram resumidas na Tabela 2.1. Esta apresentação em formato tabular permite uma visão mais clara e organizada das principais formas de ataque aos ATMs e respectivas características.

Tabela 2.1: Principais ameaças associadas ao uso de ATMs

<b>Tipo de Ameaça</b>	<b>Descrição/Observação</b>
Roubo a transportadores	Ataques direcionados às equipas responsáveis pelo abastecimento de numerário.
Furto de PINs	Obtenção ilícita de códigos de identificação dos utilizadores.
Interceção eletrónica de dados	Captura não autorizada de informações transmitidas pelo sistema.
Transações fraudulentas	Realização de operações bancárias por meios eletrónicos indevidos.
Desvio interno de numerário	Apropriação de valores diretamente dos ATMs por funcionários.
Assaltos	Ações violentas contra utilizadores ou contra o próprio equipamento.
Vandalismo	Danos intencionais à estrutura ou componentes dos ATMs.
Cumplicidade criminosa	Crimes realizados com a colaboração de terceiros.
Tentativas de homicídio	Situações extremas de violência associadas à utilização de ATMs.
Outros ataques físicos	Formas diversas de agressão direta ao equipamento.

Em locais como Cacuaco, em Angola, tem-se verificado um aumento preocupante de assaltos violentos a utilizadores de ATMs. Muitas destas pessoas já enfrentam dificuldades económicas significativas, e cada levantamento representa uma parte essencial para garantir a sobrevivência do agregado familiar. Contudo, ao dirigirem-se a uma máquina de autoatendimento, deparam-se frequentemente com cenários de insegurança, onde são abordadas por criminosos armados, coagidas a realizar levantamentos forçados e privadas dos escassos recursos de que dispõem [17]. Esta realidade evidencia de forma clara a fragilidade do atual modelo de autoatendimento no país e a urgência em adotar soluções mais modernas e eficazes, capazes de responder não apenas às necessidades operacionais do setor bancário, mas também de proteger os utilizadores em contextos de elevado risco social.

É precisamente neste enquadramento que se encontra o projeto HEFESTO, desenvolvido pela INM,

já em utilização no setor bancário em Angola.

## 2.3 Projeto HEFESTO

No contexto da modernização do autoatendimento bancário, têm surgido soluções que procuram ir além das funcionalidades tradicionais dos ATMs, oferecendo serviços mais completos e seguros. Entre estas soluções destaca-se o projeto HEFESTO, desenvolvido pela INM, empresa especializada em soluções digitais omnicanal que integra *hardware* e *software*, principalmente direcionadas ao setor financeiro. Este projeto consistiu no desenvolvimento de uma máquina de autoatendimento inovadora do tipo Virtual Teller Machine (VTM), concebida para ser modular, multifuncional e totalmente personalizável às necessidades específicas de cada cliente.

Enquanto os ATMs convencionais se limitam geralmente a operações básicas, o HEFESTO apresenta-se como uma evolução significativa relativamente aos ATMs, destacando-se por oferecer funcionalidades mais amplas e realizar operações complexas como abertura de contas, emissão de cartões bancários, pagamentos diversos e subscrição de produtos financeiros.

O HEFESTO foi desenhado com especial atenção para contextos em que existe uma elevada procura pelos balcões presenciais, como é o caso de Angola, onde frequentemente se verificam filas extensas para a execução de operações simples. A implementação do HEFESTO nestas situações permite que essas operações sejam realizadas de forma rápida e autónoma, reduzindo significativamente o volume de clientes nos balcões e otimizando o atendimento ao cliente.

Uma das grandes vantagens do HEFESTO reside na sua incorporação de tecnologias avançadas, tais como Inteligência Artificial (IA), aprendizagem automática e biometria, incluindo reconhecimento facial e verificação facial. Adicionalmente, ao integrar biometria facial e leitura de impressões digitais, o HEFESTO oferece uma vantagem clara face aos métodos tradicionais de autenticação baseados em cartões e códigos Personal Identification Number (PIN), que podem ser mais facilmente comprometidos. Estas tecnologias são cruciais para aumentar a segurança das transacções, protegendo os utilizadores contra fraudes e acessos não autorizados e, simultaneamente, tornam a experiência do cliente mais ágil e intuitiva.

Adicionalmente, a estrutura modular do HEFESTO garante às entidades bancárias uma elevada flexibilidade, permitindo que novos serviços e funcionalidades sejam facilmente adicionados e personalizados conforme as exigências do mercado e dos clientes, como ilustrado nas Figuras 2.1a e 2.1b, que apresentam respetivamente os módulos de emissão de cartões e pagamentos.



(a) Módulo de emissão de cartões integrado no HEFESTO (destacado a amarelo).

(b) Módulo de pagamentos integrado no HEFESTO (destacado a roxo).

Figura 2.1: Módulos configuráveis do HEFESTO.

Como ilustrado na Figura 2.2, o HEFESTO não se limita a substituir os ATMs convencionais, mas do que substituir acrescenta funcionalidades e redefine a experiência de autoatendimento no setor financeiro ao oferecer uma solução integrada, segura e eficiente ajustada às necessidades contemporâneas deste setor [2].



Figura 2.2: Máquina de autoatendimento HEFESTO com configuração base. [1, 2]

Neste contexto, o HEFESTO constitui o ambiente ideal para a implementação da solução proposta neste trabalho, que visa especificamente detetar e mitigar ameaças em máquinas de autoatendimento através da visão computacional. Sendo estas máquinas utilizadas frequentemente para processar informações altamente sensíveis (dados bancário, códigos PIN e dados pessoais dos clientes), torna-se imperativo assegurar que estejam equipadas com sistemas capazes de identificar situações de risco, como

operações realizadas sob ameaça ou coação.

Em complemento à análise funcional, é importante caracterizar o *hardware* do HEFESTO, baseado num mini-PC da marca Bmax, cujas capacidades computacionais influenciam diretamente a implementação de soluções de visão computacional.

A Tabela 2.2 apresenta as principais especificações técnicas do mini-PC. Para além deste, o equipamento inclui uma câmara embutida com resolução de  $720 \times 1280$  pixels e taxa de captura de 15 *frames* por segundo.

Tabela 2.2: Especificações de hardware do HEFESTO (Bmax)

Componente	Descrição
Unidade Central de Processamento (CPU)	Intel® Celeron® N5095 @ 2.00 GHz
Núcleos Lógicos	4
Núcleos Físicos	4
Frequência Máxima	2001 MHz
Memória RAM	7,78 GB Disponível
Unidade de Processamento Gráfico (GPU)	Nenhuma GPU dedicada

## 2.4 Visão Computacional e Aprendizagem Profunda

Neste trabalho, explora-se o uso da visão computacional como uma ferramenta essencial para reforçar a segurança em máquinas de autoatendimento. Esta tecnologia permite interpretar e processar informação visual de forma automatizada, simulando o comportamento da visão humana em tempo real [18]. Com algoritmos avançados, como as Redes Neuronais Convolucionais (CNN), torna-se possível identificar objectos suspeitos, rastrear movimentos no ambiente e reconhecer padrões de comportamento associados a potenciais ameaças, oferecendo uma camada adicional de proteção para os utilizadores.

O desenvolvimento de CNNs e de outras técnicas de Aprendizagem Profunda (DL) revolucionou a deteção e localização de objectos em tempo real, aliando eficiência computacional a elevada precisão. Li et al. [19] destacam a eficácia destas arquitecturas em tarefas como o reconhecimento facial e a segmentação de imagens. Tipicamente, uma CNN estrutura-se em três tipos principais de camadas:

- **Convolucionais**, que extraem características locais das imagens;
- **Pooling**, que reduzem a dimensionalidade preservando informação relevante;
- **Totalmente ligadas**, que combinam essas características para produzir a previsão final.

Um dos aspetos eficientes desta arquitetura é a partilha de pesos entre os filtros convolucionais, ou seja, o mesmo conjunto de pesos é aplicado repetidamente sobre diferentes regiões da imagem. Esta abordagem reduz a complexidade do modelo e permite extrair padrões visuais de forma mais eficiente, mantendo a precisão nas tarefas de deteção. A implementação de sistemas baseados em visão computacional envolve diversas etapas, como a deteção de características visuais (pontos de interesse, bordas e contornos), a segmentação de objectos e o reconhecimento de padrões comportamentais. Estas técnicas são fundamentais para identificar ameaças de forma rápida e eficaz, garantindo respostas adequadas a situações de risco [18]. Além disso, têm sido amplamente aplicadas em cenários de Internet das Coisas (IoT) para melhorar a segurança em ambientes inteligentes, como cidades, residências e espaços públicos. González García et al. propõem a integração da visão computacional em plataformas de IoT para automatizar a

deteção de pessoas e objectos em espaços privados, utilizando câmaras como sensores de imagem e activando respostas automáticas em casos de eventos suspeitos [20].

## 2.5 Aprendizagem automática em Produção

A aplicação de sistemas de Aprendizagem Automática (ML) em ambientes de produção exige abordagens específicas, que diferem substancialmente do contexto académico. Enquanto os modelos de ML feitos em ambiente académico dão prioridade à precisão e à exploração de conceitos inovadores, os modelos de ML para produção devem atender a requisitos como fiabilidade, escalabilidade, baixa latência e adaptabilidade. Estes sistemas também têm de responder aos requisitos empresariais e acomodarem-se às necessidades dos *stakeholders* com diferentes objetivos. Uma distinção significativa é no uso de dados. Em ambiente académico, é frequente utilizar conjuntos de dados estáticos e históricos, o que facilita o treino e os testes controlados. Em contraste, os sistemas de ML em produção lidam com fluxos de dados dinâmicos e em constante evolução, o que requer robustez para lidar com ruído, adaptações rápidas a alterações nas distribuições e atualizações regulares dos modelos para garantir um desempenho contínuo [21, 22].

### 2.5.1 Previsões em lote e Previsões em tempo real

As abordagens para previsões incluem dois métodos principais: previsões em lote e previsões em tempo real. As previsões em lote são adequadas para processar grandes volumes de dados em intervalos programados, sendo ideais para casos como sistemas de recomendação que não exigem respostas imediatas. Já as previsões em tempo real fornecem respostas instantâneas, fundamentais para cenários que requerem decisões rápidas, como a deteção de algo suspeito [21].

Para aproveitar ambas as abordagens, pode-se também adotar um sistema híbrido em que as previsões em lote atualizam periodicamente o modelo base, enquanto as previsões em tempo real garantem resposta imediata a novos eventos, equilibrando precisão e rapidez em produção. Foster & Kumar (2022) aplicaram esta abordagem num Software como Serviço (SaaS), atualizando periodicamente o modelo base através de lote e recorrendo a inferências imediatas para ajustar recomendações em tempo real, o que resultou em melhorias significativas na taxa de retenção de utilizadores e na satisfação dos clientes [23].

### 2.5.2 Edge Computing e Cloud Computing

Tecnologias como *Edge Computing* e *Cloud Computing* desempenham um papel importante nos sistemas de ML em produção [22]. O *Edge Computing* processa dados localmente, reduzindo a dependência de conectividade constante à internet, a latência e os custos de transferência de dados. Por exemplo, sensores industriais podem detetar e corrigir anomalias imediatamente, mesmo sem ligação contínua à nuvem [24]. Já o *Cloud Computing* usa servidores remotos, que oferece maior capacidade de processamento, para treinar modelos com grandes volumes de dados históricos, mas exige uma ligação rápida à rede [21]. A combinação de ambos corta o tráfego na rede, torna as previsões mais rápidas e permite escalar o processamento, assegurando fiabilidade e bom desempenho [25].

Em *Edge Computing*, muitos dos modelos ML são implantados no dispositivo, permitindo inferência local em tempo real e autonomia operacional mesmo quando a ligação à internet falha. Periodicamente, o

dispositivo recebe apenas as atualizações de que necessita, mantendo o sistema leve e evitando sobrecarga da rede [24].

## 2.6 Rastreamento e Detecção em tempo real

O rastreamento em tempo real foi identificado como um requisito inicial pela INM [1] para responder às necessidades de segurança e às exigências dos clientes em máquinas de autoatendimento. Esta tecnologia pode desempenhar um papel fundamental na deteção e monitorização de objetos em movimento contínuo. A tarefa de Rastreamento de Múltiplos Objetos (MOT) envolve a localização de múltiplos alvos e a manutenção de suas identidades ao longo de sequências de vídeo, como é demonstrado pela figura 2.3.



Figura 2.3: Ilustração de MOT em tempo real [3].

Este processo permite extrair informações do ambiente em tempo real, como o número de pessoas presentes ou a sua movimentação num determinado espaço, usando as capturas das câmaras [26].

Embora o MOT ofereça soluções robustas, a sua implementação enfrenta desafios técnicos importantes. Por exemplo, um dos métodos mais utilizados, o "tracking-by-detection", inicia a deteção ao identificar objetos em cada frame e, em seguida, associa essas deteções para formar trajetórias contínuas [27]. No entanto, este método enfrenta desafios como as oclusões temporárias de objetos e os conhecidos "ID switches", onde um objeto detetado muda inadvertidamente de identificação devido a alterações na sua aparência, iluminação ou ângulo de visão. Esses problemas dificultam o rastreamento eficiente em ambientes dinâmicos, especialmente em situações de rápida mudança no cenário [26].

A deteção de objetos é uma etapa essencial no processo de rastreamento em tempo real, pois permite identificar e localizar alvos específicos em cada frame antes de associá-los para criar trajetórias contínuas. Esse processo torna-se ainda mais complexo quando se lida com objetos pequenos, que apresentam menos características visuais e são facilmente confundidos com ruído de fundo, o que impacta negativamente a precisão do tracking [27]. Estratégias avançadas, através do uso de algoritmos de aprendizagem profunda, como o *You Only Look Once* (YOLO) [28] e o *Single Shot Multibox Detector* (SSD), têm sido aplicadas para superar esses desafios, tornando o sistema mais robusto para cenários com variabilidade de iluminação, oclusões e mudanças rápidas no ambiente. Além de serem reconhecidos pela sua eficiência na deteção de objetos, alguns desses algoritmos foram expandidos para suportar o MOT,

integrando funcionalidades que permitem identificar e seguir vários alvos ao longo do tempo, como é o caso do YOLO [29].

Para sistemas de deteção e rastreamento em tempo real é habitual segmentar o fluxo de dados em três fases principais, como ilustrado na Figura 2.4.

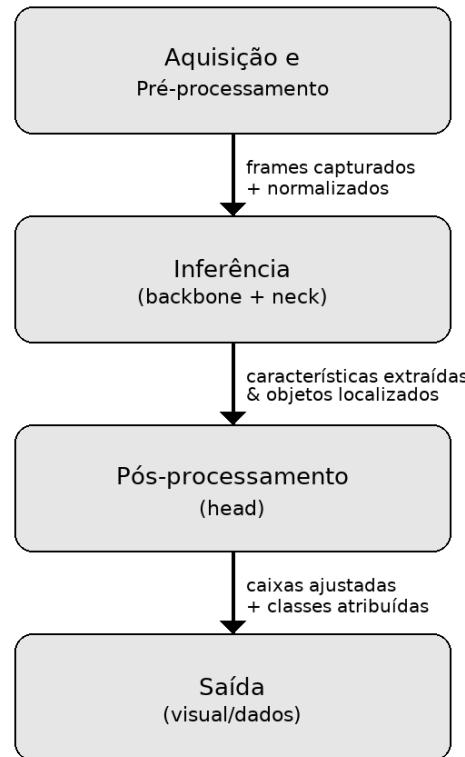


Figura 2.4: Fluxo de processamento de um sistema de deteção e rastreamento em tempo real (elaboração própria).

1. **Aquisição e pré-processamento:** captura de *frames* e operações como redimensionamento e normalização;
2. **Inferência:** a rede neural processa cada frame para extrair características (*backbone*), fundi-las em várias escalas (*neck*) e localizar objetos.
3. **Pós-processamento:** o *head* ajusta as caixas detetadas, classifica cada objeto e prepara a saída visual ou de dados.

Oh et al. demonstrou que, num sistema de CPU puro, como o HEFESTO, a fase de inferência pode corresponder a cerca de 63 % do tempo total de processamento, sendo o principal gargalo para alcançar execução em tempo real. Medir e otimizar a distribuição temporal entre estas etapas (por exemplo, através de paralelismo de tarefas) é, portanto, crucial para garantir *throughput* elevado sem sacrificar a latência [30].

Em ambientes de autoatendimento a integração de deteção de objetos e rastreamento em tempo real, como representado na figura 2.3, é uma ferramenta útil para implementar um sistema adicional de

segurança, ao permitir a identificação de alvos específicos em cada frame e a sua associação a trajetórias contínuas de forma eficiente.

## 2.7 Análise comportamental

A análise comportamental tem-se mostrado essencial no reforço da segurança através da visão computacional [31], uma vez que possibilita identificar padrões de interação normais e, em contraste, comportamentos que se afastam do uso esperado. A Figura 2.5 exemplifica estes desvios, que podem estar associados a situações de violência dirigidas ao utilizador, como agressões físicas ou tentativas de coação durante a utilização da máquina, mas também a riscos de fraude ou vandalismo [32]. Em máquinas de autoatendimento, onde a interação entre o utilizador e a máquina ocorre de forma autónoma e frequentemente sem supervisão direta, estas vulnerabilidades tornam os equipamentos alvos atrativos para atividades ilícitas. Nesses contextos, a monitorização automatizada das interações assegura um acompanhamento contínuo e a deteção precoce de comportamentos de risco, mesmo sem necessidade de intervenção humana [4].



Figura 2.5: Deteção automatizada de ação anómala, com as caixas verdes a representar o *ground truth* e as caixas vermelhas as previsões do sistema. [4].

Uma das vantagens mais significativas da visão computacional aplicada à análise comportamental está na sua capacidade de ir além da identificação de eventos isolados. Assim pode oferecer uma análise mais ampla das sequências de ações, permitindo detetar padrões comportamentais associados a atividades suspeitas. Por exemplo, enquanto um utilizador legítimo interage com a máquina de autoatendimento de forma direta e sem hesitações, comportamentos como rondar repetidamente a máquina, observar o ambiente de forma excessiva, ou manter-se próximo por períodos prolongados sem interação concreta podem ser indicadores de potenciais tentativas de assalto ou vandalismo. Essas sequências de ações, quando capturadas e analisadas em tempo real, oferecem a capacidade de alertar para situações que requerem intervenção imediata [4].

Para além da análise de sequências de ações, um dos métodos que tem ganho relevância na área da visão computacional é a estimativa de pose. Esta técnica permite extrair, a partir de imagens ou vídeo, a posição relativa das articulações do corpo humano, representando o indivíduo como um conjunto de

pontos-chave e ligações. Através desta representação torna-se possível descrever de forma objetiva a postura e os movimentos, fornecendo indicadores úteis para a identificação de situações de risco. Uma das suas principais vantagens é não depender do contexto visual, o que possibilita a deteção de comportamentos violentos ou suspeitos mesmo em ambientes complexos ou com fundos ruidosos [5]. O processo geral desta abordagem encontra-se representado na Figura 2.6.

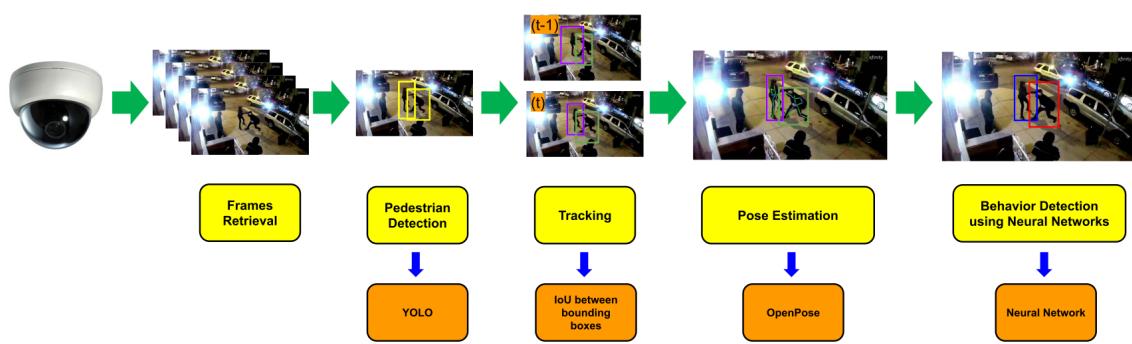


Figura 2.6: Visão geral do processo de deteção de comportamentos violentos baseado em estimativa de pose. Adaptado de [5].

No contexto das máquinas de autoatendimento, a pode ajudar a identificar situações de risco ou coação. Movimentos como murros, pontapés ou tentativas de obstruir a câmara apresentam padrões distintos das interações legítimas, como introduzir o PIN ou recolher numerário. A análise em tempo real destas variações, quando integrada em modelos de aprendizagem automática, fornece informação adicional sobre o comportamento do utilizador, tornando possível acionar alertas imediatos perante gestos que indiciem agressão ou vandalismo. Como ilustrado na Figura 2.7, este tipo de abordagem permite acompanhar a evolução de uma interação violenta, neste caso uma luta, onde as articulações do corpo são identificadas e avaliadas em diferentes instantes. Em cada momento, o sistema atribui probabilidades a comportamentos normais (N) e violentos (V), possibilitando reconhecer indícios de agressão com base nos padrões corporais observados [5].

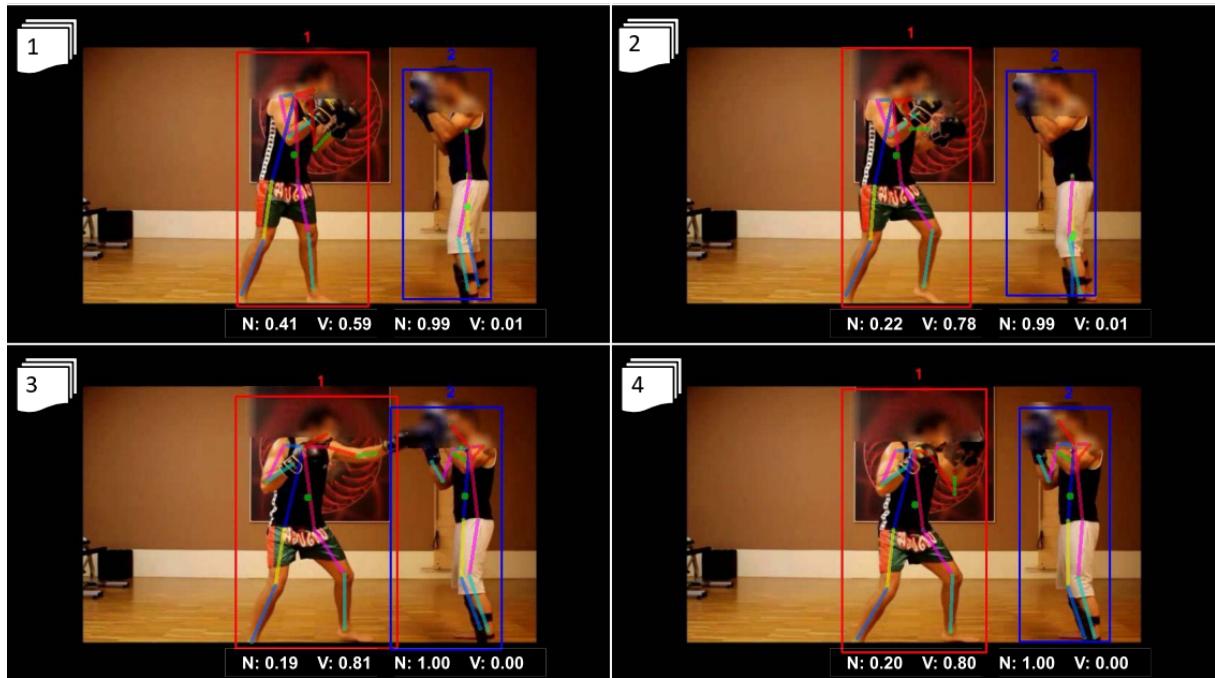


Figura 2.7: Exemplo da aplicação da estimativa de pose na análise comportamental. Adaptado de [5].

Um desafio adicional na análise comportamental está relacionado com a deteção de ações inesperadas, que não correspondem a interações previamente definidas como suspeitas. Para lidar com esse problema, têm sido utilizadas técnicas de aprendizagem supervisionada fraca, que permitem identificar comportamentos anómalos sem a necessidade de especificar todas as possibilidades de ações ilícitas. Uma das principais vantagens desta abordagem consiste em dispensar a anotação detalhada de cada movimento individual, sendo suficiente atribuir um rótulo global ao vídeo, classificando-o como normal ou anómalo, ficando a cargo do modelo a identificação autónoma dos segmentos relevantes. Estudos como o de Sultani et al. demonstraram a eficácia desta estratégia em contextos reais de vigilância, evidenciando que é possível detetar comportamentos imprevistos e gerar respostas rápidas a situações de risco sem recorrer a anotações *frame a frame* [33, 34, 35].



# Capítulo 3

## Trabalho relacionado

### 3.1 Abordagens Baseadas em Redes Neuronais para Detecção de Objetos

#### 3.1.1 Limitações dos Métodos Tradicionais de Visão Computacional

A deteção automática de objetos perigosos, como armas brancas e de fogo, é essencial para garantir a segurança em sistemas automatizados, especialmente em contextos particulares como em máquinas de autoatendimento, onde a interação ocorre sem supervisão direta e a capacidade de resposta rápida é crucial para prevenir incidentes. Entre as abordagens tradicionais de visão computacional, destaca-se o modelo *Bag of Visual Words* (BoVW), que foi amplamente utilizado na identificação de objetos em imagens estáticas, sobretudo em ambientes controlados como a triagem de segurança em aeroportos [36]. Este método, baseado na extração e quantização de descritores locais, demonstrou bons resultados quando as condições de iluminação, ângulo de visão e disposição dos objetos são estáveis e previsíveis. No entanto, esse tipo de abordagem mostra-se insuficiente em cenários mais dinâmicos e desestruturados, como as máquinas de autoatendimento. Nestes contextos, os objetos podem surgir parcialmente ocultos, em movimento rápido, com variações acentuadas de iluminação ou até sobrepostos a outros elementos. Tais condições desafiam a robustez dos métodos tradicionais, comprometendo tanto a taxa de deteção como o tempo de resposta. [37].

#### 3.1.2 Evolução das Arquiteturas com Redes Neuronais

Os primeiros avanços relevantes na deteção baseada em redes neurais surgiram com o modelo *Region-based CNN* (R-CNN), que introduziu o conceito de deteção por regiões. Este modelo começa por gerar múltiplas propostas de regiões na imagem (potenciais localizações de objetos), e aplica uma rede convolucional separadamente a cada uma delas para extrair características, que são depois classificadas individualmente. A Figura 3.1 ilustra este processo, onde se destaca a divisão explícita entre as fases de extração de regiões, processamento por CNN e posterior a classificação.

## RCNN

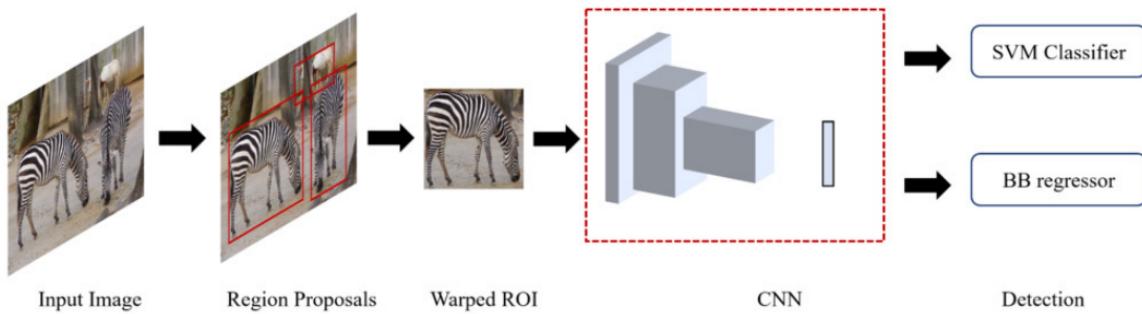


Figura 3.1: Arquitetura do modelo R-CNN

Apesar de ter melhorado substancialmente a precisão face a abordagens anteriores, o R-CNN é computacionalmente ineficiente, dado que requer o processamento independente de centenas de regiões por imagem. A evolução natural deste modelo deu origem ao *Fast R-CNN*, que passou a extrair um mapa de características global da imagem inteira, usando esse mapa para avaliar todas as regiões de interesse. Com isso, reduziu-se significativamente o tempo de computação, mantendo bons níveis de precisão. Posteriormente, o *Faster R-CNN* [38] integrou uma Rede de Propostas de Regiões (RPN) diretamente na arquitetura, eliminando a dependência de algoritmos externos para gerar as propostas iniciais, como ilustrado na Figura 3.2.

## Faster RCNN

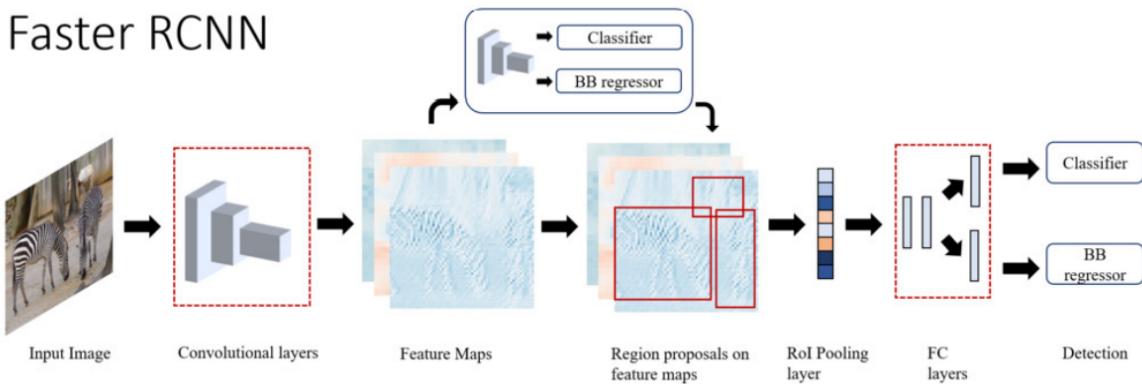


Figura 3.2: Arquitetura do modelo Faster R-CNN

No entanto, apesar destas melhorias, ambos os modelos continuam a seguir uma abordagem em duas etapas, que separa a localização da classificação dos objetos, o que limita a sua aplicabilidade em sistemas com restrições de tempo real [39].

### 3.1.3 Modelos de Uma Etapa para Detecção em Tempo Real

Como alternativa aos modelos de duas etapas, surgiram as arquiteturas de detecção de uma só etapa, como o SSD e o YOLO. Estes modelos realizam simultaneamente a localização e a classificação dos

objetos diretamente a partir da imagem completa, sem necessidade de uma fase separada de propostas de regiões. Esta abordagem unificada permite uma deteção significativamente mais rápida, tornando estas redes especialmente adequadas para aplicações em tempo real, como é o caso dos sistemas de segurança em máquinas de autoatendimento [40, 28, 41].

A Figura 3.3 ilustra o funcionamento genérico de um modelo do tipo YOLO, onde a imagem de entrada é processada por uma rede convolucional que, numa única passagem, extrai as características, classifica os objetos presentes e estima simultaneamente as suas localizações através da regressão das caixas delimitadoras.

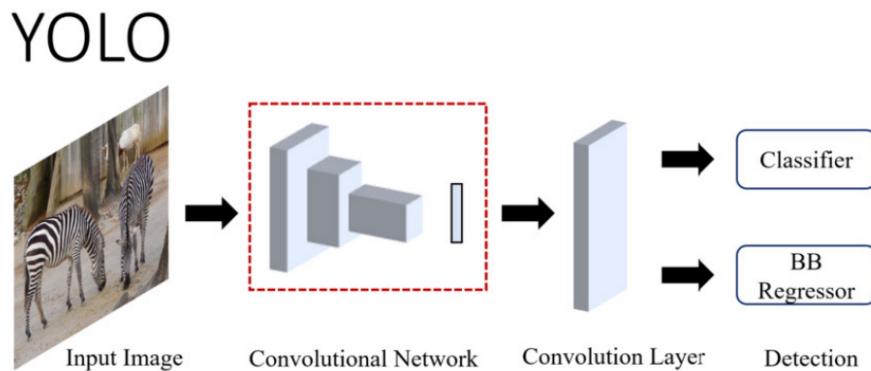


Figura 3.3: Arquitetura genérica do modelo YOLO.

### 3.1.4 Distinção entre Modelos de Detecção e de Classificação

Em visão computacional, é importante distinguir entre modelos concebidos exclusivamente para classificação de imagens e aqueles orientados para deteção de objetos. Modelos de classificação, como ResNet [42], MobileNet [43] ou EfficientNet [44], recebem uma imagem como entrada e produzem uma predição global, indicando a presença de uma ou mais categorias de interesse, sem identificar a localização exata dos objetos. Do ponto de vista arquitetural, estes modelos terminam geralmente com camadas totalmente conectadas aplicadas ao vetor de características extraído pela rede convolucional.

Por contraste, os modelos de deteção, como SSD [40], YOLO [45] ou Faster R-CNN [38], realizam simultaneamente a regressão das caixas delimitadoras e a classificação individual de cada objeto identificado. Estes modelos analisam várias regiões da imagem e geram, para cada uma, uma caixa com coordenadas e uma previsão de classe. Ao contrário dos classificadores, não dependem de camadas totalmente conectadas aplicadas à imagem inteira.

Esta distinção entre classificação e deteção está ilustrada na Figura 3.4, onde se observa que o classificador apenas indica a presença de uma ameaça, enquanto o detetor a identifica e localiza explicitamente na imagem.

Deste modo, os classificadores podem constituir uma mais-valia em sistemas de segurança, atuando como componente adicional que permite detetar a presença de conteúdo suspeito, mesmo sem fornecer localização. Esta capacidade pode complementar os modelos de deteção, contribuindo para decisões mais rápidas ou reforço da análise em contextos críticos.



Figura 3.4: Comparação entre classificação (esquerda) e deteção (direita).

### 3.1.5 Arquiteturas com Transformadores

Mais recentemente, modelos baseados em Transformadores começaram a ser aplicados à tarefa de deteção de objetos, oferecendo uma alternativa inovadora às arquiteturas tradicionais baseadas em CNNs. Originalmente desenvolvidos para o processamento de linguagem natural, os Transformadores destacam-se pela sua capacidade de modelar relações entre elementos de uma sequência através de mecanismos de atenção, que permitem ao modelo focar-se nas partes mais relevantes da entrada, independentemente da sua posição. Esta capacidade mostrou-se também útil em visão computacional, pois permite ao modelo compreender como diferentes partes da imagem se relacionam entre si. Um dos exemplos desta abordagem é o Transformador de Detecção (DETR), que propõe um sistema unificado onde a localização e a classificação dos objetos são tratadas conjuntamente, sem depender de fases intermédias como a de gerar propostas de regiões ou a Supressão Não-Máxima (NMS). O modelo utiliza uma CNN para extrair características da imagem e, em seguida, aplica um Transformador que aprende as relações espaciais entre objetos e produz diretamente as caixas delimitadoras e as classes correspondentes. O DETR apresenta limitações em aplicações com exigência de tempo real, devido à sua complexidade computacional e ao tempo de convergência mais prolongado. Além disso, o modelo mostra menor eficácia na deteção de objetos pequenos, o que representa um obstáculo adicional em cenários com elevada densidade ou múltiplos elementos visuais sobrepostos [39]. Importa ainda referir que, embora os modelos baseados em Transformadores representem uma evolução relevante no domínio da visão computacional, a sua aplicação prática continua condicionada pela necessidade de recursos computacionais elevados. Tendo em conta que este trabalho tem como foco a integração em máquinas de autoatendimento que, na sua maioria, apresentam capacidades computacionais limitadas e não dispõem de GPU dedicadas. Por esta razão, apesar de serem aqui abordadas no contexto do estado da arte, estas arquiteturas não serão consideradas como solução prática a implementar.

## 3.2 Modelos de Detecção em Tempo Real

### 3.2.1 Limitações dos Modelos com NMS

Recentes avanços têm mostrado que detetores baseados em CNNs, como as diferentes versões do YOLO até à versão 9, se destacam pela sua capacidade de conciliar desempenho e eficiência [28, 45]. O YOLO, enquanto modelo de uma só etapa (*single-stage*), realiza a deteção através de uma passagem direta pela rede (*forward process*), seguida de um pós-processamento com NMS. Apesar de esta abordagem oferecer bons resultados em termos de desempenho, o uso de NMS para remover predições redundantes pode afetar a velocidade de inferência e introduzir dependências em relação a hiperparâmetros específicos. No entanto, esse impacto tende a ser pouco significativo em implementações otimizadas, dependendo do contexto da aplicação.

### 3.2.2 Arquiteturas *End-to-End*: DETR e RT-DETR

Como alternativa ao uso do NMS, surgiram arquiteturas *end-to-end* como o DETR e a sua variante otimizada RT-DETR [46], que eliminam a necessidade de pós-processamento ao integrarem a deteção e a classificação num único passo. No entanto, apesar de resolverem esse problema, estes modelos exigem mais recursos computacionais, o que dificulta o equilíbrio entre precisão e velocidade, especialmente em aplicações com restrições de tempo real e recursos limitados, como dispositivos sem GPU, tal como foi abordado na Secção 3.1.5.

### 3.2.3 Avanços com YOLOv10

Neste contexto, o surgimento do YOLOv10 [29], representa um avanço significativo face às versões anteriores da arquitetura YOLO (como ilustrado na Figura 3.5), especialmente no que diz respeito à dependência do NMS e ao impacto deste na latência de inferência [6].

Com a introdução do conceito de "*dual assignments*", o YOLOv10 funciona de forma totalmente independente do NMS, tanto na fase de treino como na de inferência. Esta mudança permite uma deteção mais fluida, com menor latência, uma vez que o próprio modelo aprende a identificar as caixas mais relevantes sem recorrer a regras externas ou pós-processamentos adicionais. Esta estratégia elimina a necessidade de ajuste de hiperparâmetros associados ao NMS, o que torna o modelo mais robusto e adaptável a diferentes contextos [6].

Para além disso, o YOLOv10 foi projetado com foco na eficiência computacional, recorrendo a técnicas como *downsampling* desacoplado e uma *classification head* leve, o que permite realizar inferências rápidas sem comprometer a precisão. Estas melhorias posicionam o YOLOv10 como uma das soluções mais eficazes para aplicações em tempo real que exigem simultaneamente alta precisão e baixa latência.

### 3.2.4 Comparaçāo de Desempenho

Em comparação com o RT-DETR, o YOLOv10 destaca-se pela sua maior velocidade de inferência, sendo até 1,8 vezes mais rápido, ao mesmo tempo que mantém uma precisão competitiva [6].

A análise da Figura 3.5 mostra que, no gráfico à esquerda, o YOLOv10 alcança uma precisão significativamente superior com menor latência, o que o posiciona como uma solução altamente eficiente

para aplicações em tempo real. Esta métrica, baseada Média da Precisão (mAP), foi obtida utilizando a base de dados Objetos Comuns em Contexto (COCO) [47], um dos principais *benchmarks* utilizados na avaliação de modelos de deteção de objetos. Já o gráfico à direita evidencia o equilíbrio entre a precisão e o número de parâmetros, ou seja, a quantidade de pesos aprendidos pelo modelo durante o treino. Esse valor influencia diretamente a complexidade e a eficiência computacional, sendo um fator determinante no desempenho em tempo real. O YOLOv10 destaca-se por alcançar elevada precisão mesmo com modelos mais leves, demonstrando maior eficiência.

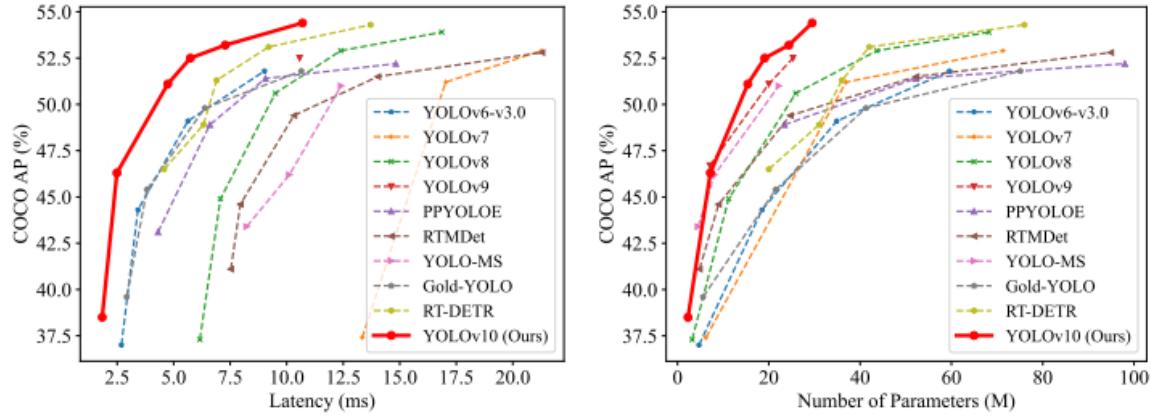


Figura 3.5: Comparação de métricas entre YOLOv10 e outras arquiteturas, incluindo latência e número de parâmetros. Gráficos retirados de [6].

### 3.3 Abordagens para Rastreamento de Múltiplos Objetos

Dando continuidade à discussão da Secção 2.6, esta secção explora as abordagens mais utilizadas na implementação do MOT. Esta tarefa é fundamental em várias áreas da visão computacional, como videovigilância, controlo de tráfego e análise de interações humanas [48]. No contexto das máquinas de autoatendimento, o MOT torna-se relevante por permitir o acompanhamento contínuo de objetos que interagem com o equipamento, contribuindo para a deteção de comportamentos suspeitos [26, 27]. Uma das estratégias mais comuns para realizar MOT é o *Tracking-by-Detection* (TbD), que combina detectores de objetos, como o YOLO, com algoritmos de rastreamento como o *Simple Online and Realtime Tracking* (SORT) [49] e apesar da sua simplicidade e eficiência, esta abordagem enfrenta limitações em cenários mais complexos, como já referido na Secção 2.6. Por outro lado, existem abordagens mais recentes, como o *Tracking-by-Attention* com Transformadores, que usam mecanismos de atenção para seguir objetos entre imagens de forma direta [50]. Este tipo de atenção permite ao modelo concentrar-se nas partes mais importantes da imagem e aprender sozinho a ligar os objetos ao longo do tempo. Assim, deixa de ser necessário recorrer a métodos externos de associação, como o uso do Interseção sobre União (IoU) para emparelhar deteções. Apesar de mais avançadas, estas soluções exigem muito mais recursos computacionais e, por isso, não serão consideradas neste trabalho, que se foca em soluções leves e adequadas ao uso em tempo real [51].

Entre os rastreadores em MOT, destacam-se abordagens em tempo real com diferentes características. O SORT é simples e eficiente [49], baseando-se em modelos de movimento linear e associação por IoU,

o que o torna adequado para cenários com deteções bem definidas. No entanto, apresenta fragilidades em situações com oclusões prolongadas ou objetos visualmente semelhantes. O *BYTETrack* melhora o *SORT* ao considerar deteções de baixa confiança no processo de associação, mantendo um elevado desempenho em tempo real [52]. É frequentemente utilizado com detetores como o *YOLO* [29], que também suporta o *BoT-SORT* [53], rastreador que integra informação de aparência com técnicas de associação mais robustas, alcançando maior precisão segundo a métrica Precisão de Rastreamento de Ordem Superior (HOTA) em cenários complexos.

O gráfico da Figura 3.6 destaca o estado da arte em rastreamento de múltiplos objetos, demonstrando como rastreadores como *SMILEtrack*, *BYTETrack* e *BoT-SORT* equilibram a HOTA e os *Frames per Second* (FPS), sendo indicados para aplicações práticas em tempo real. No entanto, importa referir que muitos destes métodos são avaliados em contexto *offline*, com acesso a frames futuros, o que pode não refletir plenamente as exigências do rastreamento *online* necessário em cenários reais de vigilância [7].

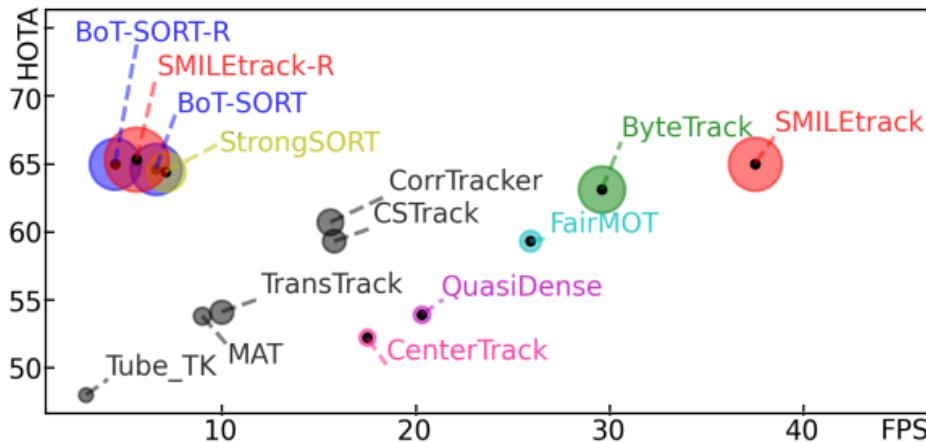


Figura 3.6: Comparação da precisão entre rastreadores do estado da arte. Gráfico retirado de [7].

### 3.4 Fluxo para Detecção e Prevenção de Crimes em ATMs

A Figura 3.7 apresenta um fluxo funcional descrito na literatura para sistemas de segurança em ATMs, baseado na integração de módulos de visão computacional e reconhecimento de eventos [8]. Este fluxo, já existente, serviu como uma das principais inspirações para a conceção do sistema desenvolvido neste trabalho. O funcionamento assenta na captação contínua de imagens por uma câmara embutida no terminal ATM, cujos *frames* são processados.

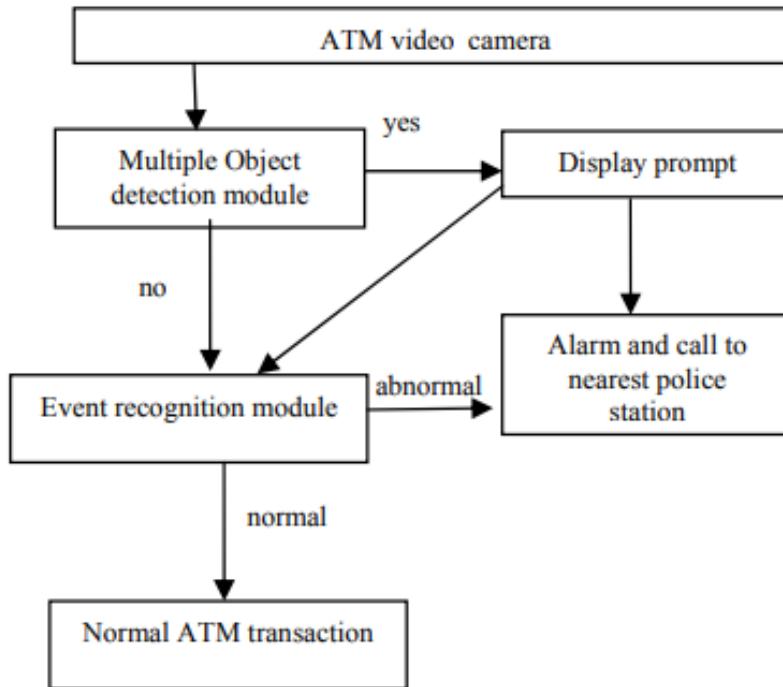


Figura 3.7: Fluxo funcional de um sistema de segurança para ATMs baseado em deteção de objetos e reconhecimento de atividades suspeitas. [8]

Inicialmente, as imagens são enviadas para um módulo de deteção de múltiplos objetos, responsável por identificar entidades presentes na cena, como pessoas adicionais ou objetos potencialmente perigosos. Sempre que este módulo deteta a presença de mais do que uma pessoa na área do ATM, é acionado um *display prompt* no ecrã, solicitando ao utilizador que confirme se se encontra em segurança e se autoriza a continuidade da operação. Caso o utilizador valide a situação, o sistema prossegue para o módulo de reconhecimento de atividades. Caso contrário, é criado um alerta imediato e acionado o mecanismo de resposta definido para situações de risco.

O segundo módulo tem como objetivo analisar de forma autónoma os padrões de interação captados pela câmara, de modo a identificar comportamentos potencialmente anómalos, como gestos agressivos ou situações de coação. Mesmo nos casos em que o utilizador tenha validado a presença de múltiplas pessoas, o sistema continua a proceder a esta verificação. Sempre que o módulo de reconhecimento de atividades deteta indícios de comportamento suspeito, o sistema aciona de imediato um alarme e notifica as autoridades competentes. Apenas quando não é identificada qualquer anomalia, seja pela resposta do utilizador ao aviso inicial, seja pela análise automática de atividades, a transação no ATM prossegue de forma normal.

Apesar de representar uma proposta interessante, esta abordagem apresenta riscos práticos significativos. Num cenário real, um assaltante poderia facilmente coagir a vítima a selecionar a opção negativa ou validar falsamente a presença de múltiplas pessoas, ocultando a situação de perigo. Além disso, exigir uma resposta explícita do utilizador em momentos de tensão pode aumentar o risco para a sua integridade física, o que limita a aplicabilidade desta solução em contextos de criminalidade violenta.

Este fluxo constitui, assim, um exemplo de abordagem híbrida que combina intervenção do utilizador e deteção automatizada, cuja análise contribuiu para a reflexão que orientou o desenvolvimento da

proposta desta dissertação.

### 3.5 Reconhecimento de Emoções Faciais

O Reconhecimento de Emoções Faciais (FER) é uma subárea relevante da visão computacional, com aplicações em domínios como a interação humano-computador, a saúde e a segurança. Esta tarefa consiste na identificação automática de estados emocionais, como felicidade, medo ou raiva, através da análise de expressões faciais captadas em imagens ou sequências de vídeo. Em contextos de segurança, como em sistemas de vigilância automatizada, o FER tem sido explorado como um componente complementar para a deteção de comportamentos anômalos. Emoções como stress ou medo podem constituir indicadores indiretos de situações de coação, tornando o FER útil em ambientes sensíveis e não supervisionados, como as máquinas de autoatendimento [Q].

As abordagens iniciais baseavam-se em técnicas de extração manual de características, como o Histograma de Gradientes Orientados (HOG) [54], ou em descritores geométricos obtidos a partir de rácios entre distâncias faciais [55]. Embora eficazes em ambientes controlados, essas técnicas revelam fragilidades significativas perante variações de pose, iluminação ou oclusões.

Com a introdução da aprendizagem profunda, redes convolucionais começaram a dominar esta tarefa, devido à sua capacidade de aprender representações discriminativas diretamente dos dados. No entanto, mesmo modelos modernos baseados em CNNs enfrentam dificuldades quando expostos a imagens degradadas por iluminação desigual, expressões sutis ou posições faciais fora do eixo frontal. Estas limitações afetam a generalização dos modelos e comprometem a precisão em ambientes não controlados.

Como resposta a estas limitações, Vignesh et al. propuseram uma arquitetura baseada no VGG-19 com blocos de segmentação facial que permite isolar regiões expressivas da face, como os olhos e a boca, antes da extração de características [56]. Esta segmentação orientada permite ao modelo focar-se nas áreas mais relevantes para a expressão emocional, reduzindo o impacto de ruído de fundo ou partes irrelevantes da imagem. Os resultados obtidos demonstraram melhorias significativas na robustez e precisão da classificação, especialmente em imagens com variações de pose e iluminação.

Um pipeline típico de FER envolve múltiplas etapas, desde a deteção da face na imagem até à classificação final da emoção. Como ilustrado na Figura 3.8, o processo inicia-se com a localização e alinhamento geométrico da região facial, com base em pontos-chave como os olhos e a boca. A imagem normalizada é então processada por redes convolucionais que extraem representações discriminativas. Finalmente, estas são classificadas numa das categorias emocionais predefinidas através de camadas totalmente conectadas ou classificadores dedicados, como Máquinas de Vetores de Suporte (SVM) [57], aplicados sobre os vetores extraídos [9]. Este pipeline permite lidar eficazmente com variações significativas na aparência facial.

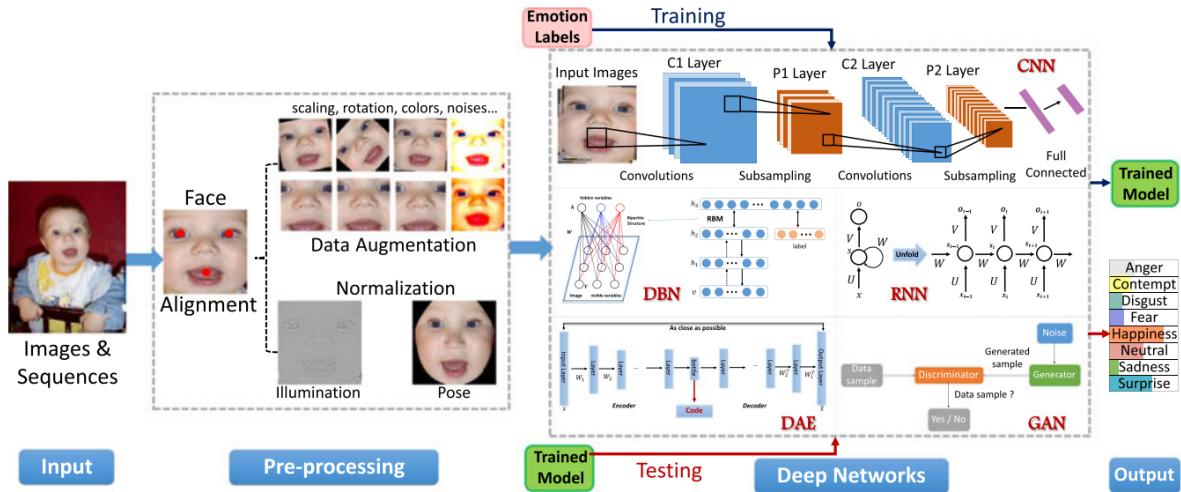


Figura 3.8: Pipeline geral de um sistema de reconhecimento de emoções faciais [9].

# Capítulo 4

## Metodologia

### 4.1 Exploração Inicial das Tecnologias de Visão Computacional

Durante os primeiros meses do estágio na empresa INM, foi iniciado o desenvolvimento de um sistema de segurança com base em visão computacional, com o foco de garantir uma integração fluida com o projeto HEFESTO. Antes de definir a arquitetura e os componentes do sistema, foi conduzida uma fase de exploração com o propósito de adquirir conhecimentos técnicos fundamentais e avaliar ferramentas e bibliotecas relevantes para esta área.

Nesta fase inicial, foram analisadas algumas das principais tecnologias da visão computacional, com o intuito de compreender as suas funcionalidades, limitações e potencial de aplicação em cenários de segurança. A biblioteca OpenCV [58] foi utilizada para entender operações básicas como a captura e processamento de vídeo, conversão de frames e aplicação de transformações simples. Estes testes ajudaram a consolidar noções fundamentais, como a importância do desempenho em tempo real e a influência das condições de iluminação na qualidade da análise.

Foram também estudados modelos de deteção de objetos baseados em redes neurais convolucionais, em particular o YOLO [6], com o objetivo de compreender o seu funcionamento e avaliar a sua adequação ao contexto do projeto. Ainda que nesta fase não tenha sido realizada uma validação aprofundada, foi possível perceber a capacidade do modelo em identificar e localizar objetos relevantes em imagens.

Em paralelo, foram analisadas soluções de reconhecimento facial e análise emocional, nomeadamente a biblioteca DeepFace [59], com o objetivo de perceber de que forma a identificação de expressões faciais poderia ser usada para detetar sinais de stress, medo ou outras emoções associadas a situações de risco.

Esta etapa exploratória permitiu estabelecer uma base sólida para as decisões técnicas que se seguiram, nomeadamente na seleção das tecnologias a integrar no sistema final e na definição dos requisitos funcionais para cada componente. O recurso à câmara integrada na máquina HEFESTO durante alguns dos testes iniciais contribuiu também para um melhor enquadramento das necessidades do sistema face ao ambiente real onde seria implementado.

## 4.2 Construção do Sistema de Segurança

### 4.2.1 Objetivos Técnicos e Critérios de Conceção

Deu-se início à projeção de um sistema para operar de forma independente, expansível e capaz de garantir a sua integração com uma máquina de autoatendimento como o HEFESTO. Pretendeu-se desenvolver uma solução simples de implementar, eficiente e sem exigir alterações significativas na infraestrutura existente. O sistema foi pensado para ser suficientemente leve para funcionar localmente, mantendo também uma estrutura modular e flexível, preparada para evoluções futuras.

Com base nas características do ambiente e nas limitações identificadas, foram definidos os seguintes critérios técnicos de conceção:

- **Execução local otimizada:** preparado para funcionar diretamente no hardware do HEFESTO, de forma autónoma e eficiente no uso dos recursos.
- **Simplicidade tecnológica:** evitar ferramentas pesadas, como contentores Docker, devido às limitações de desempenho da máquina.
- **Independência tecnológica:** o sistema não deveria ficar limitado a tecnologias específicas, de modo a garantir maior portabilidade e facilidade de adaptação no futuro.
- **Arquitetura modular:** cada componente deve ser independente, facilitando manutenção, substituições e atualizações sem afetar o restante do sistema.
- **Compatibilidade:** assegurar a integração com o HEFESTO sem necessidade de alterar a aplicação nativa da máquina de autoatendimento.

### 4.2.2 Arquitetura Modular e Estrutura Plug-and-Play

O desenvolvimento inicial centrou-se na definição da arquitetura do sistema, concebida para identificar e responder a ameaças em tempo real.

A arquitetura foi desenhada segundo um princípio modular, garantindo a separação clara entre componentes e permitindo que os modelos de computação visual sejam integrados ou substituídos sem impactar o restante sistema. Esta lógica *plug-and-play* assegura extensibilidade, manutenção simplificada e evolução contínua da solução, desde que cada novo modelo seja disponibilizado como um serviço com uma Interface de Programação de Aplicações (API) compatível com o padrão de comunicação pré-definido pelo sistema. Com este mecanismo, o sistema pode ser instalado rapidamente em diferentes máquinas de autoatendimento e integrar quase de imediato novos modelos de computação visual.

A Figura 4.1 apresenta a visão geral da arquitetura, evidenciando os componentes internos do Sistema de Segurança e o módulo externo (Modelos de Computação Visual). Evidencia-se ainda a ligação existente (1) entre o controlador e os serviços externos dos modelos, responsáveis por processar os *frames* obtidos e devolver os resultados de inferência.

### Arquitetura e Componentes do Sistema de Segurança

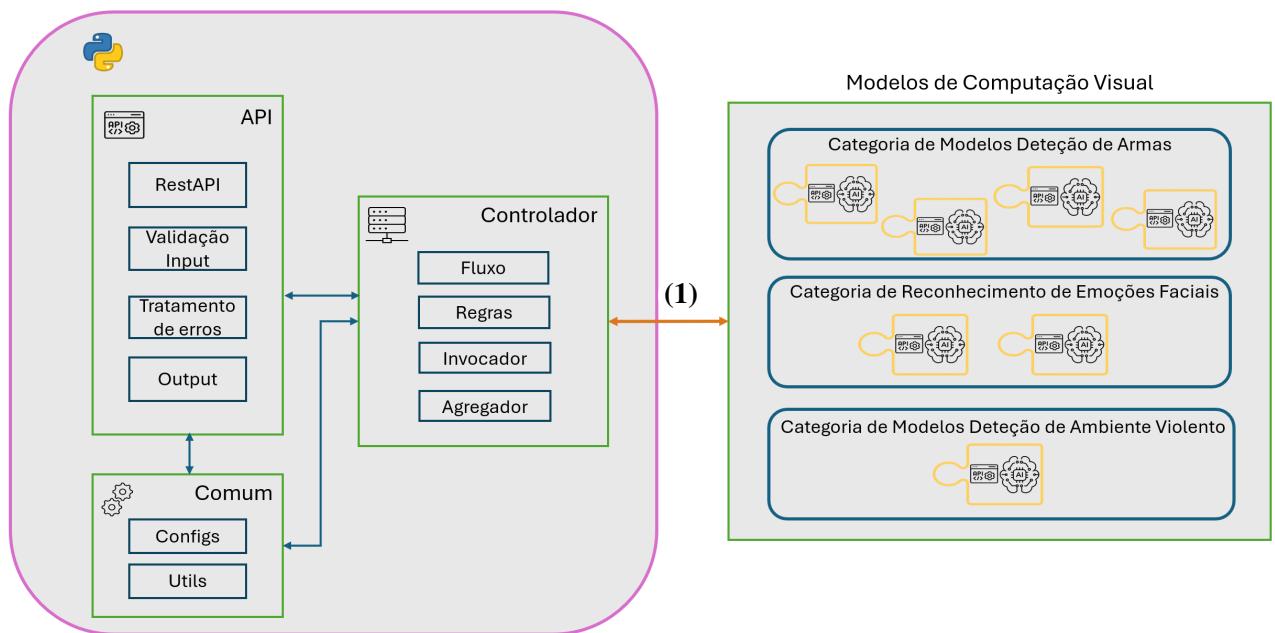


Figura 4.1: Arquitetura geral do Sistema de Segurança, com destaque na ligação assinalada em (1) para os serviços dos modelos de computação visual.

#### 4.2.3 Descrição da Arquitetura e dos seus Componentes

Após o desenho geral e das definições técnicas iniciais, tornou-se necessário detalhar cada componente funcional do sistema, de modo a assegurar uma implementação alinhada com os objetivos definidos. A arquitetura, já ilustrada na Figura 4.1, está organizada em três módulos principais: API, módulo comum e controlador, aos quais se liga um módulo externo que representa os serviços associados aos modelos de computação visual utilizados. Os módulos principais foram desenvolvidos em *Python*, devido ao seu suporte robusto a bibliotecas de computação visual e inteligência artificial, consideradas essenciais para este projeto [60].

##### API

A API constitui o ponto de entrada e de saída do sistema de segurança, implementada utilizando a framework FastAPI [61]. É responsável por intermediar a comunicação máquina-a-máquina com a aplicação cliente já existente na máquina de autoatendimento. Essa aplicação cliente, que corre de forma independente, é a responsável por captar os *frames* da câmara integrada na própria máquina e enviá-los para a API através de requisições Protocolo de Transferência de Hipertexto (HTTP) segundo o serviço RESTful.

A cada requisição enviada pela aplicação cliente, contendo um *frame* captado pela câmara, a API procede à decodificação da imagem e a uma verificação de qualidade com base na luminosidade, convertendo o *frame* para escala de cinzentos de modo a considerar apenas a intensidade luminosa de cada pixel e calculando o histograma de níveis de cinzento para analisar a distribuição global da luminosidade, sendo o *frame* descartado quando mais de 90% dos pixels apresentam intensidades próximas do preto, indicando uma possível obstrução da câmara, ou próximas do branco, indicando excesso de luz, sendo

apenas encaminhados para processamento posterior os *frames* que apresentam condições adequadas de iluminação, otimizando assim o uso dos recursos computacionais.

Para garantir um desempenho eficiente e assegurar que a análise seja realizada com dados recentes, a API mantém um *buffer* temporário de *frames*, no qual apenas os mais atuais são preservados durante cada ciclo de processamento. Sempre que o limite do *buffer* é atingido, os *frames* mais antigos são automaticamente descartados, seguindo uma política *First-In, First-Out* (FIFO). Em paralelo, o sistema garante que todos os pedidos recebem resposta, seja com resultados da análise ou com a indicação de descarte, evitando bloqueios na comunicação. Em segundo plano, a API procede à seleção periódica de um conjunto de *frames* representativos, de forma a evitar a análise de imagens demasiado semelhantes. Esses *frames* são depois agrupados e enviados em lote para o controlador, onde ocorre a análise.

Para além de receber os *frames*, a API também devolve à aplicação cliente os resultados da análise, incluindo o estado do alarme, o tempo de processamento e as principais deteções identificadas. Inclui ainda mecanismos de tolerância a falhas, assegurando que mesmo em situações anómalas, como atrasos excessivos ou erros, o cliente obtém sempre uma resposta adequada.

Apesar de eficaz esta abordagem pode apresentar algumas limitações, uma vez que o envio *frame a frame* gera vários pedidos por segundo, o que pode acrescentar algum overhead embora pouco significativo em contexto local, e a amostragem temporal implica o descarte de certas imagens, podendo perder-se eventos muito breves. A dimensão do *buffer* também influencia o desempenho, exigindo ajuste consoante os FPS da câmara e do tempo de processamento.

## Controlador

O controlador atua como o núcleo lógico do sistema e é responsável por gerir o *pipeline* de processamento dos frames validados provenientes da API. Após a receção destes dados, o controlador coordena todo o fluxo interno, desde a identificação das categorias de inferência ativas até à consolidação dos resultados finais.

Uma das responsabilidades do controlador é a invocação dos modelos externos de computação visual. Para isso, utiliza mecanismos de execução assíncrona que permitem o envio paralelo de requisições HTTP para os serviços de inferência de cada modelo ativo, evitando o bloqueio da execução principal. Cada conjunto de frames é enviado em lote para as APIs específicas dos serviços dos modelos as quais foram desenvolvidas de forma dedicada para possibilitar a integração com o sistema de segurança. Os resultados devolvidos por cada modelo são recolhidos e estruturados para posterior análise de decisão.

Para além de estruturar as respostas recebidas dos modelos, o controlador executa também um processo complementar com base nas áreas das faces detetadas. Essas áreas são fornecidas pelos serviços de reconhecimento facial e utilizadas para calcular o rácio entre a face principal do utilizador e uma eventual segunda face presente no mesmo frame. Se esse rácio ultrapassar o limiar configurado, o controlador regista informação adicional que não altera diretamente o estado do alarme, mas sinaliza a possível presença de um intruso demasiado próximo do utilizador da máquina. Esta informação suplementar fornece contexto útil para a análise de segurança, podendo apoiar decisões futuras.

Após este processamento complementar, o controlador aplica as regras de decisão configuráveis. Inicialmente, são aplicadas regras ao nível de cada categoria individual (por exemplo, regras específicas para deteção de armas ou reconhecimento de expressões faciais), sendo que cada categoria tem modelos

associados. De seguida, realiza-se uma avaliação ao nível global, combinando os resultados das várias categorias com base em limiares ou critérios lógicos específicos. Este mecanismo assegura flexibilidade na definição do que constitui uma situação de alarme com base nos modelos em comunicação.

Caso seja identificado um alarme, o controlador extrai as deteções mais relevantes por categoria, selecionando as que apresentam maior confiança. O resultado final inclui o estado do alarme, o tempo total de processamento e um resumo das principais deteções, sendo então devolvido à API.

Em caso de falha na comunicação com algum dos modelos ou de respostas fora do tempo limite esperado, o controlador regista o incidente e continua o processamento com os dados disponíveis.

### Módulo Comum

O módulo comum agrupa um conjunto de funcionalidades de suporte utilizadas pelos restantes componentes do sistema. Entre as suas principais responsabilidades está a leitura das configurações do sistema, centralizadas num ficheiro `config.json` em Notação de Objeto JavaScript (JSON). Essas configurações são carregadas apenas uma vez e mantidas em memória, ficando disponíveis para acesso eficiente pelos restantes módulos, o que garante consistência e evita redundância no carregamento dos parâmetros.

Este módulo disponibiliza funções auxiliares que permitem consultar, em tempo de execução, as categorias de modelos ativas, os modelos associados a cada categoria, as regras de decisão definidas (por categoria e globais), os limiares e os pesos aplicáveis. Essas funções não implementam as regras de decisão, mas fornecem ao controlador os elementos necessários para a sua execução e para o encaminhamento dos dados para os modelos.

Adicionalmente, o módulo comum centraliza a lógica de registo de logs, que assegura o acompanhamento de operações internas, decisões tomadas e eventuais anomalias durante o funcionamento do sistema. Esta componente é essencial para diagnóstico, manutenção e auditoria do comportamento do sistema.

### Módulo Externo: Modelos de Computação Visual

O módulo externo representa os modelos de computação visual utilizados pelo sistema. Estes modelos não fazem parte da arquitetura interna do Sistema de Segurança e são executados como serviços externos independentes. O módulo externo é responsável apenas por fornecer as inferências solicitadas, comunicando com o sistema através das interfaces específicas implementadas para esse efeito.

### Modelos de Computação Visual em Utilização pelo o Sistema

A escolha dos modelos de computação visual que foram selecionados para comunicar com o Sistema de Segurança resultou de uma fase de pesquisa e experimentação, na qual foram analisadas várias soluções disponíveis em repositórios públicos e plataformas *open-source*. Os principais critérios de seleção incluíram:

- Disponibilidade de modelos pré-treinados com classes relevantes para o contexto do sistema (armas, expressões emocionais de risco, comportamentos violentos).

- Capacidade de execução eficiente em hardware limitado, recorrendo a arquiteturas leves de computação visual que conciliem baixo consumo de recursos com tempos de resposta adequados.
- Facilidade de integração com o sistema, incluindo a possibilidade de expor o modelo através de uma API simples e compatível com a arquitetura definida.

No âmbito do Sistema de Segurança, os modelos de computação visual utilizados encontram-se organizados em três categorias principais, correspondentes aos tipos de análise considerados relevantes para o contexto da aplicação: deteção de armas, reconhecimento de expressões emocionais e deteção de ambiente violento.

Cada categoria agrupa um ou mais modelos dedicados à sua função específica, os quais comunicam com o sistema como serviços independentes. A seguir descrevem-se os modelos em utilização e a respetiva associação às categorias de análise.

### **Categoria: Deteção de armas**

Esta categoria é responsável pela identificação de armas de fogo ou brancas nos *frames* analisados pelo sistema. Para esta função, foram utilizados quatro modelos de computação visual disponibilizados em repositórios públicos, cujos pesos e configuração foram mantidos conforme fornecido pelos respetivos autores. Os modelos foram postos em funcionamento como serviços externos, e o sistema comunica com eles através dos seus endpoints definidos.

- **YOLOv8 (3 instâncias):** três modelos YOLOv8 pré-treinados, utilizados para a deteção de armas de fogo e armas brancas nos frames analisados. Cada modelo foi obtido diretamente de repositórios públicos, como [62], [63] e [64], e cada um foi integrado para operar como um serviço externo. A utilização de múltiplas instâncias do YOLOv8 explora a complementaridade entre modelos treinados com diferentes conjuntos de dados, permitindo abranger variações de classes, escala e condições visuais associadas à deteção de armas. Esta estratégia aumenta a robustez do sistema e reduz a probabilidade de situações de risco envolvendo o uso de armas não serem detetadas.
- **EfficientNet:** modelo de classificação binária utilizado também para fornecer ao sistema informação sobre a presença ou ausência de armas nos *frames*, com base numa abordagem de classificação geral da imagem. A sua utilização comprova a compatibilidade do sistema de segurança com diferentes tipos de modelos de computação visual além do padrão YOLO. [65].

#### *Fluxo de processamento dos modelos:*

**YOLOv8:** os serviços dos modelos recebem os *frames* em lote, enviados pelo controlador através dos respetivos *endpoints*. O fluxo do modelo pode também ser visualizado de forma genérica na Figura 3.3, apresentada na secção de Trabalho Relacionado. Cada modelo realiza as seguintes etapas:

- Redimensionamento de cada imagem do lote para as dimensões esperadas pelo modelo (ex.: 640×640), mantendo a proporção através de *padding* quando necessário.
- Conversão das imagens para o formato de tensores e normalização dos valores de pixel.

- *Forward pass* na rede YOLOv8, em que o lote de frames é processado sequencialmente pela *backbone* (responsável pela extração de características), pela *neck* (que realiza a fusão e enriquecimento das características) e pela *head*. Esta última gera como saída um tensor no formato  $[Batch, N + 5]$ , onde cada previsão corresponde a um vetor do tipo  $[x, y, w, h, \text{obj\_conf}, \text{prob}(\text{classe 0}), \dots, \text{prob}(\text{classe N-1})]$ , contendo as coordenadas da caixa delimitadora, a confiança na presença de um objeto e as probabilidades atribuídas a cada classe.
- Pós-processamento com eliminação de deteções redundantes através de NMS.
- Devolução ao sistema de segurança as classes detetadas, as caixas delimitadoras e os graus de confiança associados.

**EfficientNet:** o serviço do modelo recebe os *frames* em lote através do respetivo endpoint, onde são realizadas as seguintes operações:

- Redimensionamento das imagens para as dimensões de entrada do modelo (ex.:  $224 \times 224$ ).
- Normalização dos valores de pixel para o intervalo  $[0, 1]$  e conversão para tensores.
- *Forward pass* na rede EfficientNet, no qual os dados são processados sequencialmente pelos blocos da rede (por exemplo, camadas convolucionais, totalmente conectadas e MBConv), até à obtenção de um vetor com os logits de cada classe.
- Aplicação de pós-processamento: os logits são passados por uma função de ativação *softmax*, que converte os valores em probabilidades, e é calculado o *argmax* para determinar a classe predita.
- Por fim devolve ao sistema de segurança uma classificação binária (presença ou ausência de arma) acompanhada do grau de confiança.

### Categoria: Reconhecimento de expressões emocionais

Esta categoria tem como principal função identificar expressões faciais associadas a estados emocionais de risco, nomeadamente medo e raiva, nos frames analisados pelo sistema. Para essa finalidade são utilizados dois recursos de computação visual disponibilizados em repositórios públicos: *DeepFace* [59] e *FER* [66], ambos integrados como serviços externos que comunicam com o sistema através de *endpoints* definidos.

- **DeepFace:** funciona como serviço externo que recebe os *frames*, deteta e alinha as faces e devolve a análise das expressões. Internamente, recorre à biblioteca DeepFace, que por sua vez utiliza diferentes métodos de detecção de face (nesta implementação foi usado o detetor OpenCV Haar, pela sua rapidez) e utiliza um modelo integrado baseado numa CNN leve (*mini-Xception*) treinada no conjunto FER-2013 [56], que produz a distribuição de probabilidades pelas várias emoções (felicidade, tristeza, raiva, surpresa, medo, nojo e neutro).

- **FER:** também integrado como serviço externo, recebe os *frames* e realiza a deteção da face (neste caso com o MTCNN, escolhido pela maior robustez a ângulos e variações de pose) antes de classificar a emoção. Trata-se de uma biblioteca simples e focada exclusivamente no reconhecimento de expressões emocionais. Tal como o DeepFace, utiliza uma CNN leve treinada no FER-2013 [56], e devolve apenas a emoção dominante de cada face com o respetivo grau de confiança.

Para que o reconhecimento emocional seja possível é necessário que os serviços identifiquem previamente as faces presentes em cada *frame*. Todas as faces detetadas são então ordenadas de acordo com a sua área seguindo o critério de maior dimensão. Cada serviço seleciona internamente no máximo as duas faces de maior área enquanto se existir apenas uma face apenas essa é processada e quando não for detetada nenhuma não é realizada classificação emocional nesse *frame*. Esta opção privilegia as faces mais próximas da máquina de autoatendimento seguindo a lógica de abranger o utilizador e um eventual agressor ou transeunte próximo.

As áreas das faces selecionadas são incluídas na resposta dos serviços e reutilizadas como métrica complementar de contexto. A métrica é transmitida ao controlador apenas como informação adicional, sem influenciar a decisão de alarme, sendo posteriormente utilizada para calcular o rácio entre a área da segunda face e a do utilizador principal. Sempre que esse rácio ultrapassa o limiar definido nas configurações do sistema, é emitido um aviso que sugere a presença de um intruso demasiado próximo. Em cenários típicos, a face do utilizador legítimo ocupa a maior área da imagem por estar em primeiro plano.

A Figura 4.2 ilustra estes dois cenários: à esquerda observa-se uma situação normal, em que a segunda face corresponde a uma pessoa em segundo plano, claramente afastada do utilizador principal, e à direita apresenta-se um caso em que a segunda face surge demasiado próxima, simulando a presença de um intruso a invadir o espaço pessoal do utilizador.

A decisão de implementar esta funcionalidade com base nos métodos já usados para a análise emocional deve-se à eficiência computacional, uma vez que técnicas mais sofisticadas como RetinaFace ou Dlib CNN exigem maior processamento e não são adequadas a máquinas de autoatendimento. Como o reconhecimento facial ou biométrico não é objetivo desta dissertação, aproveitou-se a deteção de faces já necessária para as emoções, acrescentando esta capacidade sem impacto significativo no tempo de inferência.

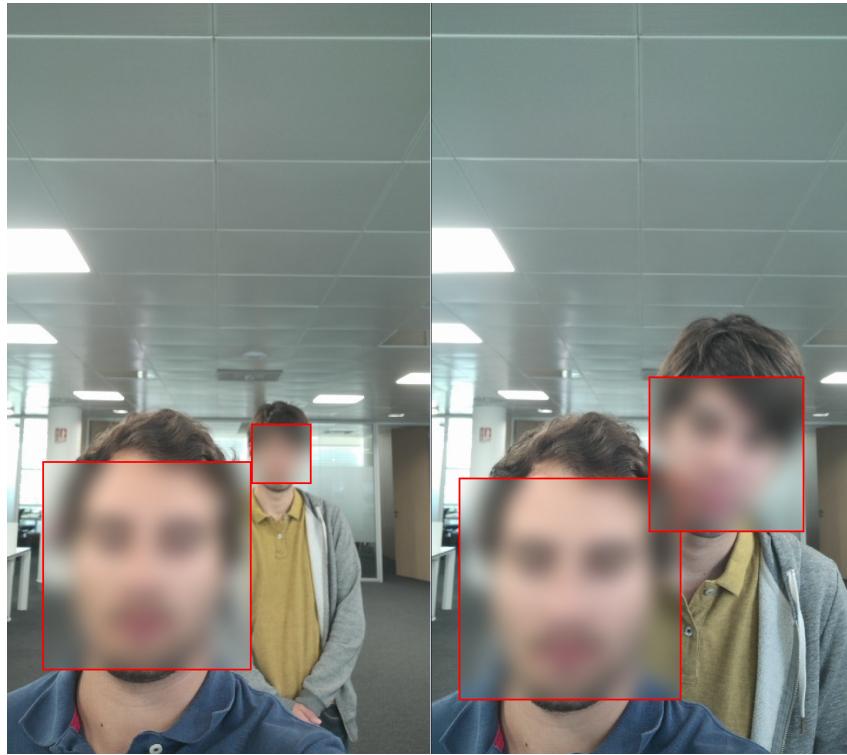


Figura 4.2: Exemplo de utilização da métrica de áreas faciais para informar sobre situações de intrusão.

### Fluxo de processamento dos modelos

Apesar de utilizarem mecanismos distintos de detecção e alinhamento, os serviços DeepFace e FER seguem um fluxo de processamento semelhante. A Figura 4.3 ilustra de forma resumida as etapas que compõem esse fluxo, enquanto a descrição seguinte detalha cada uma delas:

1. **Pré-processamento:** o *frame* é convertido para um formato de imagem manipulável (Array NumPy).
2. **Deteção da face:** cada serviço de emoções recorre ao seu modelo interno para localizar as faces no *frame*. No máximo, as duas maiores são enviadas para as etapas seguintes de alinhamento e classificação.
  - DeepFace – recorre à sua pipeline interna (nesta implementação configurada com o detetor OpenCV Haar).
  - FER – utiliza o detetor MTCNN, com pontos-chave (olhos, nariz, boca) para maior precisão.
3. **Alinhamento:** a face é padronizada em termos de posição e orientação.
  - DeepFace – alinhamento interno automático.
  - FER – alinhamento explícito baseado na posição dos olhos.
4. **Normalização:** a região da face é convertida para escala de cinzento, redimensionada para 48×48 pixels e normalizada para o intervalo de entrada da rede.
5. **Classificação:** é executada a CNN de cada serviço para atribuição de probabilidades às sete classes de emoção (felicidade, tristeza, raiva, surpresa, medo, nojo e neutro).

6. **Saída:** a API de cada serviço devolve, para até duas faces por *frame*, as emoções dominantes com o respetivo grau de confiança e as áreas faciais. As áreas são reutilizadas no controlador como métrica de proximidade entre indivíduos.

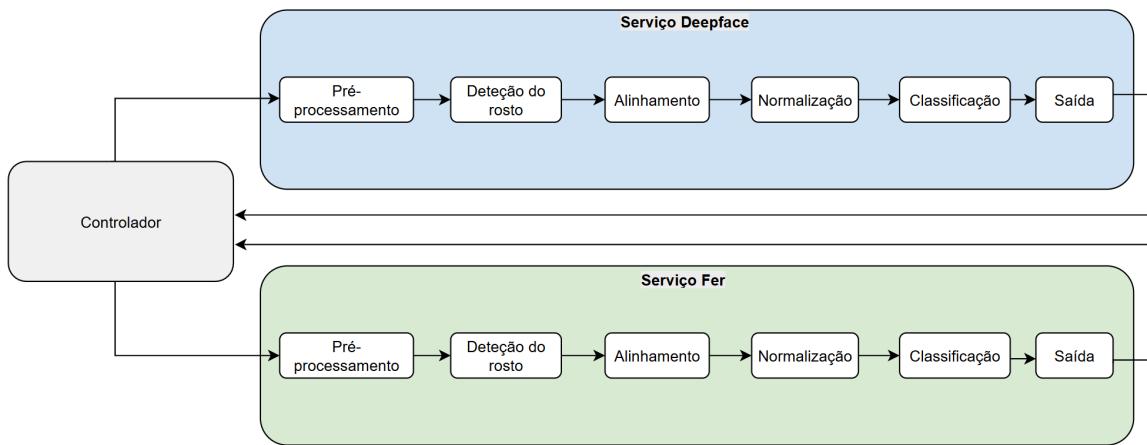


Figura 4.3: Fluxo global de processamento dos serviços de análise emocional (DeepFace e FER).

#### Categoria: Deteção de ambiente violento

Esta categoria visa identificar situações de violência física ou comportamentos agressivos no ambiente. Para esta tarefa foi utilizado um modelo YOLOv8 pré-treinado, disponibilizado em repositório público [67]. O modelo está em funcionamento como serviço externo, comunicando com o sistema através do respetivo *endpoint*.

O fluxo de processamento deste modelo segue o mesmo padrão técnico descrito para os modelos YOLOv8 da categoria de deteção de armas, incluindo as etapas de redimensionamento e normalização dos *frames*, passagem pela *backbone*, *neck* e *head* da CNN, e aplicação de NMS.

A principal diferença reside nas classes detetadas, que neste modelo são específicas para cenários de violência e restringem-se a duas possibilidades:

- ***Violence/Fight*** – instâncias em que ocorre violência física ou confrontos agressivos.
- ***No Violence/No Fight*** – instâncias em que não há qualquer tipo de confronto físico.

#### 4.2.4 Regras de Decisão

O Sistema de Segurança não se limita a receber passivamente as análises e deteções produzidas pelos diferentes modelos, sendo necessário transformar esses resultados em decisões concretas sobre a ativação ou não do alarme. É neste ponto que entram as **regras de decisão**, responsáveis por definir como os diferentes outputs dos modelos são combinados para determinar o resultado final. No sistema, essas regras são aplicadas pelo controlador com base nos parâmetros definidos no ficheiro de configuração.

As regras de decisão estão organizadas em dois níveis hierárquicos, conforme ilustrado na Figura 4.4:

- **Nível de categoria:** as decisões são tomadas com base nos resultados dos modelos que integram cada categoria (por exemplo, armas, emoções ou ambiente). Em cada categoria, o controlador

aplica a regra definida na configuração para efetuar a fusão dos outputs dos respetivos modelos, produzindo um resultado agregado representativo dessa categoria.

- **Nível global:** após a avaliação individual das categorias, os resultados agregados de cada uma são combinados de acordo com a regra global definida, a fim de determinar de forma unificada se o sistema deve ou não proceder à ativação do alarme.

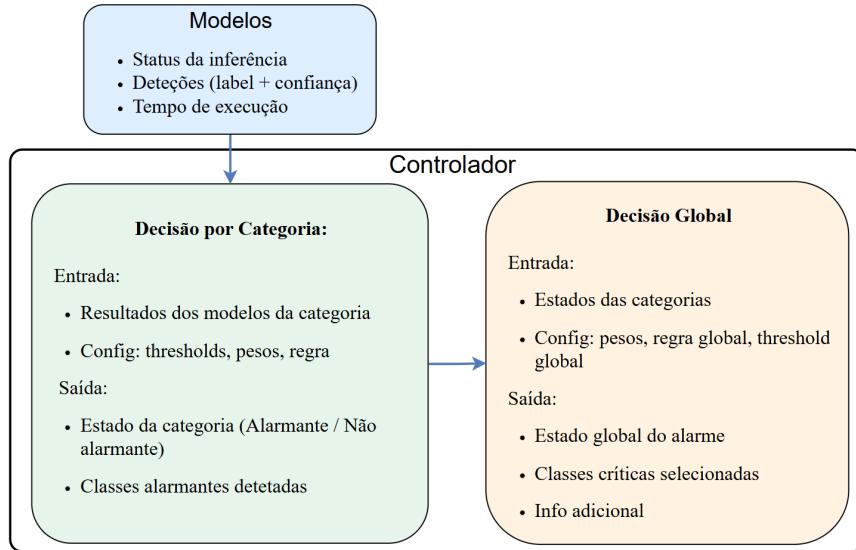


Figura 4.4: Diagrama dos níveis de decisão realizados pelo controlador no Sistema de Segurança.

Definidos os dois níveis hierárquicos de decisão, importa detalhar os mecanismos aplicados em cada um deles. Antes da aplicação das regras propriamente ditas, o controlador realiza uma verificação preliminar sobre as respostas brutas de cada modelo relativas ao lote de *frames* analisado: considera-se *positivo* se existir pelo menos um *frame* no lote com uma deteção cuja classe pertença ao conjunto de classes alarmantes  $\mathcal{A}_m$  e cuja confiança seja superior ou igual ao limiar individual  $t_m$ . Basta que um único *frame* do lote satisfaça estes critérios para que o modelo seja considerado como positivo. Esta condição encontra-se representada no pseudocódigo do Algoritmo 1 (linhas 5 e 15). Apenas os modelos que satisfazem esta condição prosseguem para a decisão ao nível da categoria, segundo as regras de *consensus* ou *scoring*, apresentadas de seguida. A decisão global segue o mesmo princípio, mas combina as categorias ativas em vez dos modelos.

- **Consensus:** implementa-se de forma hierárquica em dois níveis. No nível de *categoria*, considera-se alarmante a categoria em que pelo menos metade dos modelos associados confirmam uma deteção relevante, isto é, uma classe alarmante com confiança igual ou superior ao limiar definido para cada modelo. No nível *global*, o sistema apenas ativa o alarme quando a maioria das categorias avaliadas são alarmantes, garantindo assim que a decisão final decorre de um acordo entre diferentes áreas de análise. Esta abordagem é particularmente útil em cenários onde não se dispõe de informação detalhada sobre o desempenho individual dos modelos, permitindo combinar os seus *outputs* de forma equitativa e aumentando a robustez face a erros isolados [68].
- **Scoring:** no nível de *categoria* cada modelo contribui com uma pontuação igual ao seu peso quando deteta uma classe alarmante com confiança acima do seu limiar individual. A soma destas

pontuações é comparada com o limiar da categoria. No nível *global* cada categoria ativa contribui com um peso fixo definido na configuração `config.json`. A soma dos pesos das categorias alarmantes é comparada com o limiar global para decidir a ativação do alarme. Esta abordagem é mais adequada quando existe conhecimento detalhado sobre o desempenho relativo dos modelos envolvidos, permitindo atribuir-lhes pesos proporcionais à sua fiabilidade [69].

---

**Algorithm 1:** Decisão por Categoria (regra *consensus* ou *scoring*).

---

**Input:** Categoria  $c$  ativa; regra  $R_c \in \{\text{consensus}, \text{scoring}\}$ ; limiar da categoria  $T_c$ ; modelos  $\mathcal{M}_c$  (cada  $m$  com  $w_m, t_m, \mathcal{A}_m$ ) e respetivas detecções.

**Output:** Resultado booleano  $y_c$ .

```

1  $N \leftarrow |\mathcal{M}_c|$ ;  $y_c \leftarrow \text{falso}$ 
2 if  $R_c = \text{consensus}$  then
3    $votos \leftarrow 0$ ;  $maj \leftarrow \lfloor N/2 \rfloor + 1$ 
4   foreach modelo  $m \in \mathcal{M}_c$  do
5      $positivo \leftarrow \exists(\text{label}, \text{conf})$  tal que  $\text{label} \in \mathcal{A}_m \wedge \text{conf} \geq t_m$ 
6     if  $positivo$  then
7        $votos \leftarrow votos + 1$ 
8     if  $votos \geq maj$  then
9        $y_c \leftarrow \text{verdadeiro}$ ; return  $y_c$                                 // short-circuit
10    return  $y_c$       // verdadeiro se maioria alcançada; caso contrário,
11          falso
11 else                                         //  $R_c = \text{scoring}$ 
12    $score \leftarrow 0$ ; if  $T_c = \text{then}$ 
13      $T_c \leftarrow 1$ 
14   foreach modelo  $m \in \mathcal{M}_c$  do
15      $positivo \leftarrow \exists(\text{label}, \text{conf})$  tal que  $\text{label} \in \mathcal{A}_m \wedge \text{conf} \geq t_m$ 
16     if  $positivo$  then
17        $score \leftarrow score + w_m$ 
18     if  $score \geq T_c$  then
19        $y_c \leftarrow \text{verdadeiro}$ ; return  $y_c$                                 // short-circuit
20   return  $y_c$ 
```

---

Em termos conceptuais, as regras de decisão definem como os outputs dos modelos e das categorias são combinados. Na prática, o ficheiro de configuração (`config.json`) especifica as regras a aplicar por categoria e a nível global, também reúne limiares, pesos e o estado de cada categoria. A configuração é carregada no arranque e permanece em memória para acesso eficiente. A sua estrutura organiza-se em três blocos: **Decisão global** inclui a regra de combinação das categorias, o limiar global e pode incluir parâmetros adicionais como a razão entre áreas faciais. **Categorias** inclui o nome, o estado ativo, a própria regra e o limiar, o peso na decisão global e o conjunto de modelos associados. **Modelos** inclui o endpoint, as classes alarmantes, o limiar individual e o peso na decisão da categoria. A Figura 4.5 apresenta um excerto representativo do `config.json` e evidencia a função de cada parâmetro.

**Estrutura do config.json:**

**categories\_decision\_rule:** regra global de combinação das categorias (*consensus* ou *scoring*).

**threshold:** limiar global aplicado à decisão final.

**faces\_areas\_ratio\_threshold:** percentagem que define o rácio mínimo entre a área da segunda face e a área da face do utilizador para sinalizar proximidade.

**categories:** definição das categorias do sistema.

**name:** identificador da categoria (ex.: *weapon\_detection*).

**active:** indica se a categoria está ativa ou inativa.

**decision\_rule:** regra de decisão da categoria (*consensus* ou *scoring*).

**threshold:** limiar associado à decisão da categoria.

**weight:** peso atribuído à categoria na decisão global.

**models:** lista de modelos associados à categoria.

**name:** nome do modelo.

**url:** endereço de comunicação com o modelo (endpoint).

**classes:** classes alarmantes do modelo.

**thresholdDetection:** confiança mínima exigida para considerar uma deteção válida do modelo.

**weight:** peso do modelo na decisão da categoria, caso a categoria seja por *scoring*.

Figura 4.5: Excerto descritivo do config.json, com indicação da função de cada parâmetro.

A Tabela 4.1 mostra um exemplo de entradas agregadas por lote de dois *frames* com uma resolução de  $720 \times 1280$ . O controlador aplica a regra *scoring* por categoria: sempre que um modelo apresenta pelo menos uma deteção ou classificação com confiança  $\geq \text{thr}$  (*thresholdDetection*) em qualquer um dos *frames* do lote, o seu peso  $w$  é somado à pontuação da categoria. Esta torna-se *alarmante* quando a soma dos pesos atinge ou excede  $T_c$ . Na tabela 4.1,  $\text{thr}$  corresponde ao limiar de confiança definido individualmente para cada modelo (*thresholdDetection* na Figura 4.5),  $w$  representa o peso atribuído ao modelo na decisão da respetiva categoria (*weight*), e  $T_c$  é o limiar da própria categoria, isto é, a soma mínima de pesos necessária para que esta seja considerada *alarmante*.

Concretamente, na linha da categoria **Armas**, os modelos YoloA e YoloB apresentam, no primeiro frame do lote, a classe *gun* com confiança  $\geq \text{thr}$ , contribuindo respetivamente com  $w = 3$  e  $w = 2$ . A soma destes pesos atinge  $T_c = 5$ , pelo que a categoria **Arma** é considerada *alarmante*, enquanto YoloC e EfficientNet não apresentam deteções em nenhum dos frames e, por isso, não contribuem para a decisão.

Importa salientar que os valores de limiar e peso ilustrados nesta tabela têm caráter exemplificativo. A definição destes parâmetros resulta de análise experimental, explorada em detalhe na Secção 5.2.4, com recurso a métricas como curvas ROC.

Categoría	Exemplo de entradas (JSON)	thr / w	Regra	Janela
Arma	<pre>{   'model': 'YoloA',   'detections': [ [ {'label': 'gun', 'conf': 0.87}], [] ] } {   'model': 'YoloB',   'detections': [ [ {'label': 'gun', 'conf': 0.91}], [] ] } {   'model': 'YoloC',   'detections': [ [], [] ] } {   'model': 'EfficientNet',   'detections': [ [], [] ] }</pre>	YoloA: thr=0.4 / w=3 YoloB: thr=0.2 / w=2 YoloC: thr=0.4 / w=1 eff: thr=0.2 / w=1	scoring, $T_c = 5$	lote com 2 frames (720×1280)
Emoções	<pre>{   'model': 'deepface',   'detections': [ [ {'emotion': 'fear', 'conf': 0.82}],     [ {'emotion': 'neutral', 'conf': 0.72}] ] } {   'model': 'fer',   'detections': [ [ {'emotion': 'neutral', 'conf': 0.78}],     [ {'emotion': 'neutral', 'conf': 0.87}] ] }</pre>	deepface: thr=0.4 / w=2 fer: thr=0.4 / w=1	scoring, $T_c = 1$	lote com 2 frames (720×1280)
Violência	<pre>{   'model': 'violence',   'detections': [ [], [] ] }</pre>	thr=0.3 / w=5	scoring, $T_c = 1$	lote com 2 frames (720×1280)

Tabela 4.1: Exemplo de entradas agregadas por lote de 2 *frames*, onde cada modelo aparece uma vez e o campo *detections* inclui os resultados de ambos os *frames*.

### 4.3 Dinâmica Operacional do Sistema de Segurança

Após a definição da arquitetura modular e dos componentes funcionais do sistema, importa compreender a dinâmica de funcionamento em tempo de execução. Esta secção descreve o fluxo de operação do Sistema de Segurança, desenvolvido como solução de segurança adicional para as máquinas de autoatendimento, que destaca o modo como os diferentes módulos interagem para processar os dados recebidos, executar inferências com os modelos de computação visual e tomar decisões baseadas em regras configuradas.

A explicação está dividida em duas perspetivas complementares: o **fluxo interno** do sistema, que detalha o percurso dos dados no interior do sistema e a sua interação com os modelos de computação visual, e o **fluxo cíclico**, que representa o ciclo completo para demonstrar como o sistema lida com a receção de múltiplos frames, ilustrando a forma como organiza, processa e responde continuamente aos dados visuais recebidos.

#### 4.3.1 Fluxo Interno de Processamento

O fluxo interno de processamento representa a sequência completa de operações realizadas dentro do Sistema de Segurança, desde a receção dos *frames* até à emissão da decisão final. A Figura 4.6 ilustra esse percurso, detalhando a interação entre os módulos internos do sistema e os modelos de computação visual.

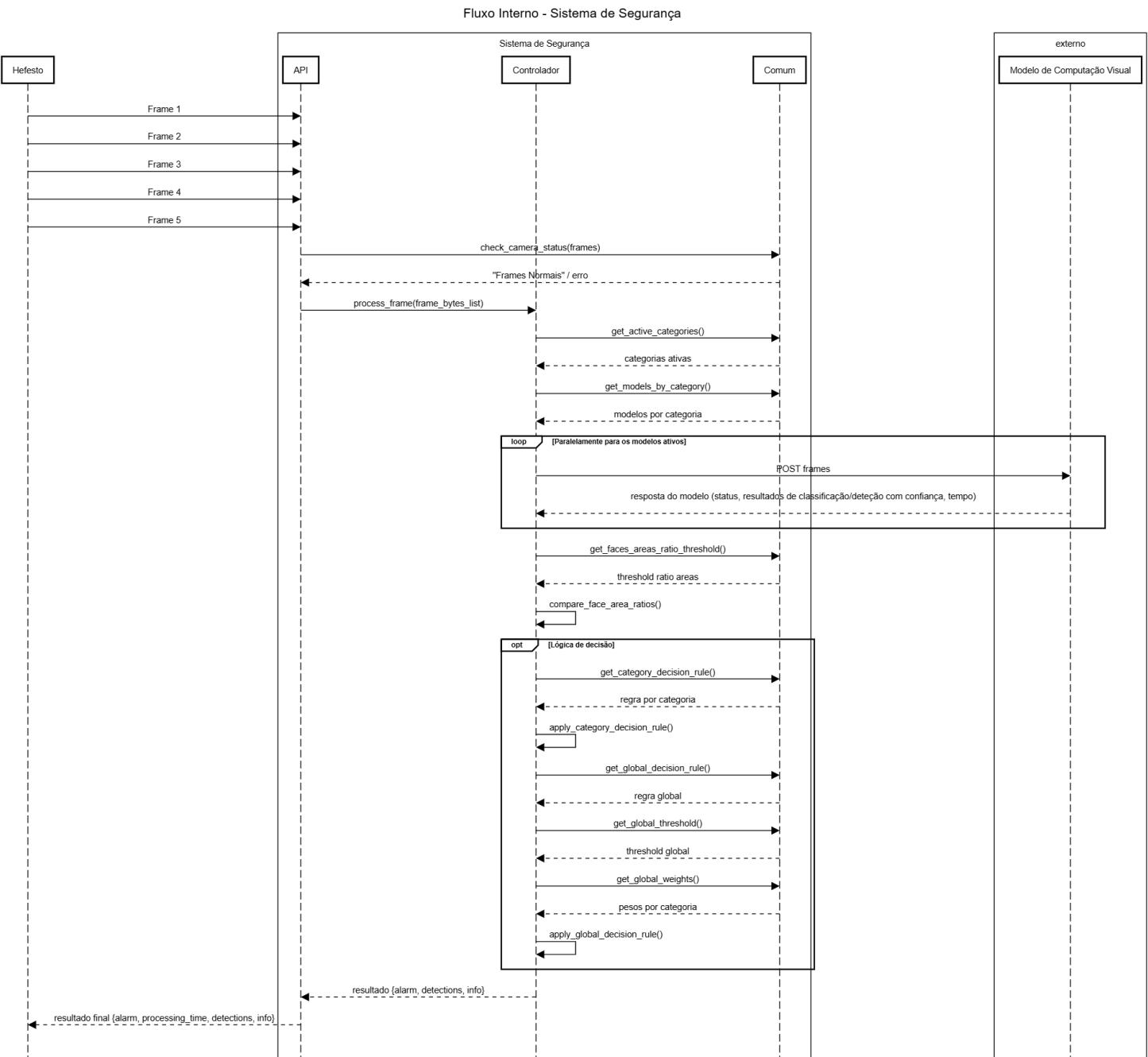


Figura 4.6: Fluxo interno de processamento no Sistema de Segurança, desde a receção dos frames até à decisão final.

O processo inicia-se na máquina de autoatendimento como o HEFESTO, que captura os frames através da sua câmara embutida. Esses frames são enviados sequencialmente para a API do sistema de segurança. A API é responsável por receber os frames em formato de Bytes e validar a sua qualidade antes de permitir o seu processamento.

Cada frame recebido passa por uma verificação de integridade, realizada pela função `check_camera_status`. Esta validação serve para assegurar que a câmara não está bloqueada, coberta, exposta a uma iluminação excessiva ou com sinais de adulteração (como coloração anormal por pintura). Caso os frames estejam em condições aceitáveis, são considerados válidos para análise.

A API mantém uma estrutura interna baseada num buffer circular (`deque`) que armazena os frames válidos recebidos. No entanto, ao contrário de uma abordagem que simplesmente processaria os últimos cinco frames recebidos, o sistema foi concebido para selecionar, dentro desse buffer, um conjunto de frames espaçados temporalmente. Esta estratégia permite aumentar o valor informativo da análise, evitando que os frames processados sejam demasiado semelhantes entre si, um problema comum quando os dados provêm de uma fonte com um elevado FPS. Desse modo, o sistema distribui os frames escolhidos ao longo do tempo de forma mais equilibrada, que dá uma visão mais ampla da cena captada.

Quando a API acumula o número necessário de frames, e todos são validados com sucesso, o lote é encaminhado para o controlador através do método `process_frame`. A partir deste ponto, o controlador assume a responsabilidade pelo pipeline de processamento.

O controlador começa por consultar o módulo *Comum* para obter a lista de categorias ativas e a seguir vai opter os modelos associados a cada categoria, com base nas configurações definidas no ficheiro `config.json`. Com essa informação, o controlador envia em paralelo os *frames* para a API de cada serviço correspondente a um modelo de computação visual ativo. Cada serviço recebe os dados, executa a inferência em lote e devolve ao controlador os resultados, incluindo as deteções e os respetivos níveis de confiança.

A partir destas respostas, o controlador realiza também uma avaliação complementar, centrada na proximidade entre as faces detetadas. Para tal, calcula o rácio entre a área da face do utilizador e a área de uma segunda face identificada. Caso esse rácio ultrapasse o limiar definido na configuração, é registada informação adicional que, embora não altere diretamente o estado do alarme, fornece contexto útil sobre a presença de uma potencial segunda pessoa junto ao utilizador.

Com a posse das respostas dos modelos, o controlador avança para a etapa de lógica de decisão. Esta etapa decorre em duas fases: decisão por categoria e decisão global, aplicadas de acordo com as regras e parâmetros definidos no ficheiro `config.json`, tal como detalhado na Secção 4.2.4.

O resultado final do controlador é um objeto que inclui o estado de alarme final, o tempo total de processamento, as deteções mais relevantes associadas às categorias que dispararam alarme e também a informação adicional sobre o rácio das faces. Esta resposta é enviada de volta para a API, que a devolve ao cliente em formato JSON, pronto para ser interpretado pela aplicação da máquina de autoatendimento.

### 4.3.2 Fluxo Cíclico do Sistema

O funcionamento do Sistema de Segurança baseia-se num ciclo contínuo de receção, análise e resposta, que decorre em paralelo com o envio permanente de *frames* por parte da aplicação da máquina de autoatendimento (neste caso representada pelo HEFESTO). A Figura 4.7 ilustra esse ciclo, utilizando como exemplo o envio de 5 FPS pelo HEFESTO e a análise realizada pelo sistema em lotes de 5 *frames*.

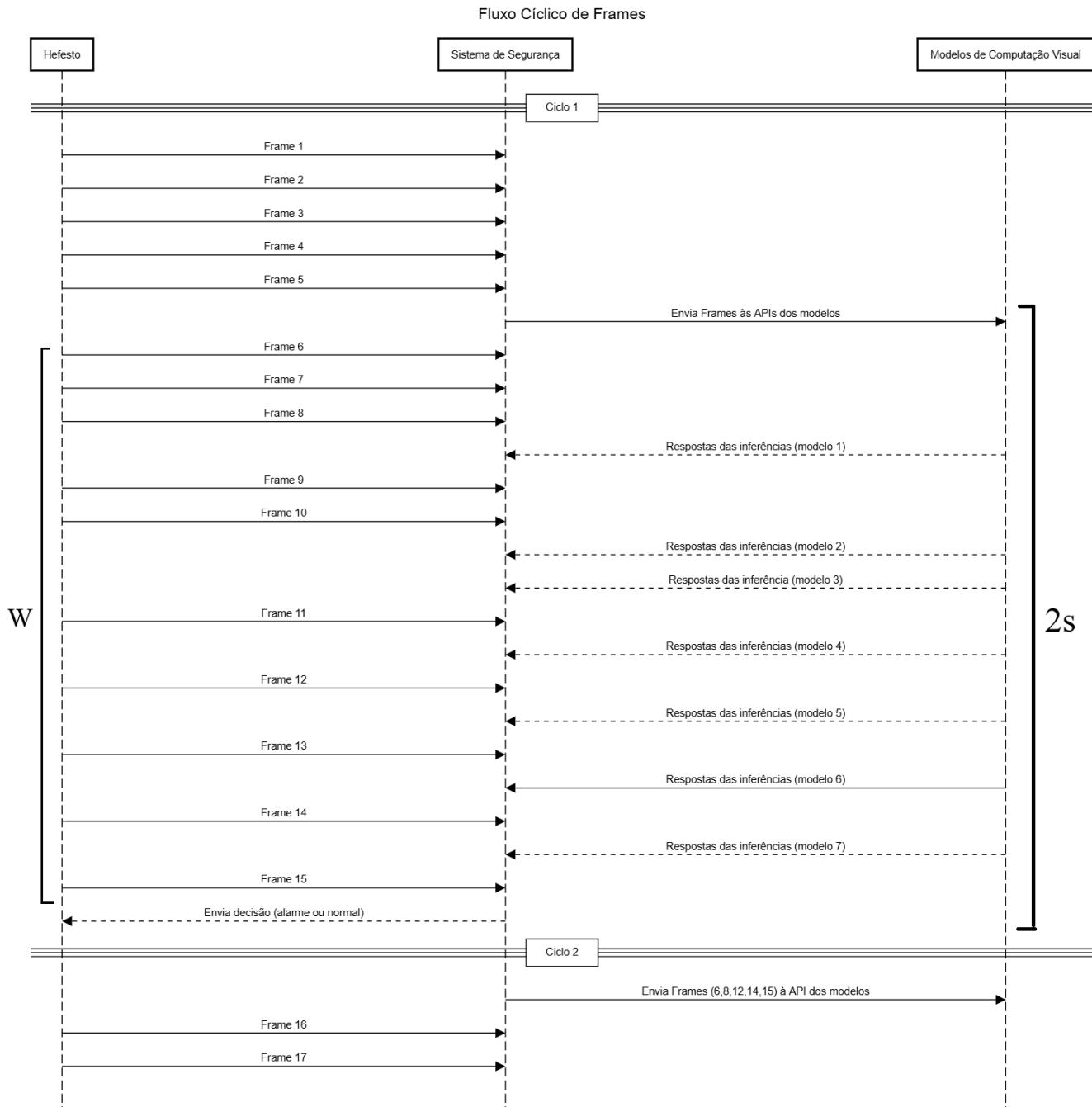


Figura 4.7: Exemplo de ciclo de análise no Sistema de Segurança, com receção contínua de 5 FPS e envio em lote para os modelos de computação visual, sendo  $W$  a janela de decisão.

Neste contexto, o sistema de segurança necessita de ter disponíveis pelo menos 5 *frames* para iniciar o processamento de uma análise. Os *frames* recebidos são armazenados no buffer da API, que acumula todos os que ainda não foram utilizados em análises anteriores. Assim que o buffer contém o número mínimo de 5 *frames*, estes são selecionados e enviados para os modelos de computação visual ativos. O sistema aguarda pelas respostas de inferência e aplica a lógica de decisão previamente descrita. A decisão final é então enviada de volta para a máquina de autoatendimento, marcando o fim de um ciclo de análise.

Enquanto o sistema realiza o processamento e a inferência dos *frames* do ciclo atual, novos *frames*

continuam a chegar. Esses *frames* são todos armazenados no *buffer*, e quando a análise em curso termina, são utilizados apenas os novos para iniciar o ciclo seguinte. A seleção dos novos 5 *frames* baseia-se num critério de espaçamento uniforme ao longo do conjunto de *frames* acumulados, garantindo uma representação equilibrada de todo o intervalo temporal. O processo de seleção e análise dos *frames* pode ser descrito em quatro elementos principais: como são escolhidos (*Amostragem*), como as janelas avançam (*Janelamento*), como os resultados são combinados (*Agregação*) e quanto tempo decorre até à decisão (*Latência*).

- **Amostragem:** no exemplo ilustrado na Figura 4.7, os *frames* são recebidos da câmera a uma taxa de  $r = 5$  FPS. Durante cada janela de decisão  $W$ , esses *frames* são acumulados no *buffer*, mas apenas uma parte é selecionada para análise. Cada lote é constituído por  $N = 5$  *frames*, escolhidos do *buffer* de forma equidistante, garantindo diversidade temporal no conjunto enviado aos modelos de computação visual. Os índices  $i$  dos *frames* escolhidos são calculados por:

$$i = \text{int}\left(\frac{k \cdot T}{N}\right), \quad k = 0, 1, \dots, N - 1 \quad (4.1)$$

em que  $i$  corresponde ao índice do *frame* selecionado no *buffer*,  $T$  representa o número total de *frames* disponíveis na janela,  $N$  é o número de *frames* a incluir no lote (neste caso, cinco) e  $k$  é o contador que percorre os valores de 0 a  $N - 1$ , permitindo selecionar sucessivamente os diferentes *frames*.

Por exemplo, com  $T = 50$  *frames* acumulados no *buffer* ao longo de uma janela, seriam escolhidos aproximadamente os índices 0, 10, 20, 30 e 40, assegurando o espaçamento uniforme.

- **Janelamento:** a janela de decisão  $W$  corresponde ao intervalo temporal de vídeo considerado para cada análise, sendo representada por um conjunto de *frames* amostrados do *buffer*. O avanço  $H$  define o deslocamento entre o início de duas janelas consecutivas. No sistema implementado, foi definido  $H = W$ , o que significa que uma nova janela só é iniciada após a conclusão da anterior, sem sobreposição temporal entre elas. Na implementação de exemplo ilustrada na Figura 4.7, o valor de  $W$  correspondeu a aproximadamente 2 segundos, servindo apenas como exemplo ilustrativo da política adotada. Nesta configuração, cada janela  $W$  corresponde ao conjunto de *frames* recebidos da câmera durante o período em que um ciclo de análise está em execução. Esses *frames* são acumulados no *buffer* e apenas processados no ciclo seguinte, o que evita redundância mas atrasa a sua utilização efetiva. Como consequência, o sistema só produz uma decisão a cada  $W$  segundos, o que reduz a frequência de atualização em comparação com políticas em que  $H < W$  e as janelas se sobreponham. A escolha de  $H = W$  foi motivada por restrições computacionais, uma vez que um avanço inferior implicaria o processamento simultâneo de janelas sobrepostas, resultando em maior custo computacional e em maior consumo de memória para armazenar múltiplos conjuntos de *frames*.
- **Agregação:** os *frames* escolhidos segundo a política de espaçamento temporal definida são analisados em lote no sistema, considerando-se que um modelo contribui para a decisão da categoria se apresentar pelo menos uma deteção válida com confiança  $\geq \text{thresholdDetection}$  em qualquer dos *frames*. Não se aplica média ou máximo das confianças ao longo do lote, mas sim a verificação da

existência de uma ocorrência válida. A seguir, é feita a decisão para cada categoria e, posteriormente, a decisão final do ciclo de análise.

- **Latência:** a latência do evento é definida como o intervalo entre o instante em que ocorre o primeiro *frame* de um evento real (por exemplo, o primeiro *frame* violento) e o final da primeira janela de decisão que o reconhece. No sistema implementado, as janelas têm duração  $W$  e são contíguas ( $H = W$ ). No entanto, como os *frames* captados durante uma janela apenas são processados no ciclo seguinte, a latência não pode ser inferior a  $W$ , situando-se no intervalo  $[W, 2W]$ . O valor mínimo ocorre quando o evento se inicia no final de uma janela, pois será rapidamente incluído no ciclo seguinte e reconhecido no seu final, resultando numa latência próxima de  $W$ . O valor máximo ocorre quando o evento começa no início de uma janela, já que os seus *frames* apenas serão processados no ciclo seguinte e a decisão só estará disponível no final deste, resultando numa latência próxima de  $2W$ . No exemplo da Figura 4.7, com  $W \approx 2$  segundos, um evento iniciado no *frame* 14 ( $t = 2.8$  s) foi reconhecido apenas no final do Ciclo 2 ( $t = 5$  s), resultando numa latência de 2.2 s.

Na Figura 4.7, o Ciclo 1 inicia-se com os primeiros 5 *frames* válidos recebidos, sem intervalos. Já no Ciclo 2, os lotes são formados apenas a partir dos *frames* captados durante o processamento do ciclo anterior. No exemplo, foram selecionados os *frames* 6, 8, 12, 14 e 15, ilustrando a política de espaçamento temporal definida. Esta abordagem cíclica garante que o sistema mantém um fluxo contínuo de percepção e análise, privilegiando a diversidade temporal dos dados. Importa salientar que os valores usados nesta secção têm apenas caráter ilustrativo, servindo para descrever o funcionamento dos ciclos do sistema. A análise experimental que fundamenta a escolha do número de *frames* por lote é apresentada na Subseção 5.3.2.

### 4.3.3 Implantação Local, Integração com o HEFESTO e Considerações de Privacidade

O Sistema de Segurança foi concebido para execução local nas máquinas de autoatendimento, como o HEFESTO, garantindo operação autónoma e em tempo real, sem necessidade de ligação constante a servidores externos.

A instalação é efetuada por técnicos que transferem os executáveis do sistema e dos modelos de inferência, acompanhados dos ficheiros de configuração. O processo é modular: um executável principal para o sistema de segurança e outros independentes para cada modelo, o que permite substituir ou adicionar modelos sem reconfigurar o executável do sistema.

Figura 4.8 apresenta uma visão macro da arquitetura local do sistema de segurança instalado na máquina de autoatendimento, evidenciando não apenas a sua separação em relação à aplicação principal do HEFESTO, mas também a sua estrutura modular composta por API, controlador, modelos de inferência e componentes auxiliares. Nesta representação, destaca-se ainda a separação entre os executáveis: o contorno rosa representa o executável principal do sistema de segurança, enquanto o contorno amarelo identifica os executáveis independentes dos modelos de computação visual.

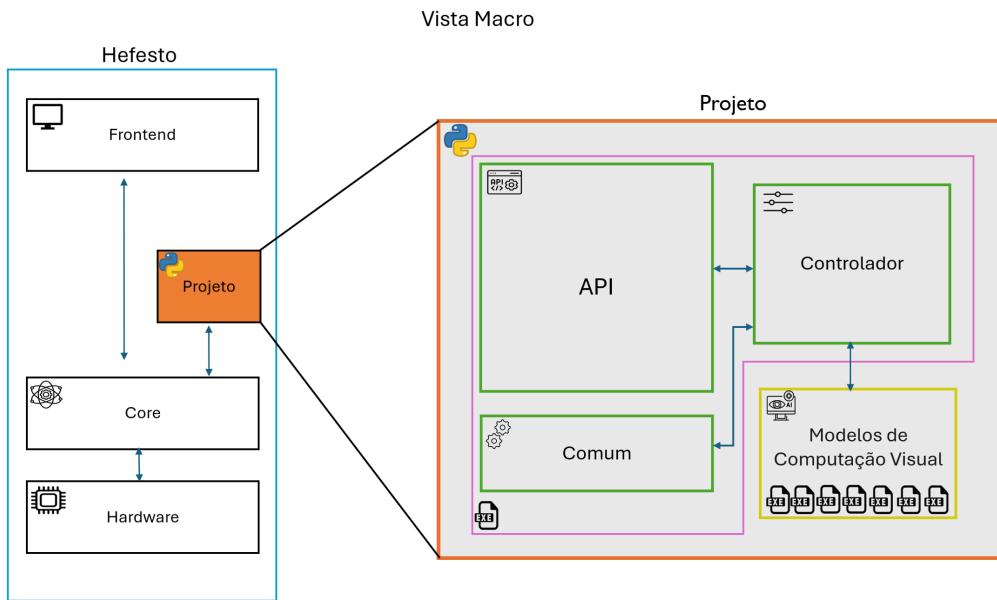


Figura 4.8: Arquitetura local do sistema de segurança e sua integração com o HEFESTO.

A integração com o HEFESTO ocorre sem alterações na lógica da aplicação principal: os executáveis são definidos no ficheiro de inicialização, que estabelece as portas e endereços de comunicação tanto do sistema principal como de cada modelo de inferência. A aplicação do HEFESTO apenas precisa de capturar os *frames* da câmara e enviá-los, via HTTP, para o sistema de segurança, o que simplifica a integração e permite o acoplamento a diferentes aplicações.

Quanto à privacidade, o sistema adota uma política de não armazenamento. Nenhum *frame* é guardado em disco, mesmo em situações de alarme, sendo processados apenas em memória e descartados após a análise. Esta decisão garante conformidade com os princípios de proteção de dados, especialmente relevantes em contextos bancários.

# Capítulo 5

## Análise

### 5.1 Avaliação dos modelos de Computação Visual

Esta secção descreve o processo de avaliação dos modelos de computação visual utilizados no Sistema de Segurança, desenvolvido com o objetivo de verificar o desempenho dos modelos em tarefas específicas, como a deteção de armas, o reconhecimento de expressões emocionais e a identificação de comportamentos violentos.

Para isso, foram selecionados conjuntos de dados adequados a cada uma destas categorias, com base em critérios de relevância e disponibilidade. A avaliação foi realizada de forma controlada, utilizando scripts dedicados para a execução e medição de métricas como precisão, *recall* e *F1-score*. Esta análise permitiu observar o comportamento individual de cada modelo e identificar limitações ou pontos fortes relevantes para a integração com o sistema de segurança. Os resultados obtidos serviram também de apoio à fase de validação do sistema, permitindo uma melhor compreensão do impacto de cada modelo na decisão final.

#### 5.1.1 Pesquisa e Seleção de Datasets

Para a avaliação dos modelos de computação visual utilizados pelo Sistema de Segurança, foi necessário proceder à recolha de adequados às categorias em análise, nomeadamente armas, expressões emocionais e ambientes violentos. No caso da deteção de armas, a seleção de datasets revelou-se particularmente desafiante devido à ausência de um conjunto de dados padrão amplamente aceite pela comunidade científica. Esta limitação é referida na literatura, que aponta a inexistência de um *benchmark* universal para deteção de armas e a consequente dificuldade na criação de bases de dados fiáveis e consistentes para avaliar os modelos [70, 71].

Face a esta realidade, os datasets utilizados nesta análise aos modelos de deteção de armas foram obtidos maioritariamente a partir de plataformas abertas como o Roboflow [72]. Apesar de fornecerem um volume significativo de dados, estas plataformas têm limitações, sobretudo por permitirem o envio de conteúdos por vários utilizadores. Isso levanta a possibilidade de que alguns dos conjuntos usados na avaliação incluem imagens que já foram utilizadas no treino dos próprios modelos analisados. Tal pode enviesar os resultados, ao testar os modelos com imagens que já conhecem.

Um aspecto crítico identificado na utilização de dados abertos é o risco de *leakage*. Alguns conjuntos disponíveis no Roboflow podem incluir imagens que foram utilizadas no treino de modelos de deteção de armas de uso público, o que pode enviesar a avaliação ao expor os modelos a exemplos previamente

conhecidos. Para além disso, existe também o desafio da mudança de domínio, uma vez que a maioria dos conjuntos foi recolhida em contextos diferentes do previsto para a aplicação final (ambientes de autoatendimento), o que pode impactar a capacidade de generalização dos modelos.

Para as restantes categorias (emoções e ambiente violento), os datasets foram selecionados pela sua relevância face ao contexto do sistema, procurando conjuntos que incluíssem as classes de maior interesse. Assim, optou-se por datasets já utilizados e reconhecidos em trabalhos científicos anteriores, ou com anotações de qualidade verificável e adequadas ao objetivo da análise.

Antes de apresentar os conjuntos de dados utilizados, importa esclarecer que as métricas de precisão e *recall* associadas aos datasets de deteção de armas foram disponibilizadas pelas próprias plataformas. Estas métricas referem-se ao desempenho de modelos demonstrativos treinados nesses datasets, que servem como referência para comparar os resultados obtidos pelos modelos utilizados neste trabalho. Para as restantes categorias, os datasets foram utilizados sem métricas associadas às suas plataformas, sendo descritos com base na literatura ou documentação oficial.

### Datasets para deteção de armas

A Tabela 5.1 apresenta os conjuntos de dados utilizados neste trabalho para a análise da categoria de deteção de armas. As métricas de precisão e *recall* indicadas correspondem aos valores publicados pelas plataformas que fornecem os datasets, obtidos a partir de modelos demonstrativos treinados nos próprios conjuntos. Estes valores são incluídos como referência e não refletem os resultados obtidos pelos modelos avaliados neste trabalho.

Tabela 5.1: Datasets de deteção de armas e respetivas métricas publicadas.

Origem	Descrição breve	Precisão (%)	Recall (%)
[73]	Imagens de armas (pistolas, facas) em contexto de CCTV	84,6	73,6
[74]	Armas de fogo e facas em múltiplos cenários	68,7	78,9
[75]	Dataset com classes <i>normal</i> , <i>suspicious</i> ( <i>suspeitos com armas</i> ), <i>victim</i> e <i>weapon</i>	77,1	73,8
[76]	Imagens da classe <i>silah</i> (arma de fogo)	97,2	95,0
[77]	Imagens de facas para deteção de armas brancas	88,5	83,8

### Datasets para reconhecimento de emoções

Para a categoria de reconhecimento de expressões emocionais foram utilizados dois conjuntos de dados amplamente reconhecidos na literatura, selecionados pela sua relevância para o contexto do Sistema de Segurança.

O *FER-2013* [78] contém cerca de 35000 imagens faciais em tons de cinzento (48x48 px), recolhidas de fontes online e anotadas em sete emoções básicas: felicidade, tristeza, raiva, surpresa, medo, nojo e neutro. É amplamente usado em estudos de referência e disponibiliza um volume significativo de amostras para tarefas de classificação de emoções.

O *RAF-DB* [79] inclui aproximadamente 15000 imagens faciais anotadas nas mesmas sete emoções, recolhidas em condições não controladas, com variações de pose, iluminação e expressão. É amplamente utilizado como *benchmark* na literatura, com estudos a reportarem valores de *accuracy* entre 77% e 85%

em tarefas de classificação emocional. Este conjunto oferece maior diversidade e aproxima-se melhor das condições reais de utilização em sistemas de vigilância.

Ambos os conjuntos de dados forneceram uma base adequada para avaliar os modelos na classificação de expressões associadas a situações de risco, como medo e raiva, que são o foco deste projeto.

### **Dataset para deteção de ambiente violento**

Para a categoria de deteção de ambientes violentos, foi selecionado um dataset de vídeo especificamente desenvolvido para representar cenários reais de segurança.

O *RWF-2000* [10] contém cerca de 2000 vídeos captados por câmeras de vigilância em espaços públicos, com distribuição equilibrada entre sequências violentas e não violentas. Este conjunto de dados foi escolhido por refletir situações reais de vigilância e pela sua ampla adoção em estudos sobre deteção de violência em vídeo.

Como síntese, a Tabela 5.2 apresenta em conjunto os datasets utilizados, organizados por categoria, tipo de dados e volume.

Tabela 5.2: Resumo dos datasets organizados por categoria, utilizados na avaliação dos modelos.

Dataset	Tipo de dados	Volume
<b>Categoria: Deteção de Armas</b>		
Roboflow Weapon CCTV [73]	Imagens	~ 7.000
Roboflow YOLO Weapon [74]	Imagens	~ 18.210
Roboflow Suspicious with Weapons [75]	Imagens	~ 7.000
Roboflow Silah [76]	Imagens	~ 28.962
Roboflow Knife [77]	Imagens	~ 5.000
<b>Categoria: Reconhecimento de Emoções</b>		
FER-2013 [78]	Imagens	~ 35.000
RAF-DB [79]	Imagens	~ 15.000
<b>Categoria: Deteção de Ambiente Violento</b>		
RWF-2000 [10]	Vídeos	2.000

### **5.1.2 Procedimento de Avaliação dos Modelos**

Para avaliar os modelos de computação visual utilizados no Sistema de Segurança, foi desenvolvido um conjunto de *scripts* em *Python* que permitiu aplicar de forma sistemática os datasets recolhidos aos modelos em análise e calcular as métricas de desempenho. O processo incluiu as seguintes etapas:

- **Execução dos modelos sobre os datasets:** cada modelo foi testado com os conjuntos de dados selecionados, através de um *script* dedicado que realizava a inferência sobre todas as imagens ou vídeos e armazenava os resultados.

- **Comparação com as anotações dos datasets:** os resultados produzidos pelos modelos foram automaticamente comparados com os disponibilizados nos datasets, para calcular as métricas de desempenho.
- **Cálculo de métricas:** foi criado um *script* específico para o cálculo das métricas principais (precisão, recall, F1-score, IoU médio, entre outras), permitindo uma análise quantitativa consistente dos modelos.
- **Análise de balanceamento:** foi ainda implementado um *script* para avaliar o balanceamento das classes em cada dataset, identificando possíveis desequilíbrios que pudessem influenciar a avaliação.

Este procedimento assegurou uma avaliação uniforme dos modelos e possibilitou a recolha de métricas fiáveis, úteis não apenas para comparação e análise, mas também para orientar a uma configuração do Sistema de Segurança, em particular no ajuste dos pesos atribuídos a cada modelo.

### 5.1.3 Avaliação dos Modelos de Detecção de Armas

Para a categoria de deteção de armas, os modelos pré-treinados utilizados pelo Sistema de Segurança foram avaliados com base nos datasets apresentados em [5.1.1](#) e nos modelos descritos em [4.2.3](#). O objetivo foi aferir o desempenho individual de cada modelo na identificação das classes relevantes para o sistema.

A avaliação foi orientada por um conjunto de métricas quantitativas, entre as quais se destacam a precisão, o *recall*, o *F1-score* e a média do *IoU*.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad \text{Precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \text{F1-score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}} \quad (5.1)$$

- **TP (True Positives)** – número de casos positivos corretamente identificados pelo modelo.
- **FP (False Positives)** – número de casos incorretamente classificados como positivos.
- **FN (False Negatives)** – número de casos positivos que o modelo não conseguiu detetar.

Neste contexto, o *recall* assume especial relevância, dado que o principal objetivo de um sistema de segurança é maximizar a deteção de eventos potencialmente perigosos. Uma taxa elevada de *recall* indica que o sistema consegue detetar a maioria das ocorrências relevantes, mesmo que isso implique um número superior de falsos positivos. Por essa razão, é preferível que o sistema gere um alarme falso do que falhe na deteção de uma ameaça real.

No contexto específico das máquinas de autoatendimento, os falsos positivos não constituem um problema crítico, uma vez que podem ser facilmente geridos através de mecanismos operacionais, como solicitar ao utilizador que volte a tentar, redirecionar o atendimento para um balcão físico ou aplicar

verificações adicionais. Já os falsos negativos representam uma falha muito mais grave, pois significam a não deteção de uma ameaça efetiva. Assim, o sistema foi concebido para dar prioridade à sensibilidade da deteção, ainda que isso implique lidar com alguns alarmes indevidos.

No entanto, a ocorrência excessiva de falsos positivos pode reduzir a confiança no sistema, tornando essencial encontrar um equilíbrio entre sensibilidade e fiabilidade. Para esse efeito, são também utilizadas métricas como o *F1-score*, que combina precisão e *recall* numa medida única, permitindo avaliar simultaneamente a capacidade de deteção e a robustez do sistema face a alarmes indevidos.

Outras métricas complementares, como a taxa de falsos positivos (*FPR*) e falsos negativos (*FNR*), foram também consideradas para caracterizar o comportamento dos modelos em diferentes cenários.

## Resultados

Os resultados obtidos estão apresentados na Tabela 5.3. Os valores apresentados correspondem às métricas obtidas a partir da avaliação dos modelos efetivamente utilizados no sistema de segurança, aplicados sobre cada um dos conjuntos de dados selecionados. Esta avaliação permite verificar o desempenho real dos modelos usados neste trabalho, em comparação com as métricas publicadas pelas plataformas de origem dos datasets.

Tabela 5.3: Desempenho dos modelos na deteção de armas, avaliados sobre múltiplos datasets.

Modelo	Dataset	Precisão	Recall	F1-score	IoU médio (%)
YoloA [62]	Yolo Weapon Detection [74]	0.798	0.660	0.722	43.24
	Knife Detection [77]	0.999	<b>0.838</b>	0.912	54.71
	Silah [76]	0.975	0.690	0.808	42.08
	Suspicious Detection [75]	0.399	0.556	0.465	23.56
	Weapon CCTV [73]	0.807	<b>0.713</b>	0.757	48.65
YoloB [63]	Yolo Weapon Detection [74]	0.779	0.421	0.546	48.24
	Silah [76]	0.982	0.344	0.509	44.13
	Suspicious Detection [75]	0.429	0.508	0.465	19.51
	Weapon CCTV [73]	0.798	0.533	0.639	48.87
EfficientNet [65]	Yolo Weapon Detection [74]	0.699	0.192	0.301	0.00
	Silah [76]	0.831	0.053	0.100	0.00
	Suspicious Detection [75]	0.242	0.129	0.168	0.00
	Weapon CCTV [73]	0.702	0.321	0.441	0.00
YoloC [64]	Yolo Weapon Detection [74]	0.840	<b>0.713</b>	0.771	65.30
	Knife Detection [77]	1.000	0.567	0.724	55.46
	Silah [76]	0.987	0.482	0.648	49.69
	Suspicious Detection [75]	0.383	0.371	0.377	28.55
	Weapon CCTV [73]	0.833	<b>0.738</b>	0.782	64.95

A comparação com os valores publicados na Subsecção 5.1.1 mostra que os modelos avaliados apresentaram desempenhos consistentes, mesmo quando testados diretamente nos datasets selecionados, sem treino ou ajuste específico para esses dados. Destacam-se os modelos YoloA [62] e YoloC [64], ambos com valores de *recall* superiores a 0.70 em múltiplos conjuntos (por exemplo, 0.713 no *Weapon CCTV* e 0.738 no mesmo dataset, respectivamente), o que demonstra capacidade de identificar a maioria das

ocorrências relevantes. O modelo YoloA alcançou ainda um desempenho particularmente elevado no dataset *Knife Detection*, com precisão de 0.999 e *recall* de 0.838, refletindo um equilíbrio robusto entre sensibilidade e fiabilidade. Já o modelo YoloC evidenciou os melhores valores de IoU médio, atingindo 65.30% no dataset *Yolo Weapon Detection*, o que reforça a qualidade da deteção espacial.

Estes resultados evidenciam uma boa capacidade de generalização dos modelos, e permitem, caso esteja configurada a regra de decisão por *scoring*, ajustar os pesos atribuídos a cada modelo no ficheiro de configuração do sistema, contribuindo para decisões mais eficazes.

#### 5.1.4 Avaliação dos Modelos de Reconhecimento de Emoções

Tal como na deteção de armas, foram avaliados os modelos pré-treinados de reconhecimento de emoções utilizados no Sistema de Segurança, nomeadamente o *DeepFace* e o *FER*.

A avaliação centrou-se nas emoções *fear* e *angry*, consideradas mais relevantes em cenários de segurança. Para tal, foram usados os conjuntos de dados FER-2013 [78] e RAF-DB [79], sendo cada imagem processada pelos modelos para inferência. No FER-2013 estavam disponíveis aproximadamente 1000 imagens de *Angry* e 1000 de *Fear*, dentro de um total de cerca de 35.000 imagens. No RAF-DB, a análise incidiu sobre cerca de 700 imagens de *Angry* e 300 de *Fear*, num total aproximado de 15.000 imagens.

As previsões geradas foram comparadas com as anotações reais de cada conjunto, permitindo calcular métricas como precisão, *recall* e *F1-score*, de forma a aferir o desempenho dos modelos na identificação das duas emoções alvo.

### Resultados

A análise evidenciou que o modelo *DeepFace* [59] apresentou melhor desempenho global, alcançando valores de *recall* superiores a 0.68 na classe *Fear* no dataset FER-2013, o que demonstra elevada sensibilidade para esta emoção crítica. Em contraste, o modelo *FER* [66] revelou fortes limitações na mesma classe, com valores de *recall* bastante reduzidos no dataset RAF-DB (inferiores a 0.30), indicando dificuldade na deteção de situações de medo. Para a classe *Angry*, o *FER* mostrou-se mais consistente, mas ainda assim o *DeepFace* obteve valores superiores de *F1-score*, confirmando a sua maior robustez geral.

Estes resultados encontram-se sintetizados na Tabela 5.4, que apresenta as métricas de desempenho por modelo e emoção, testadas sobre os dois datasets. A diferença de comportamento entre os modelos pode ser explorada pelo sistema através da regra de decisão por *scoring*, ajustando o contributo de cada um. Apenas com estes dados, seria recomendável atribuir ao *DeepFace* um peso ligeiramente superior, dada a sua maior capacidade de identificar a emoção *Fear*, considerada crítica em cenários de risco.

Tabela 5.4: Desempenho dos modelos de reconhecimento de emoções nas classes *Angry* e *Fear*.

Modelo	Dataset	Emoção	Precisão	Recall	F1-score
DeepFace [59]	FER-2013 [78]	Angry	0.918	0.696	0.792
	FER-2013 [78]	Fear	0.927	0.689	0.791
	RAF-DB [79]	Angry	0.898	0.435	0.586
	RAF-DB [79]	Fear	0.426	0.132	0.201
FER [66]	FER-2013 [78]	Angry	0.694	0.384	0.494
	FER-2013 [78]	Fear	0.708	0.262	0.233
	RAF-DB [79]	Angry	0.790	0.454	0.576
	RAF-DB [79]	Fear	0.937	0.053	0.101

Importa referir que o modelo FER foi treinado no FER-2013, o que pode enviesar a sua avaliação nesse mesmo conjunto. Para mitigar esse efeito recorreu-se ao RAF-DB, que permitiu avaliar o comportamento do modelo. Ainda assim, o FER obteve resultados fracos, refletindo limitações da sua arquitetura e confirmando problemas já descritos na literatura relativamente ao FER-2013, como a baixa resolução das imagens e o desequilíbrio entre classes [78, 9].

### 5.1.5 Avaliação do Modelo de Detecção de Ambiente Violento

A avaliação da capacidade do sistema para identificar comportamentos violentos foi realizada com base no modelo pré-treinado *ViolenceModel* [67], que se encontra integrado no Sistema de Segurança como serviço externo. O teste foi conduzido com recurso ao conjunto de dados RWF-2000 [10], que é constituído por cerca de 2000 vídeos de câmaras de vigilância, equilibrados entre sequências violentas e não violentas. Para efeitos desta avaliação, foi utilizada a partição de validação disponibilizada pelo dataset, composta por 400 vídeos distintos (200 violentos e 200 não violentos). Durante os testes, cada vídeo foi processado com extração de frames, sendo considerado apenas um frame a cada dez, de forma a reduzir o número total de imagens analisadas. Esses frames foram posteriormente agrupados em pequenos segmentos correspondentes a intervalos curtos do vídeo. Cada frame foi classificado individualmente pelo modelo como *Violence* ou *NoViolence*, e a decisão de cada segmento resultou de um voto maioritário sobre as classificações dos seus frames. Por fim, a decisão ao nível do vídeo foi obtida de forma permissiva: um vídeo foi considerado violento sempre que pelo menos um dos seus segmentos fosse classificado como tal. Esta abordagem permite reduzir o impacto de erros em frames isolados e aproxima a avaliação da noção de evento, refletindo de forma mais adequada a natureza sequencial dos vídeos.

## Resultados

A Tabela 5.5 apresenta o desempenho do modelo *ViolenceModel* [67] após aplicação da estratégia de agregação temporal, em que a decisão final é obtida ao nível do vídeo. Os resultados revelam valores equilibrados de precisão e *recall* (ambos próximos de 0.75), demonstrando que o modelo consegue identificar uma parte significativa dos vídeos violentos sem comprometer excessivamente a taxa de falsos positivos. Esta abordagem mostrou-se mais adequada do que a análise isolada de frames ou segmentos, permitindo reduzir o impacto de erros pontuais e refletir melhor a noção de evento característica de cenários de segurança. A matriz de confusão representada na Figura 5.1 evidencia este comportamento,

mostrando que as principais falhas continuam a corresponder a episódios violentos não detetados, embora em menor número face a abordagens mais simplistas.

Tabela 5.5: Desempenho do modelo *ViolenceModel* ao *dataset* RWF-2000 [10].

Modelo	Dataset	Precisão	Recall	F1-score	FPR	FNR
ViolenceModel [67]	RWF-2000 [10]	0,75	0,74	0,75	0,25	0,26

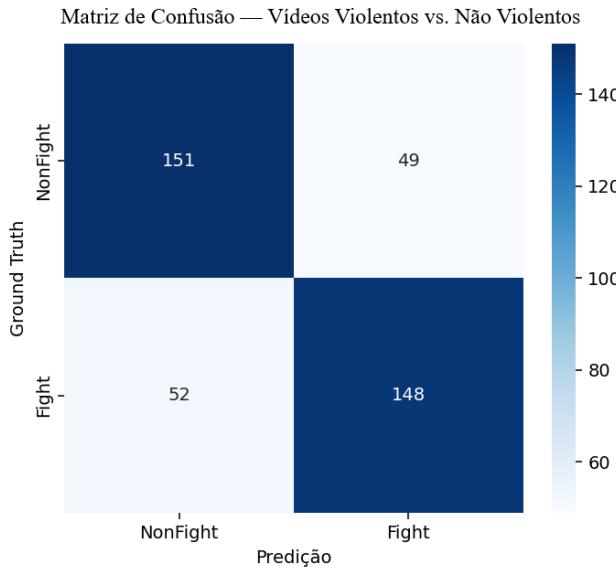


Figura 5.1: Matriz de confusão resultante da avaliação do modelo *ViolenceModel* no *dataset* RWF-2000 [10].

## 5.2 Validação do Sistema de Segurança

Conforme previsto no plano de trabalho, foi realizada uma etapa dedicada à **validação global do sistema de segurança**. Esta fase teve como objetivo analisar o funcionamento do sistema em cenários representativos do seu contexto de aplicação.

### 5.2.1 Criação do Dataset de Validação

Para validar o comportamento do sistema de segurança em condições próximas da realidade, foi criado um *dataset* de validação com imagens classificadas em duas classes: *alarme* (positivas) e *não alarme* (negativas). O objetivo foi garantir que o conjunto de dados refletisse contextos compatíveis com o ambiente de uma máquina de autoatendimento.

A construção do *dataset* baseou-se em três fontes complementares:

- **Seleção manual em datasets públicos:** foram escolhidas imagens de bases de dados já existentes, selecionadas uma a uma de forma a corresponderem a condições visuais compatíveis com o cenário de máquinas de autoatendimento. Este processo incluiu tanto exemplos positivos (situações com armas, expressões de medo ou raiva, ou violência) como negativos (contextos neutros sem risco).

Para reforçar estes últimos, recorreram-se também a imagens do COCO Dataset [47]. Esta foi a principal fonte do dataset, reunindo a maioria dos exemplos positivos e negativos.

- **Aquisição através do Hefesto:** utilizou-se a câmara da própria máquina de autoatendimento para registar simulações de cenários de alarme e de utilização normal. Embora não representem o maior volume do dataset, estas imagens foram importantes por refletirem o contexto mais próximo das condições reais de funcionamento que foi possível obter.
- **Imagens sintéticas geradas por IA:** para reforçar a diversidade de casos positivos, recorreu-se a ferramentas de geração de imagens que produziram exemplos adicionais de situações de risco adaptadas ao contexto do sistema.

No total, o *dataset* integra 405 imagens provenientes de diferentes origens. A Figura 5.2 apresenta a respetiva distribuição por fonte, mostrando que a maioria corresponde a *datasets* públicos (cerca de 74% do total), enquanto as simulações realizadas no Hefesto representam aproximadamente 15% e as imagens sintéticas geradas por IA cerca de 11%. Esta composição garante uma predominância de exemplos reais, mas reforçada por casos simulados e gerados artificialmente, o que contribui para uma maior diversidade e adequação ao contexto de aplicação.

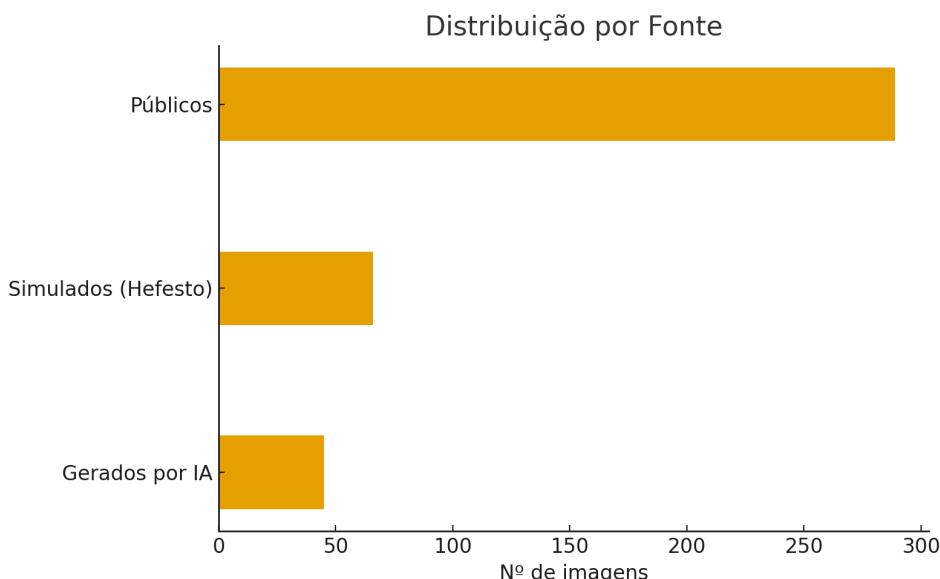


Figura 5.2: Distribuição do número de imagens do *dataset* de validação por fonte.

### Critérios de classificação

O *dataset* foi estruturado como uma tarefa de classificação binária, com duas classes (*alarme* e *não alarme*), sendo as imagens rotuladas manualmente através da sua organização em pastas correspondentes. Uma imagem foi classificada como *alarme* sempre que incluía pelo menos um dos indícios de ameaça descritos na Tabela 5.6, sendo atribuída à classe *não alarme* quando correspondia a situações de utilização normal igualmente definidas na tabela. Importa salientar que fatores como condições de iluminação, qualidade da imagem ou se o local era interior/exterior não foram considerados na definição dos critérios de classificação.

Classe	Condições para classificação
<i>Alarme</i>	<ul style="list-style-type: none"> <li>• Arma de fogo visível, segurada ou apontada.</li> <li>• Arma branca (faca, navalha, etc.) segurada na mão.</li> <li>• Agressão ou ameaça explícita (socos, estrangulamento, imobilização violenta ou gesto de apontar arma/objeto).</li> <li>• Expressão de medo ou raiva associada a contexto de ameaça (não considerada isoladamente).</li> </ul>
<i>Não alarme</i>	<ul style="list-style-type: none"> <li>• Utilização normal da máquina de autoatendimento.</li> <li>• Pessoas em enquadramentos próximos, compatíveis com a perspectiva da câmara.</li> <li>• Indivíduos a segurar objetos usuais (telemóveis, carteiras, chaves, documentos).</li> <li>• Posturas e expressões neutras, sem sinais de ameaça.</li> <li>• Presença de objetos semelhantes a armas, ou de brinquedos e ilustrações sem intenção hostil.</li> </ul>

Tabela 5.6: Critérios de classificação utilizados nas imagens do *dataset* de validação.

A aplicação destes critérios permitiu assegurar uma atribuição consistente dos rótulos, evitando a inclusão de casos ambíguos ou de difícil interpretação. No final, a distribuição por classes ficou equilibrada, com aproximadamente metade das imagens a corresponder à classe *alarme* e a outra metade à classe *não alarme*, conforme ilustrado na Figura 5.3.

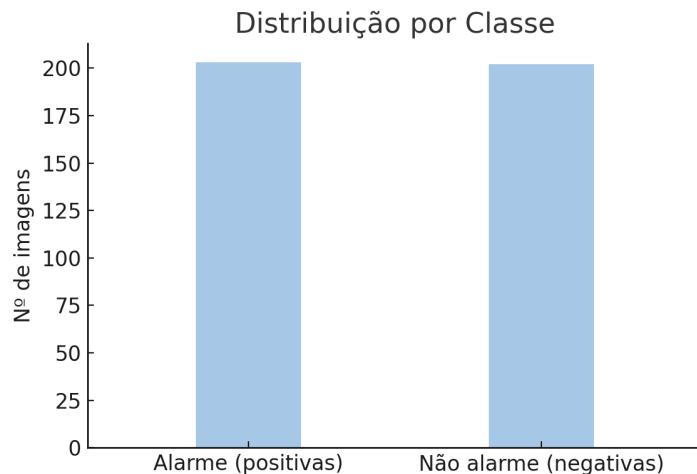


Figura 5.3: Distribuição do número de imagens do *dataset* de validação por classe.

Para além da caracterização numérica, a Figura 5.4 apresenta exemplos representativos de ambas as

classes, ilustrando a diversidade de situações consideradas no *dataset*.



Figura 5.4: Exemplos de imagens incluídas no *dataset* de validação, contendo casos de *alarme* (situações de risco) e de *não alarme* (utilização normal).

### Verificação de duplicados e mitigação de *leakage*

Um dos principais riscos associados à seleção manual de imagens em *datasets* públicos reside na presença de imagens replicadas ou quase idênticas, frequentemente redistribuídas em múltiplos repositórios. Este risco é agravado pelo facto de diferentes conjuntos de dados partilharem subconjuntos visuais iguais sem indicação explícita dessa sobreposição. Tal situação origina fenómenos de *data leakage*, conduzindo a métricas de validação artificialmente inflacionadas e a conclusões que não refletem a verdadeira capacidade de generalização do sistema.

Para mitigar este risco, foram aplicadas técnicas de deteção de duplicados às imagens provenientes de conjuntos de dados públicos, removendo instâncias redundantes sempre que foram identificados pares semelhantes. As abordagens utilizadas foram as seguintes:

- **Hashes perceptuais (pHash):** cada imagem foi convertida num *hash* que resume o seu conteúdo visual. Quando dois *hashes* diferiam em poucos bits (distância de Hamming até 8), as imagens foram consideradas praticamente iguais e uma delas foi removida;
- **Embeddings:** cada imagem foi representada por um vetor (*embedding*) gerado pelo modelo *Contrastive Language–Image Pre-training* (CLIP). Se a semelhança entre dois vetores, medida pelo cosseno, fosse igual ou superior a 0.92, as imagens foram consideradas variantes quase idênticas e apenas uma foi mantida.

Os resultados deste processo de deduplicação mostram o impacto direto sobre o subconjunto de imagens provenientes de fontes públicas. O número de exemplos positivos foi reduzido em cerca de 37%, passando de 139 para 87 instâncias, refletindo a existência de várias situações redundantes de risco que foram eliminadas. Já o número de exemplos negativos manteve-se inalterado em 150 imagens, uma

vez que estas correspondiam a situações não alarmantes variadas, sem elementos visuais suficientemente semelhantes que justificassem a sua remoção. Estes resultados encontram-se resumidos na Tabela 5.7.

Importa referir que não foi identificada necessidade de aplicar um processo de deduplicação às restantes fontes. No caso das imagens encenadas, todas as situações foram captadas intencionalmente para representar alguns cenários, pelo que a sua preservação integral assegura maior realismo contextual. Já no caso das imagens geradas por inteligência artificial, a diversidade obtida foi considerada suficiente para os objetivos definidos, não se tendo observado redundâncias significativas que justificassem um processo adicional de filtragem nesta fase.

Tabela 5.7: Impacto da deduplicação no subconjunto de imagens de fontes públicas.

	<b>Antes da deduplicação</b>	<b>Após a deduplicação</b>
Positivas (alarme)	139	87
Negativas (não alarme)	150	150

Considerando todas as fontes, o *dataset* de validação passou a integrar um total de 348 imagens após a deduplicação, em contraste com as 405 imagens inicialmente reunidas. A Figura 5.5 apresenta a nova distribuição por origem, confirmando que a redução incidiu exclusivamente sobre as imagens provenientes de *datasets* públicos.

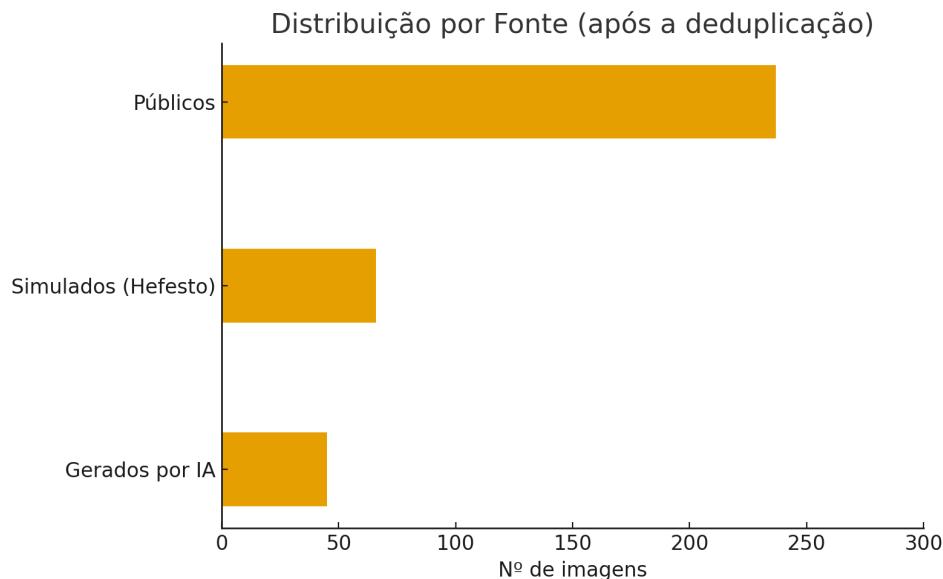


Figura 5.5: Distribuição do número de imagens por fonte após a deduplicação.

## 5.2.2 Avaliação dos Modelos por Origem dos Dados

Com o *dataset* de validação devidamente rotulado (*alarme* ou *não alarme*) e organizado em três subconjuntos distintos (*público*, *encenado* e *sintéticos/gerados por IA*), todas as imagens foram processadas pelos diferentes modelos integrados no sistema. Os resultados foram mantidos de forma separada para cada subconjunto, permitindo avaliar e comparar o desempenho dos modelos em contextos distintos.

De cada saída obtida foram extraídos apenas os valores de interesse para a validação, tendo em conta as classes disponibilizadas pelos próprios modelos:

- **Modelos de deteção de armas** (*YoloA*, *YoloB*, *YoloC* e *Efficient-net*): cada um com o seu conjunto interno de classes possíveis, mas apenas foram retidas as maiores confianças associadas às classes consideradas alarmantes, como *gun*, *knife* ou *pistol*.
- **Modelo de deteção de violência** (*Modelo Violence*): entre as classes previstas por este modelo, foi considerada exclusivamente a confiança relativa à classe *violence*, definida como alarmante.
- **Modelos de classificação facial** (*DeepFace* e *FER*): embora produzam probabilidades para várias expressões faciais, apenas foram preservadas as correspondentes às classes *fear* e *angry*, previamente identificadas como alarmantes.

Em paralelo, foi igualmente preservada a anotação de referência (*ground truth*) de cada imagem, indicando a sua pertença ao conjunto positivo ou negativo, assegurando uma base consistente para a avaliação comparativa do desempenho dos modelos.

### Validação Leave-Source-Out

Para avaliar a capacidade de generalização dos modelos em contextos distintos, foi adotada a estratégia de *leave-source-out*. Em vez de considerar o *dataset* de validação como um todo indiferenciado, as imagens deste conjunto foram analisadas separadamente por origem, considerando em particular as imagens públicas, as imagens captadas no ambiente encenado Hefesto e as imagens sintéticas geradas por inteligência artificial. Esta abordagem permite identificar de que forma a proveniência dos dados influencia o desempenho dos modelos e em que medida estes se adaptam a condições variadas.

Na Tabela 5.8 apresentam-se as métricas clássicas de avaliação (acurácia, precisão, *recall* e F1-score) obtidas para todos os modelos integrados no sistema, discriminadas por subconjunto de validação.

Tabela 5.8: Métricas de classificação por modelo e subconjunto de validação (*leave-source-out*).

Subconjunto	Modelo	Acurácia	Precisão	Recall	F1
Encenado	YoloA	0.619	1.000	<b>0.455</b>	0.625
Encenado	YoloB	0.349	1.000	0.068	0.128
Encenado	YoloC	0.397	1.000	0.136	0.240
Encenado	Efficient-net	0.302	0.000	0.000	0.000
Encenado	DeepFace	0.254	0.000	0.000	0.000
Encenado	Fer	0.366	1.000	0.091	0.167
Encenado	Violence	0.301	0.000	0.000	0.000
Público	YoloA	0.730	0.658	<b>0.552</b>	0.600
Público	YoloB	0.624	0.450	0.103	0.168
Público	YoloC	0.684	0.620	0.357	0.453
Público	Efficient-net	0.797	0.868	<b>0.528</b>	0.657
Público	DeepFace	0.565	0.400	0.368	0.383
Público	Fer	0.599	0.250	0.046	0.077
Público	Violence	0.717	0.955	0.241	0.385
Sintético	YoloA	0.586	1.000	0.393	0.564
Sintético	YoloB	0.439	1.000	0.179	0.303
Sintético	YoloC	0.586	1.000	0.393	0.564
Sintético	Efficient-net	0.293	0.000	0.000	0.000
Sintético	DeepFace	0.610	0.833	<b>0.536</b>	0.652
Sintético	Fer	0.463	1.000	0.214	0.353
Sintético	Violence	0.317	0.000	0.000	0.000

A análise da Tabela 5.8 evidencia que o desempenho dos modelos varia significativamente tanto em função da origem das imagens como do modelo.

No subconjunto **Encenado**, vários modelos apresentaram precisão máxima (1.0), como o *YoloA*, *YoloB*, *YoloC* e *FER*. Este valor significa que, sempre que estes modelos detetaram classes consideradas alarmantes, a previsão correspondeu a um verdadeiro positivo. Contudo, os respetivos valores de *recall* foram bastante reduzidos (variando entre 0.07 e 0.45), o que demonstra que apenas uma fração limitada dos casos alarmantes foi identificada. Esta combinação traduz um comportamento altamente conservador: o modelo acerta quando deteta algo alarmante, mas fá-lo em muito poucas ocasiões. Já os modelos *EfficientNet*, *DeepFace* e *Violence* praticamente não detetaram classes alarmantes neste subconjunto, apresentando métricas nulas em termos de precisão, *recall* e F1. Apesar disso, é possível observar valores de acurácia não nulos: isto ocorre porque a acurácia considera também os acertos na classe negativa (ou seja, os casos sem alarme). Assim, mesmo não identificando classes alarmantes, estes modelos acertaram na classificação de várias instâncias negativas, o que explica a acurácia acima de zero. No caso específico do modelo *Violence*, este resultado poderá estar associado ao número reduzido de exemplos de situações de violência explícita (como confrontos físicos ou lutas) disponíveis neste subconjunto, limitando assim a sua capacidade de deteção.

No subconjunto **Público**, observa-se um maior equilíbrio entre precisão e *recall*, destacando-se os modelos *EfficientNet* e *YoloA*, que obtiveram valores de F1 de 0.657 e 0.600, respectivamente, sugerindo uma capacidade de deteção mais consistente neste tipo de imagens. Em contrapartida, o modelo *FER* registou um desempenho bastante fraco (F1 = 0.077), refletindo grande dificuldade em generalizar para

este conjunto, enquanto o modelo *Violence*, apesar de ter atingido uma precisão muito elevada (0.955), apresentou um *recall* bastante reduzido (0.241), evidenciando uma vez mais um perfil de deteção conservador.

Nas imagens **Sintéticas**, o comportamento é mais heterogéneo, com alguns modelos a apresentarem precisão máxima (*YoloA*, *YoloB*, *YoloC* e *FER*), mas acompanhada de valores de *recall* bastante reduzidos. O modelo *DeepFace* destacou-se neste subconjunto por atingir um equilíbrio relativamente bom entre precisão (0.833) e *recall* (0.536), resultando num F1 de 0.652, enquanto os modelos *EfficientNet* e *Violence* não detetaram classes alarmantes, apresentando valores nulos em todas as métricas e, consequentemente, desempenho nulo.

Em síntese, os resultados variam consoante a origem das imagens: no subconjunto Encenado observa-se elevada precisão mas baixo *recall*, no Público um maior equilíbrio e nas Sintéticas um desempenho mais irregular. Destaca-se o Encenado, por reproduzir de forma controlada condições próximas da utilização real, podendo assim contribuir mais diretamente para a definição dos pesos dos modelos numa futura configuração do sistema por *scoring*.

### 5.2.3 Avaliação da Influência dos Modelos

Nesta fase, o *dataset* de validação foi considerado como um todo, sem distinção entre subconjuntos de origem. Todas as imagens foram processadas pelos modelos integrados no sistema, preservando-se a anotação de referência (*alarme* ou *não alarme*) e apenas os valores mais relevantes das suas saídas: nos modelos de deteção, as confianças associadas às classes alarmantes e, nos classificadores, as probabilidades correspondentes às classes de interesse.

Estes valores numéricos foram utilizados como variáveis de entrada num modelo auxiliar de aprendizagem automática. O objetivo não foi substituir a lógica interna do sistema, mas sim avaliar a influência relativa de cada modelo no resultado final (*alarme* ou *não alarme*). Para esse efeito, recorreu-se a um *Random Forest*, uma vez que se trata de um método robusto e interpretável, amplamente utilizado quando se pretende analisar o contributo das variáveis para a decisão. Uma das suas vantagens é fornecer de forma direta uma estimativa de importância, permitindo avaliar o grau em que cada variável contribui para a distinção entre *alarme* e *não alarme*. Assim, a importância pode ser entendida como uma medida do peso relativo de cada modelo no processo de decisão do sistema.

As saídas dos modelos foram agregadas numa matriz de características, em que cada linha representa uma imagem e cada coluna corresponde à saída de um modelo (confiança na deteção ou probabilidade de classe). Essa matriz foi emparelhada com a classe de referência de cada imagem (*alarme* ou *não alarme*) e serviu de base ao treino do *Random Forest*. Para assegurar maior robustez estatística, recorreu-se a validação cruzada 5×2, ou seja, cinco repetições de partições estratificadas em dois subconjuntos treino/teste, totalizando dez avaliações independentes. Com este procedimento, todos os exemplos participam em treino e teste várias vezes, reduzindo o risco de que uma única divisão influencie de forma desproporcionada os resultados.

Além disso, as métricas foram avaliadas com recurso a *bootstrap* (10 000 reamostragens), é uma técnica que consiste em voltar a sortear, com reposição, os valores originais várias vezes, que simula diferentes conjuntos possíveis. A partir dessa distribuição calcula-se a média e Intervalo de Confiança (IC) 95%, permitindo quantificar de forma rigorosa a variabilidade e a incerteza associadas ao desem-

penho do sistema. Os valores obtidos encontram-se na Tabela 5.9, onde se apresentam as métricas *Recall*, *FNR* e *F1* com os respectivos IC95%, formato que facilita a comparação direta dos indicadores e a avaliação da estabilidade dos resultados.

Tabela 5.9: Resultados médios das métricas com IC95% (5x2 CV + bootstrap).

Métrica	Média	IC95%	Interpretação
Recall	62.6%	[61.3%, 63.9%]	Proporção de alarmes corretamente detetados
FNR	37.4%	[36.1%, 38.7%]	Proporção de alarmes não detetados (falsos negativos)
F1	68.2%	[67.7%, 68.7%]	Equilíbrio entre precisão e recall

Para além das métricas globais, que permitem avaliar o desempenho médio do sistema e a respetiva estabilidade, interessa compreender de forma mais detalhada quais os modelos que mais contribuem para a decisão. Assim, o *Random Forest* fornece ainda uma estimativa direta da importância relativa de cada variável, permitindo identificar de forma objetiva quais os detetores com maior peso no processo de classificação. A Figura 5.6 apresenta esses resultados, onde se observa a média das importâncias atribuídas a cada modelo ao longo das dez execuções da validação cruzada, incluindo o respetivo desvio padrão.

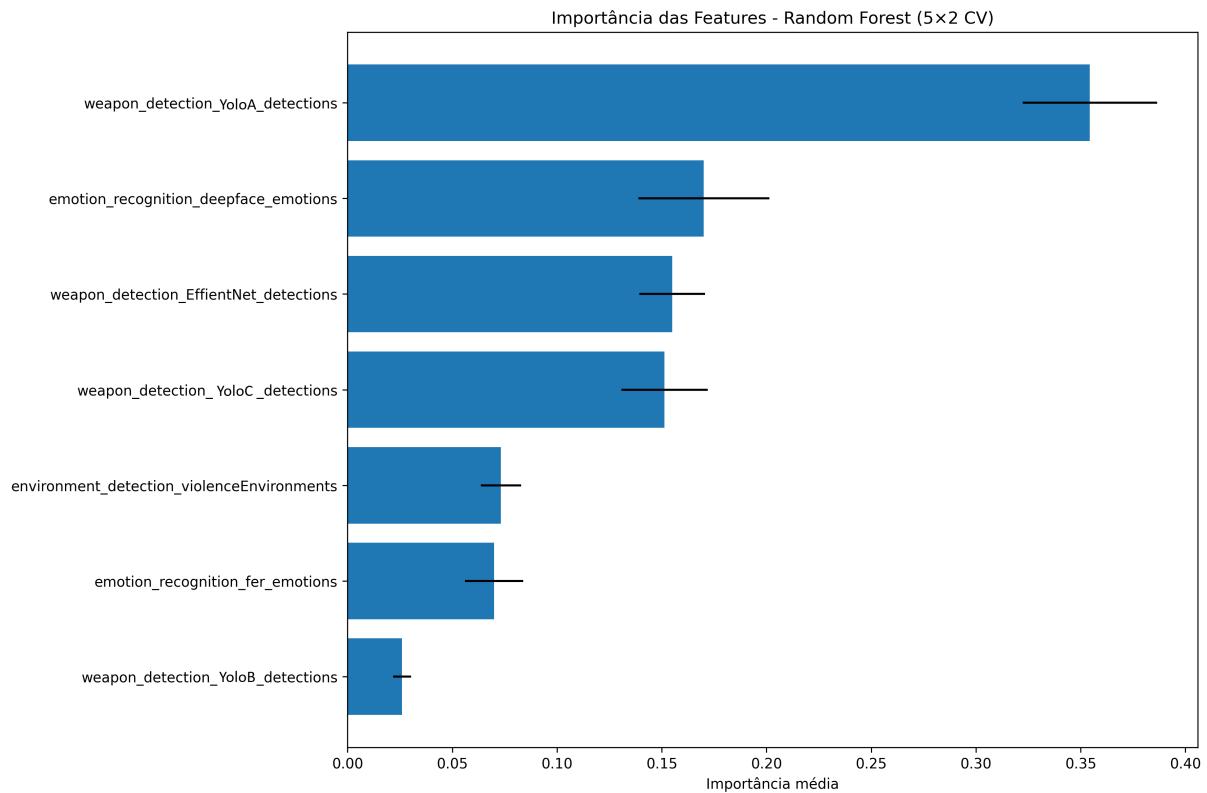


Figura 5.6: Importância média das *features* (modelos) segundo o Random Forest, com desvio padrão.

Os resultados da Figura 5.6 evidenciam uma clara predominância dos modelos de detecção de armas, em particular o *YoloA* [62], que surge como a variável mais determinante na classificação. Outros detetores de armas, como o EfficientNet [65] e o YoloC [64], mantêm também relevância considerável, reforçando o peso desta categoria no processo de decisão. Entre os classificadores, o modelo de reco-

nhecimento de emoções baseado no *DeepFace* destaca-se com importância intermédia, sugerindo que indicadores emocionais contribuem de forma complementar, enquanto o modelo FER e o de violência apresentam impacto mais reduzido.

Esta distribuição de importâncias é coerente com os resultados médios das métricas apresentados na Tabela 5.9. Apesar da forte influência dos modelos da categoria de armas, o sistema atinge apenas um *Recall* global de 62.6%, acompanhado de uma taxa de falsos negativos de 37.4%. Tal significa que, quando existem armas visíveis, o desempenho é elevado, mas em cenários alternativos, como episódios de violência sem armas ou sinais emocionais mais subtils, a decisão perde robustez, originando falhas de deteção. O valor de F1 (68.2%) confirma a consistência do compromisso entre precisão e *Recall*, enquanto os intervalos de confiança estreitos em todas as métricas demonstram que este padrão não é um artefacto de uma divisão particular dos dados, mas sim um comportamento consistente. Em síntese, a análise conjunta das métricas e das importâncias revela um sistema fortemente dependente dos modelos de armas, o que pode garantir boas prestações nesses contextos mas limita a capacidade de generalização para outros tipos de ameaça.

Importa salientar que o *Random Forest* foi utilizado apenas com fins analíticos, para avaliar a influência relativa das variáveis. A sua adoção como mecanismo de decisão no sistema não seria adequado, uma vez que implicaria um conjunto fixo de modelos ativos, em contraste com a flexibilidade do sistema que permite ativar, desativar ou substituir modelos livremente através do ficheiro de configuração (*config.json*).

### Análise complementar dos modelos com base no AUC

Para além da análise de influência obtida com o *Random Forest*, procedeu-se a uma avaliação direta do desempenho individual dos modelos, recorrendo ao cálculo da métrica AUC no dataset de validação. Esta análise funciona como complemento e validação da perspetiva anterior, permitindo medir de forma independente a capacidade discriminativa de cada modelo em relação à sua tarefa específica. Convém esclarecer que um AUC de 0.5 corresponde ao desempenho esperado por mero acaso, isto é, equivalente a uma classificação aleatória sem qualquer capacidade real de distinção. Valores significativamente acima deste nível indicam maior capacidade discriminativa, enquanto valores próximos de 0.5 revelam contributo reduzido.

Na categoria de armas, os valores de AUC refletem a capacidade dos detetores em distinguir entre imagens que contêm armas e aquelas que não apresentam armas. Os resultados apresentados na Tabela 5.10 evidenciam uma hierarquia clara no interior da categoria: o modelo *YoloA* destaca-se com um AUC de 0.769, demonstrando elevada robustez, seguido pelo *YoloC* (0.648) e pelo *EfficientNet* (0.589), que revelam contributos intermédios. Já o *YoloB* (0.518) apresenta um valor muito próximo do nível de acaso, sugerindo utilidade reduzida.

Na categoria de emoções, os valores de AUC dizem respeito à distinção entre imagens em que se observam predominantemente as emoções de medo ou raiva e aquelas que exibem outros estados emocionais. Os resultados mostram que ambos os classificadores têm desempenho limitado, apenas ligeiramente superior ao acaso, com o *DeepFace* a alcançar 0.560 e o *FER* 0.533. Este padrão indica que, considerados isoladamente, os modelos de emoções têm fraca capacidade discriminativa, embora possam ainda fornecer contributos complementares quando combinados com os detetores de armas.

Esta análise não foi repetida para a categoria de violência, uma vez que nela existe apenas um único modelo. Não se justificaria a apresentação de uma comparação interna, dado que o objetivo desta subsubsecção é evidenciar as diferenças de desempenho entre modelos pertencentes à mesma categoria.

Tabela 5.10: Valores de AUC dos modelos das categorias de armas e emoções no dataset de validação.

Categoría	Modelo	AUC
Armas	YoloA	0.769
Armas	YoloB	0.518
Armas	YoloC	0.648
Armas	EfficientNet	0.589
Emoções	DeepFace	0.560
Emoções	FER	0.533

### 5.2.4 Ajuste dos Pesos no Ficheiro de Configuração

#### Análise de correlação entre modelos

Antes de ajustar os pesos no ficheiro *config.json*, avaliou-se a existência de redundância entre os modelos, isto é, se os seus valores de confiança estavam altamente correlacionados. Esta análise é relevante porque na regra de decisão por *scoring* a soma dos pesos dos modelos é comparada com um limiar e, nesse contexto, sinais muito semelhantes provenientes de vários modelos podem levar a que cada um contribua para a soma e acabem por facilitar o disparo da categoria. Quando o limiar é ultrapassado a categoria passa a ser considerada alarmante e contribui apenas com o peso fixo que lhe está associado independentemente da soma dos pesos dos seus modelos. Para além disso, esta verificação da correlação permite identificar possíveis modelos redundantes que pouco acrescentam à decisão, possibilitando a sua remoção e tornando o sistema mais eficiente.

Para esse efeito, calcularam-se as correlações entre as saídas dos modelos sobre as imagens do *dataset* de validação, considerando como variáveis as confianças ou probabilidades associadas às classes alarmantes. Foram aplicados dois coeficientes complementares:

- **Correlação de Pearson**, coeficiente que mede a força e a direção da relação linear entre duas variáveis numéricas.
- **Correlação de Spearman**, coeficiente que avalia a relação monotónica entre duas variáveis, isto é, se tendem a variar no mesmo sentido, considerando apenas a ordenação dos valores.

Valores próximos de 1 indicam forte correlação positiva (os modelos tendem a dar pontuações semelhantes), valores próximos de 0 indicam ausência de relação, e valores negativos sugerem que os modelos respondem de forma inversa. Na literatura, valores superiores a 0.8 são geralmente considerados como sinal de correlação forte, justificando o agrupamento dos modelos para evitar redundância [80].

A Tabela 5.11 apresenta os resultados obtidos para os principais pares de modelos dentro de cada categoria. Observa-se que, mesmo entre os modelos de armas, as correlações se mantêm baixas, e uma correlação baixa entre modelos pode ser interpretada de duas formas. Por um lado, sugere que os modelos estão a cometer erros diferentes: por exemplo, um detetor pode ser mais sensível a armas

pequenas mas falhar em armas grandes, outro pode ter maior robustez a oclusões mas ser afetado por iluminação, e assim sucessivamente. Esta diversidade é benéfica, já que pode aumentar a cobertura em diversos cenários. Por outro lado, uma correlação baixa pode também refletir instabilidade ou ruído, isto é, um modelo que erra de forma aleatória. Nesse caso, a diversidade não acrescenta valor e pode mesmo introduzir ruído desnecessário.

Nos classificadores de emoções, a baixa correlação entre o DeepFace e o FER em angry e fear confirma que exploram sinais distintos, como esperado pelas suas arquiteturas diferentes. Acresce que estas duas emoções partilham características visuais e significado parcialmente sobrepostos, o que acentua a divergência entre modelos.

Tabela 5.11: Correlação entre pares de modelos segundo os coeficientes de Pearson e Spearman.

Modelo A	Modelo B	Pearson	Spearman
YoloA (armas)	YoloB (armas)	0.22	0.23
YoloA (armas)	YoloC (armas)	0.24	0.24
YoloA (armas)	EfficientNet (armas)	0.21	0.22
YoloB (armas)	YoloC (armas)	0.19	0.21
YoloB (armas)	EfficientNet (armas)	0.22	0.24
YoloC (armas)	EfficientNet (armas)	0.15	0.15
DeepFace (emoções)	FER (emoções)	0.39	0.37

### Critérios para definição de pesos

Os pesos utilizados no sistema foram definidos em dois níveis distintos, que não devem ser confundidos:

- **Pesos das categorias:** definidos em função das prioridades do cliente, com possibilidade de ajuste em diferentes cenários de aplicação, assegurando a devida influência de cada categoria na decisão global do sistema.
- **Pesos dos modelos dentro de cada categoria:** atribuídos de acordo com o desempenho individual de cada modelo no *dataset* de validação, de forma aproximadamente proporcional ao respetivo *recall*.

Na presente configuração, os pesos das categorias, usados na decisão global, foram definidos segundo os requisitos da **INM**, que solicitaram maior ênfase na categoria **armas**, por representar a ameaça de maior relevância no setor bancário. As categorias de **emoção** e **violência**, por sua vez, foram consideradas de relevância equivalente, tendo sido atribuídos pesos iguais na configuração do sistema.

O limiar da decisão global foi definido de forma a garantir que a categoria de deteção de armas, por si só, é suficiente para acionar um alarme, enquanto as restantes categorias podem também contribuir para atingir o mesmo efeito quando ocorrem em conjunto. Já a atribuição dos pesos aos modelos dentro de cada categoria baseou-se em três elementos complementares:

1. **Importâncias estimadas pelo Random Forest e análise de AUC:** utilizadas como referência para avaliar a relevância relativa dos modelos. O *Random Forest* permitiu identificar a influência das variáveis no funcionamento global, enquanto o AUC mediu a capacidade discriminativa de cada modelo no *dataset* de validação.

2. **Desempenho individual de cada modelo (com foco no *recall*):** considerada a métrica central para atribuição de pesos. Os valores de *recall* serviram como base para a definição de pesos proporcionais, refletindo diretamente a sensibilidade de cada modelo na deteção de situações de risco. Esta análise foi realizada tanto na avaliação isolada (Sec. 5.1.2) como nos subconjuntos do *dataset* de validação (Sec. 5.2.2).
3. **Calibração dos parâmetros `thresholdDetection`:** preenchidos com base na análise da confiança média dos *True Positives* de cada modelo, de forma a equilibrar sensibilidade e fiabilidade na decisão.

Assim, os pesos foram estabelecidos de forma proporcional ao *recall*, concluindo esta etapa de definição e introduzindo a secção seguinte, dedicada à definição dos *thresholds* das categorias.

### Definição dos limiares de decisão por categoria

A determinação dos *thresholds* por categoria baseou-se nas pontuações agregadas de cada grupo de modelos. Para cada imagem, a pontuação categorial foi calculada como a soma dos pesos dos modelos cuja confiança excedeu o respetivo `thresholdDetection`. Os pesos foram definidos a partir do *recall* médio de cada modelo, conforme descrito na Sec. 5.2.4, assegurando que modelos mais sensíveis à deteção de situações de risco tenham maior influência na decisão final.

Com estas pontuações, foram construídas curvas ROC ao nível de cada categoria, avaliando a área sob a curva (AUC) como medida da capacidade discriminativa. O *threshold* ótimo foi determinado através do  $J$ , métrica amplamente utilizada em classificação binária que identifica o ponto da curva ROC que maximiza simultaneamente a sensibilidade (TPR) e a especificidade (1-FPR). Formalmente, o índice é definido como  $J = \text{TPR} - \text{FPR}$ , sendo escolhido o limiar correspondente ao valor máximo desta diferença.

Na categoria **armas**, foram considerados quatro modelos com pesos proporcionais ao *recall* (`YoloA=0.467`, `YoloB=0.117`, `YoloC=0.295` e `EfficientNet=0.259`). A curva ROC apresentou **AUC = 0.750** e o índice de Youden indicou um limiar ótimo de  $T = 0.259$  (ou  $\tilde{T} \approx 0.228$  na versão normalizada), valor a partir do qual a categoria de armas passa a ser considerada alarmante, com **TPR = 0.698** e **FPR = 0.253**, conforme ilustrado na Figura 5.7. Este limiar foi considerado adequado no contexto bancário, privilegiando a sensibilidade à deteção de armas mesmo ao custo de mais falsos positivos.

Na categoria **emoções**, com dois modelos (`deepface=0.3013` e `fer=0.117`), a curva ROC, apresentada na Figura 5.7, mostra **AUC = 0.505**, praticamente sobreposta à linha diagonal de referência ( $AUC = 0.5$ ). Tal resultado indica ausência de capacidade discriminativa, ou seja, o comportamento da categoria aproxima-se de uma decisão aleatória. O limiar ótimo foi definido em  $T = 0.117$  (ou  $\tilde{T} \approx 0.280$ ), correspondendo a **TPR = 0.346** e **FPR = 0.335**. Apesar de formalmente definido, este resultado reforça a necessidade de explorar modelos alternativos ou dados adicionais em trabalhos futuros para melhorar a utilidade prática desta categoria.

Por fim, na categoria **violência**, composta por um único modelo, foi atribuído um peso  $w$  correspondente ao seu *recall*, de modo a manter consistência de escala com as restantes categorias. Assim, o *score* categorial assume valores em  $\{0, w\}$  e o limiar ótimo é, por definição,  $T = w$ . Na prática, a categoria

é considerada alarmante sempre que o modelo emite uma deteção, e, por se tratar de um caso binário, a curva ROC não acrescenta informação relevante e não é aqui apresentada.

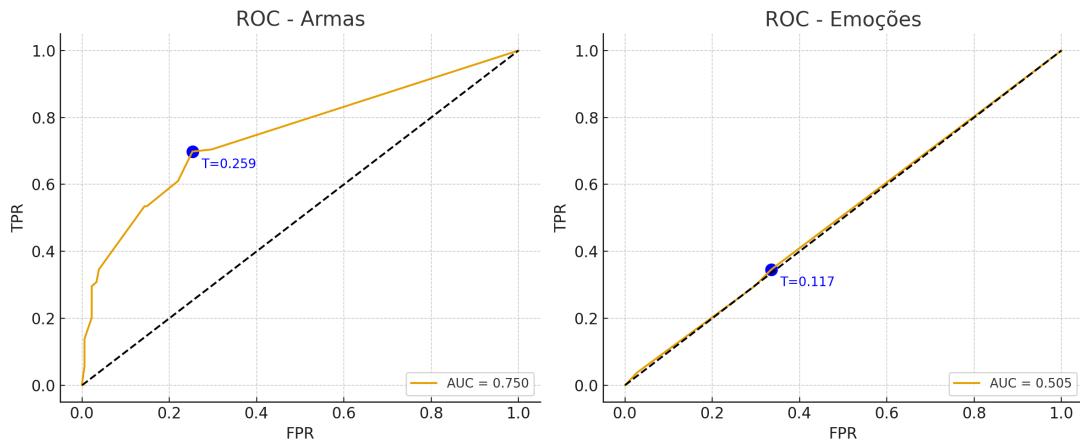


Figura 5.7: Curvas ROC obtidas para as categorias armas e emoções. O ponto azul representa o limiar ótimo segundo o índice de Youden, com o respetivo valor indicado. A linha diagonal corresponde ao desempenho esperado por acaso ( $AUC = 0.5$ ).

### Avaliação do impacto prático

Para avaliar o impacto prático das definições de pesos e limiares por categoria, foi realizado um teste direto ao sistema completo sobre o *dataset* de validação. Cada imagem foi processada pela regra de decisão por *scoring*. Compararam-se dois cenários: (i) *configuração equilibrada*, com pesos uniformes e sem ajustes, e (ii) *configuração ajustada*, com pesos proporcionais ao *recall* dos modelos e limiares por categoria definidos via índice de Youden (Sec. [5.2.4]).

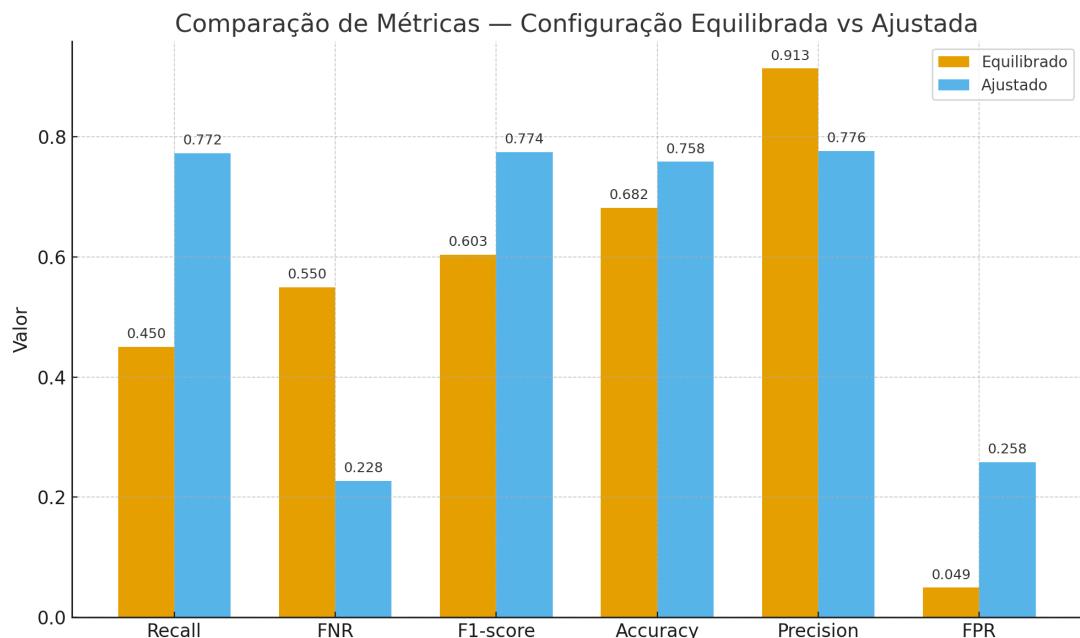


Figura 5.8: Comparaçāo das métricas do sistema antes e após o ajuste dos pesos, com base na execução sobre o *dataset* de validação.

Na Figura 5.8 observa-se que, após os ajustes, o sistema passou a detetar de forma muito mais eficaz as situações de risco: o *recall* aumentou de **0.45** para **0.77**, correspondendo a uma redução da FNR de **0.55** para **0.23**. Em termos práticos, isto significa que o sistema comete menos falhas críticas, deixando passar menos casos de situações alarmantes.

Contudo, este ganho de sensibilidade veio acompanhado de um aumento de alarmes indevidos: a *precision* desceu de **0.91** para **0.78**, e o *FPR* subiu de **0.05** para **0.26**, refletindo um maior número de falsos positivos. Apesar desta perda, o *F1-score* evoluiu positivamente (de **0.60** para **0.77**), evidenciando um melhor equilíbrio global entre precisão e sensibilidade.

Deste modo, os resultados confirmam que o ajuste de pesos adotado esteve alinhado com a prioridade operacional definida: privilegiar a sensibilidade na deteção de situações de risco, ainda que à custa de um maior número de alarmes indevidos. Para além deste objetivo específico, a comparação entre os dois cenários evidencia que a análise e definição de pesos e limiares resultou em melhorias consistentes no desempenho global do sistema. Assim, a abordagem metodológica seguida revelou-se adequada, validando a relevância do processo analítico desenvolvido.

### 5.2.5 Avaliação da métrica de proximidade facial

Com o objetivo de avaliar a viabilidade de utilizar a razão entre áreas faciais como indicador de intrusão, desenvolveu-se uma fase experimental composta pela recolha de dados e pela análise estatística dos resultados. Importa salientar que esta métrica não integra a regra de decisão principal do sistema, assumindo apenas um papel complementar de caráter informativo, destinado a fornecer contexto adicional em cenários de potencial proximidade intrusiva.

#### Recolha de dados e extração da métrica

Foi construído um *dataset* de 80 imagens, distribuídas em duas classes: **negativos**, correspondentes a situações normais (utilizador em primeiro plano, com outras pessoas presentes em segundo plano, mas sem intrusão), e **positivos**, que simulam a presença de uma segunda face em proximidade imediata ao utilizador, representando um potencial agressor.

As imagens resultaram de frames de vídeos encenados, captados diretamente na máquina HEFESTO através da sua câmara integrada, de forma a refletir a resolução e as condições reais de utilização. A Figura 4.2, já apresentada anteriormente, ilustra os dois cenários simulados, enquanto a Figura 5.9 apresenta a distribuição final das 80 imagens pelas duas classes (35 negativas e 45 positivas).

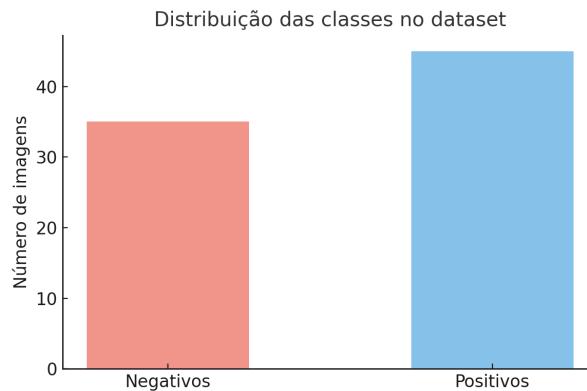


Figura 5.9: Distribuição das imagens do dataset pelas classes negativas e positivas.

Cada frame foi processado através do serviço FER [66], que utiliza o detetor de faces *MTCNN* (*Multi-task Cascaded Convolutional Networks*) para localizar as faces presentes na imagem e, em seguida, aplicar a classificação das expressões emocionais. A partir das caixas delimitadoras devolvidas pelo detetor, calculou-se a área de cada face ( $w \times h$ ), sendo considerada como referência a face de maior dimensão, assumida como a do utilizador. A métrica de interesse correspondeu à razão percentual entre a área da segunda maior face e a área da face principal, conforme descrito na Sec. 4.2.3.

## Análise estatística

As razões calculadas foram analisadas separadamente para os conjuntos positivo e negativo, considerando apenas os casos em que foram detetadas duas faces (excluindo, portanto, os rácios nulos resultantes de imagens com uma única face).

No caso dos negativos, a média situou-se em torno dos 10%, com uma mediana próxima de 9% e um valor máximo de aproximadamente 23%. Já no caso dos positivos, a média atingiu cerca de 48% e a mediana rondou os 47%, registando-se, de forma consistente, valores bastante superiores a 40%.

Estes resultados sugerem uma separação clara entre cenários normais e de intrusão, o que se encontra ilustrado na Figura 5.10, onde se apresenta a distribuição das razões em cada classe sob a forma de *boxplot*, destacando as respetivas medianas.

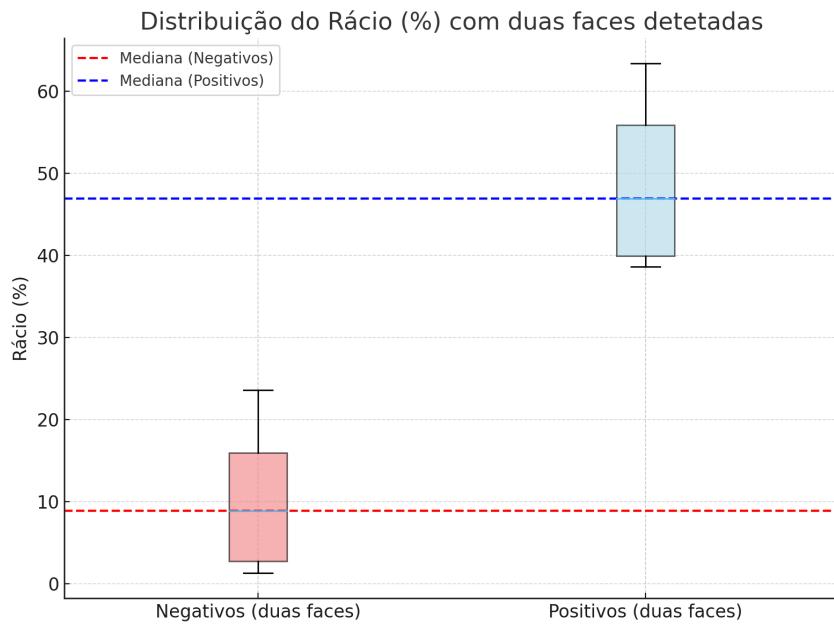


Figura 5.10: Distribuição da razão percentual entre a segunda maior face e a face principal, para as classes negativas e positivas.

### Definição do limiar

Com base nos resultados da Figura 5.10, identificou-se um intervalo de separação natural entre os máximos da classe negativa (23.6%) e os mínimos da classe positiva (40%). Como compromisso entre ambos os extremos, definiu-se o limiar de decisão pelo ponto médio do intervalo:

$$T = \frac{23.6 + 40}{2} \approx 31\% \quad (5.2)$$

Este limiar maximiza a separação entre cenários normais e de intrusão, garantindo uma margem de segurança relativamente às variações observadas em ambos os grupos. Assim, valores de razão inferiores a 30% são interpretados como normais, enquanto valores iguais ou superiores são registados como potenciais intrusões de proximidade.

A escolha foi validada experimentalmente no *dataset*, considerando apenas as imagens em que foram detetadas duas faces. Os resultados obtidos foram: **Accuracy = 0.96**, **Recall = 0.93**, **Precision = 1.00** e **F1-score = 0.97**. Em complemento, a curva ROC apresentou uma área sob a curva (AUC) de **1.00**, evidenciando uma capacidade discriminativa perfeita neste conjunto de dados. Estes resultados demonstram que, no contexto experimental definido, o limiar de 31% é adequado e estatisticamente consistente para separar situações normais de intrusão.

### Discussão

A métrica de proximidade facial assume um caráter meramente informativo, funcionando como um indicador contextual adicional em cenários de possível intrusão, mas sem impacto direto na decisão de alarme.

Importa salientar, contudo, algumas limitações práticas. Em primeiro lugar, o valor da razão entre áreas faciais é sensível ao *Field of View* (FOV) da câmara: alterações no ângulo de captação ou na

distância entre utilizador e dispositivo podem modificar significativamente as dimensões relativas das faces detetadas, influenciando o resultado da métrica. Em segundo lugar, a fiabilidade desta abordagem depende diretamente da qualidade do detetor de faces utilizado (MTCNN através do FER). Falhas na deteção de uma segunda face ou imprecisões na delimitação das caixas podem conduzir a rácios incorretos, originando falsos negativos ou falsos positivos.

## 5.3 Análise de Desempenho e Otimizações

Esta secção apresenta uma análise complementar focada no desempenho do sistema e nas possibilidades de otimização. Foram realizadas três avaliações distintas: comparação entre o uso de NMS na deteção com YOLO, impacto do envio de imagens em lote e medição dos tempos de execução dos modelos e do sistema em ambiente real.

### 5.3.1 Comparação entre YOLO com e sem NMS

Tendo sido referida no trabalho relacionado a proposta de eliminar a etapa de NMS em versões do YOLO, procedeu-se a uma breve análise para avaliar se essa alteração poderia impactar a latência do sistema desenvolvido. Para isso, testou-se um modelo de deteção de armas baseado no YOLOv8 [62], no qual foi possível ativar e desativar o NMS, recolhendo os tempos de execução em 50 imagens do *dataset* de validação.

A Figura 5.11 apresenta os resultados obtidos. Verifica-se que a remoção do NMS traz apenas uma leve redução de tempo, sem relevância prática para o sistema. Assim, manteve-se o uso dos modelos pré-treinados tal como fornecidos, com NMS ativo, sem que isso represente uma limitação relevante.

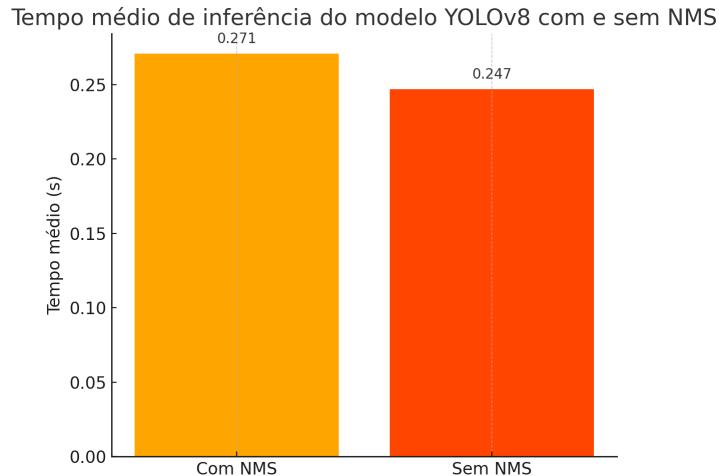


Figura 5.11: Comparação dos tempos de execução do modelo YOLO com e sem NMS, com base em 50 imagens.

### 5.3.2 Avaliação do Processamento em Lote

Foi avaliado o impacto de processar várias imagens em lote no sistema, com o objetivo de reduzir o tempo de processamento por imagem. Para tal, os testes foram realizados em dois ambientes distintos,

ambos limitados a processamento em CPU e utilizando uma câmara configurada para capturar a uma taxa de 15 fps:

- (i) uma máquina equipada com processador Intel® Core™ i7-1255U e 16 GB de RAM;
- (ii) a máquina HEFESTO, com um processador Intel® Celeron® N5095 e 8 GB de RAM.

A escolha de dois ambientes justifica-se pelo facto de nem todas as máquinas de autoatendimento disporem do mesmo hardware que o HEFESTO, e o próprio equipamento está sujeito a evoluções e atualizações de componentes ao longo do tempo. Assim, esta análise comparativa permite compreender o impacto do processamento em lote em diferentes configurações de CPU e antecipar o comportamento do sistema em cenários com hardware diversificado.

A Tabela 5.12 apresenta os tempos de processamento correspondentes ao P95 para lotes de 1, 5 e 10 imagens em ambos os cenários. Os testes foram conduzidos em ambiente integrado, simulando o funcionamento típico do sistema e incluindo a encenação de tentativas de eventos de alarme, durante um período contínuo de 10 minutos.

Tabela 5.12: Comparação dos tempos de processamento por lote em ambos os ambientes de teste.

<b>Tamanho do Lote</b>	<b>Tempo Total P95 (s)</b>		<b>Tempo Médio por Frame (s)</b>	
	<b>Máquina (i)</b>	<b>Máquina (ii)</b>	<b>Máquina (i)</b>	<b>Máquina (ii)</b>
1 imagem	1.14	2.23	1.07	2.11
5 imagens	4.33	10.38	0.86	1.91
10 imagens	8.53	21.91	0.85	2.06

Os resultados obtidos evidenciam que o envio em lote permite uma redução no tempo médio de processamento por imagem. Embora o tempo total aumente proporcionalmente ao tamanho do lote, o processamento conjunto conduz a uma maior eficiência relativa por frame, ou seja, um maior *throughput* global do sistema.

Contudo, importa salientar que o tempo médio por frame apresentado na Tabela 5.12 não corresponde à latência efetiva de cada frame individual. Na prática, todos os frames de um lote só ficam disponíveis após a conclusão do processamento completo desse lote. Por exemplo, no caso da Máquina (i), um lote de 5 imagens demorou cerca de 4,33 s a ser processado, o que equivale a uma média de 0,86 s em termos de *throughput* ( $\approx 1,15$  fps), mas a latência *end-to-end* de cada frame nesse lote é de 4,33 segundos. Assim, valores médios de processamento por imagem inferiores a um segundo traduzem-se em ganhos de taxa de processamento, mas não implicam necessariamente que cada frame isolado seja processado em menos de um segundo. Os ganhos entre 5 e 10 imagens revelaram-se reduzidos, sobretudo quando analisados na Máquina (ii).

Neste ambiente Máquina (ii), o processamento de lotes de 10 imagens provocou uma utilização da CPU próxima de 98% e um consumo de memória superior a 84%, comprometendo a estabilidade do sistema. Por essa razão, este foi o único cenário em que os testes tiveram de ser interrompidos antes dos 10 minutos inicialmente definidos, dada a sobrecarga de recursos verificada. Na mesma máquina (HEFESTO) a configuração com lotes de 5 imagens manteve-se estável e com 10 imagens verificou-se saturação confirmado que tal configuração não é adequada ao hardware em causa.

Face a esta análise, a configuração com lotes de 5 imagens mostrou-se a mais equilibrada, proporcionando ganhos de *throughput* sem comprometer de forma excessiva a latência *end-to-end* nem a capacidade de resposta em ambientes com recursos limitados. Deste modo, a estratégia foi integrada no sistema como uma forma simples e eficaz de otimizar o tempo de inferência, assegurando a sua aplicabilidade em diferentes configurações de hardware, principalmente em máquinas de autoatendimento.

### 5.3.3 Benchmark na Máquina de Autoatendimento

Para além da análise de processamento em lote, foi também realizado um *benchmark* diretamente na máquina de autoatendimento HEFESTO, de forma a aferir a viabilidade da execução local do sistema. Esta avaliação centrou-se na medição do tempo médio de resposta por *frame*, do consumo de memória e do tamanho dos executáveis associados aos serviços dos modelos de visão computacional, ao sistema de segurança e à aplicação nativa da máquina (*kiosk*).

A Tabela 5.13 resume os resultados obtidos, permitindo identificar os componentes com maior impacto nos recursos do sistema.

Tabela 5.13: Benchmarks dos modelos de visão computacional, do sistema de segurança e da aplicação nativa (*kiosk*).

Executável avaliado	Tempo Médio de Resposta (ms)	Tamanho do Executável (MB)	Pico de Memória (MB)
weaponDetectionYoloA.exe	1815.0	243	374.50
weaponDetectionYoloB.exe	1809.0	242	373.29
weaponDetectionYoloC.exe	2148.0	257	462.11
weaponEfficient-net.exe	1658.0	622	348.30
violenceDetection.exe	1794.0	223	346.81
emotion_deepface.exe	367.0	486	348.01
emotion_fer.exe	1260.0	462	459.39
SecurityDetection.exe	2179.0	105	691.90
kiosk.exe	—	—	362.35

A Tabela 5.13 apresenta o desempenho dos diferentes executáveis de visão computacional, do sistema de segurança e da aplicação principal da máquina de autoatendimento (*kiosk.exe*). Os valores da coluna Pico de Memória foram obtidos com lote de 5 imagens conforme definido na Secção 5.3.2. As outras métricas não estão ligadas a esta configuração.

Observa-se que o *SecurityDetection.exe*, correspondente ao sistema de segurança, regista o maior pico de utilização de memória, com aproximadamente 692 MB. Este valor não resulta apenas da execução do sistema, mas também do armazenamento temporário das imagens recebidas para análise, o que contribui significativamente para o aumento do consumo de recursos. Entre os modelos específicos, destacam-se o *weaponDetectionYoloC.exe* e o *emotion\_fer.exe*, ambos com valores próximos de 460 MB. Já o *kiosk.exe*, que representa a aplicação nativa da máquina responsável pelas operações, acrescenta cerca de 362 MB ao consumo total.

Considerando o somatório dos picos de memória de todos os processos listados, o valor global ascende a aproximadamente 4,37 GB. Face à capacidade disponível na máquina de teste (8 GB instalados, dos quais 7,89 GB utilizáveis), esta utilização corresponde a cerca de 55,4% da memória total.

Estes resultados demonstram que a execução simultânea do sistema de segurança em conjunto com os modelos de visão computacional e a aplicação principal é tecnicamente viável no equipamento avaliado (HEFESTO), não comprometendo a operação do sistema nem as funcionalidades essenciais da aplicação cliente.

# Capítulo 6

## Conclusão

### 6.1 Síntese Final

O objetivo central desta dissertação foi o desenvolvimento de um sistema de segurança baseado em visão computacional, direcionado para o reforço da proteção em máquinas de autoatendimento. Este objetivo foi concretizado com a conceção de um sistema que acoplou ao produto HEFESTO como camada adicional de segurança, desenvolvido no âmbito de um estágio curricular e capaz de responder a desafios reais associados à deteção automática de ameaças físicas particularmente relevantes em contextos bancários.

A partir da análise do estado da arte e das vulnerabilidades específicas destas máquinas, identificaram-se as limitações das abordagens tradicionais de segurança. Optou-se pela utilização de redes neurais convolucionais, em particular os modelos YOLO para deteção em tempo real, dada a sua adequação ao problema e a compatibilidade com as restrições de hardware. Evitou-se, por outro lado, o uso de arquiteturas mais pesadas, como os transformadores, privilegiando soluções leves e eficientes, mais ajustadas à infraestrutura disponível.

O sistema desenvolvido agregou deteção de armas, reconhecimento de comportamentos violentos e análise emocional sobre *frames* em tempo real, suportado por uma arquitetura modular *plug-and-play*. Os modelos de inferência pré-treinados foram implementados como serviços externos expostos por uma API REST, o que permitiu ativar, substituir ou atualizar modelos sem alterar a lógica principal do sistema. A integração com as aplicações já existentes nas máquinas de autoatendimento é direta e não exige alterações estruturais: basta que a aplicação capture os *frames* da câmara e os envie, via HTTP, para o sistema de segurança. As decisões são configuráveis, combinando outputs por categoria através de regras de *consensus* ou *scoring*, com limiares e pesos ajustáveis. Além disso, a execução local e a política de não armazenamento garantem a operação independente do sistema, assegurando baixo acoplamento e respeito com os princípios de privacidade.

A validação do sistema foi realizada com recurso ao *dataset* criado no âmbito deste trabalho, que combinou dados abertos, encenados e sintéticos, permitindo concluir que o sistema está apto a operar em condições próximas das reais de utilização. Complementarmente, os testes de desempenho mostraram que a execução em CPU permite assegurar funcionamento contínuo do sistema, embora com diferentes níveis de desempenho consoante a capacidade do *hardware* disponível, tendo-se verificado que o envio de *frames* em lote contribuiu para aumentar a eficiência do processamento por *frame*. Entre os pressupostos e metas definidos no início do trabalho, alguns foram confirmados de forma explícita: o sistema opera localmente de forma estável, sem dependência de rede, e demonstrou ser capaz de manter uma taxa de

processamento próxima de 1 FPS e tempos médios de processamento por frame abaixo de 1 segundo em ambiente de desenvolvimento, assegurando também valores de *recall* superiores aos 0.7 definidos. Já no contexto real da máquina de autoatendimento HEFESTO, verificaram-se limitações de desempenho, com *throughput* inferior a 1 FPS e latências superiores a 1 segundo por frame. Apesar de não atingir o valor inicialmente estabelecido, o sistema manteve-se funcional e estável, validando a arquitetura no ambiente operacional, ainda que com compromissos de desempenho inerentes ao *hardware* limitado disponível.

Para além da análise de desempenho, procedeu-se ao ajuste de parâmetros como pesos e limiares, que otimizou a sensibilidade a eventos críticos sem introduzir um número excessivo de alertas indevidos. Verificou-se ainda que a regra de decisão por *scoring* permite incorporar melhor o conhecimento sobre o desempenho relativo de cada modelo, enquanto a opção por *consensus* mantém utilidade em cenários de menor informação.

Adicionalmente, funcionalidades como a métrica de proximidade facial fornecem informação adicional que não interfere na decisão de alarme, mas emite alertas contextuais sobre a presença de pessoas próximas do utilizador. Esta capacidade contribuiu para enriquecer a percepção da situação e poder apoiar a interpretação em cenários críticos, reforçando a utilidade prática do sistema sem aumentar a complexidade da decisão principal.

No conjunto, os resultados mostram que a integração de visão computacional em máquinas de autoatendimento pode conciliar eficácia na deteção de ameaças com as restrições impostas pelo hardware disponível. O sistema constitui um contributo prático, validado em condições próximas da realidade, e demonstra flexibilidade para evoluir com novas tecnologias e cenários de aplicação. Para além de reforçar os mecanismos tradicionais de segurança, acrescenta uma camada inteligente e adaptável, capaz de apoiar a resposta a situações críticas e de abrir caminho a futuros avanços na interseção entre visão computacional e sistemas de autoatendimento.

## 6.2 Trabalho Futuro

A experiência obtida com o desenvolvimento e a validação do sistema revelou diversas oportunidades de melhoria e de evolução que poderão ser exploradas em trabalhos futuros.

Em primeiro lugar, a atual implementação está focada na execução local. No entanto, a arquitetura do sistema já prevê a possibilidade de um modelo híbrido, no qual os modelos de inferência pré-treinados seriam executados em servidores locais com maior capacidade computacional e comunicariam com o sistema de segurança instalado na máquina de autoatendimento. Esta abordagem permitiria maior flexibilidade e escalabilidade, embora não tenha sido implementada nesta fase.

Outro ponto de evolução prende-se com a integração entre o serviço de emoções e a métrica de proximidade. Atualmente, o sistema considera no máximo duas faces para a decisão de alarme: a principal (utilizador) e a secundária. É ainda calculado o rácio entre as áreas das faces para estimar a proximidade da secundária em relação ao utilizador, embora esta métrica seja usada apenas para gerar avisos, uma evolução natural seria combinar estas duas informações, reportando emoções da face secundária apenas quando o rácio ultrapassasse o limiar definido. Esta abordagem poderia reduzir interferências de pessoas em segundo plano, mas exige uma análise cuidada para não descartar expressões relevantes, como sinais de agressividade de indivíduos mais distantes.

No que respeita à validação, embora neste trabalho tenha sido criado um *dataset* de imagens (reais, simuladas e sintéticas), seria importante recolher vídeos diretamente em máquinas de autoatendimento em funcionamento. Este *dataset* de validação com dados reais e contínuos permitiria aproximar a avaliação das condições efetivas de utilização, analisando a consistência temporal e a capacidade de resposta do sistema em cenários prolongados.

Outra linha de evolução relevante diz respeito à redução dos falsos positivos. Na construção inicial do sistema, a prioridade foi assegurar uma elevada sensibilidade (*recall*), de forma a minimizar o risco de falhar situações de alarme. Para isso, foram aplicadas técnicas de fusão que favoreceram o *recall*, o que resultou num aumento do número de alertas indevidos. Para uma utilização prática em cenários reais, torna-se agora importante reequilibrar esta estratégia, procurando reduzir falsos positivos sem comprometer a capacidade de deteção de eventos críticos.

Adicionalmente, poderá ser explorada a utilização de fusão aprendida através de regressão logística regularizada (*stacking*), capaz de calibrar pesos automaticamente e reduzir redundâncias entre modelos correlacionados. Esta abordagem seria especialmente útil em configurações estáveis de clientes, onde os modelos escolhidos não sofrem alterações frequentes, permitindo criar uma fusão otimizada e especializada para esse cenário. Contudo, em contextos em que a flexibilidade de ativar ou desativar modelos é essencial, tal solução teria limitações, pois exigiria uma nova calibração sempre que a configuração fosse modificada.

Outro aspecto importante é a avaliação da credibilidade dos modelos de visão computacional integrados como serviços no sistema de segurança. A inclusão de um mecanismo interno de validação permitiria aferir não só a qualidade técnica dos modelos adicionados, mas também reduzir o risco de incorporação de componentes potencialmente maliciosos ou manipulados. Desta forma, garantir-se-ia que apenas modelos fiáveis e devidamente certificados seriam utilizados, reforçando a confiança e a robustez do sistema no seu funcionamento contínuo.

Outro caminho relevante é investir no treino de modelos mais especializados, construídos a partir de dados representativos do contexto real das máquinas de autoatendimento. Atualmente, os modelos pré-treinados utilizados foram treinados com *datasets* públicos e genéricos relacionados às suas categorias. A criação de modelos especializados, por exemplo para a deteção de armas, análise emocional ou comportamentos violentos, com base em dados recolhidos em ambientes reais ou simulados, centrados em interações típicas e situações de risco concretas, permitirá construir modelos mais adaptados aos desafios do domínio. Esta abordagem não só tem potencial para aumentar a precisão da deteção, como também para reduzir significativamente a ocorrência de falsos positivos e negativos. A longo prazo, estes modelos especializados poderão substituir alguns dos modelos atualmente utilizados, contribuindo para um sistema de segurança mais fiável e sintonizado com o seu cenário de aplicação. No caso da máquina de autoatendimento HEFESTO, presente em países como Angola, torna-se importante considerar fatores culturais e demográficos no treino dos modelos, de modo a garantir maior precisão e reduzir enviesamentos. Por exemplo, no reconhecimento emocional, os dados devem incluir expressões faciais de pessoas com traços étnicos representativos da população local, enquanto na deteção de armas as imagens de treino devem refletir variações como o tom de pele das mãos que seguram os objetos, assegurando que o sistema responda adequadamente à diversidade dos diversidade do público-alvo.

Por fim, recomenda-se a avaliação do sistema de segurança em máquinas que disponham de GPU. A

comparação do desempenho em cenários com e sem aceleração gráfica permitirá compreender o impacto real dessa diferença e avaliar a viabilidade de implementar o sistema em máquinas de autoatendimento com maior capacidade computacional, ponderando se os ganhos em tempo de resposta justificam o investimento.

Em conjunto, estas propostas representam caminhos viáveis e úteis para a evolução do sistema de segurança, aumentando a sua eficácia, adaptabilidade e robustez perante novos desafios.

# Glossário

**Array NumPy** Estrutura de dados multidimensional da biblioteca NumPy, usada para manipulação eficiente de dados numéricos. 37

**Bytes** Unidade de informação digital composta, em geral, por 8 bits. É utilizada para representar e armazenar dados em sistemas informáticos, como caracteres de texto, instruções de máquina ou elementos multimédia. No plural, *bytes* refere-se a múltiplas unidades desta medida. 43

**FastAPI** Framework moderna em Python para criação de APIs rápidas e seguras, baseada nos princípios RESTful. É otimizada para alto desempenho utilizando *Starlette* e *Pydantic*, com suporte a tipagem estática, documentação automática (OpenAPI/Swagger) e integração fácil com aplicações de computação visual, inteligência artificial e micro-serviços. 31

**FER-2013** Conjunto de dados (*dataset*) para reconhecimento de expressões faciais, criado em 2013. É amplamente utilizado em tarefas de treino e avaliação de modelos de FER (*Facial Emotion Recognition*). 50

**Inferência** Processo de executar um modelo previamente treinado para obter previsões sobre novos dados. 14

**OpenCV** Biblioteca de código aberto de visão computacional (*Open Source Computer Vision Library*). Fornece ferramentas para processamento de imagem e vídeo, deteção e descrição de características, calibração de câmera, rastreamento, e integrações com aprendizagem automática. Suporta C/C++, Python e outras linguagens. 29

**RAF-DB** *Real-world Affective Faces Database*. Conjunto de dados de faces em contexto real, anotadas com diferentes expressões emocionais. É amplamente utilizado em tarefas de reconhecimento de emoções faciais. 50

**Random Forest** Consiste num conjunto de árvores de decisão treinadas sobre diferentes subconjuntos dos dados, cujas previsões são combinadas de forma agregada.. 63

**RESTful** Estilo arquitetural para serviços Web baseado nos princípios do (*Representational State Transfer*). Um sistema RESTful segue estas restrições de forma consistente, usando operações HTTP para criar, ler, atualizar e remover recursos identificados por URLs. 31

**RWF-2000** *Real-world Fighting Dataset 2000.* Conjunto de dados composto por 2000 vídeos recolhidos de ambientes reais, utilizados para treinar e avaliar modelos de deteção de comportamentos violentos. 51

**Tensor** Estrutura de dados utilizada em aprendizagem profunda, equivalente a um array multidimensional, fundamental em frameworks como PyTorch e TensorFlow. 35

**VGG-19** Arquitetura de rede neuronal convolucional profunda com 19 camadas, desenvolvida pelo *Visual Geometry Group* da Universidade de Oxford. 27

# Bibliografia

- [1] “Innovation makers,” 2009. Accessed Nov. 2024.
- [2] J. Económico, “Innovation makers lança máquinas de autoatendimento hefesto para o setor bancário,” 2024.
- [3] D. Meimetus, I. Daramouskas, I. Perikos, and I. Hatzilygeroudis, “Real-time multiple object tracking using deep learning methods,” *Neural Computing and Applications*, vol. 35, no. 1, pp. 89–118, 2023.
- [4] C. Chen, R. Surette, and M. Shah, “Automated monitoring for security camera networks: promise from computer vision labs,” *Security Journal*, vol. 34, no. 3, pp. 389–409, 2021.
- [5] K. B. Kwan-Loo, J. C. Ortíz-Bayliss, S. E. Conant-Pablos, H. Terashima-Marín, and P. Rad, “Detection of violent behavior using neural networks and pose estimation,” *IEEE Access*, vol. 10, pp. 86339–86352, 2022.
- [6] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding, “Yolov10: Real-time end-to-end object detection,” *arXiv preprint arXiv:2405.14458*, 2024.
- [7] Y.-H. Wang, J.-W. Hsieh, P.-Y. Chen, M.-C. Chang, H.-H. So, and X. Li, “Smiletrack: Similarity learning for occlusion-aware multiple object tracking,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 5740–5748, 2024.
- [8] B. Sujith, “Crime detection and avoidance in atm: a new framework,” *International Journal of Computer Science and Information Technologies*, vol. 5, no. 5, pp. 6068–6071, 2014.
- [9] S. Li and W. Deng, “Deep facial expression recognition: A survey,” *IEEE transactions on affective computing*, vol. 13, no. 3, pp. 1195–1215, 2020.
- [10] M. Cheng, K. Cai, and M. Li, “Rwf-2000: An open large scale video database for violence detection,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 4183–4190, 2021.
- [11] European Association for Secure Transactions (EAST), “Atm physical attacks rise in europe,” 2024. Acessado em março de 2025.
- [12] V. Singh, S. Singh, and P. Gupta, “Real-time anomaly recognition through cctv using neural networks,” *Procedia Computer Science*, vol. 173, pp. 254–263, 2020.

- [13] H. Iberahim, N. M. Taufik, A. M. Adzmir, and H. Saharuddin, “Customer satisfaction on reliability and responsiveness of self service technology for retail banking services,” *Procedia Economics and Finance*, vol. 37, pp. 13–20, 2016.
- [14] N. F. A. Roslan, M. H. Mifzal, R. Kassim, and W. N. W. Azman, “The implementation of self-service kiosk at larkin bus station,” 2024.
- [15] J. Tomé, “Tomi lança serviço inovador para tirar filas das lojas do cidadão,” <https://insider.dn.pt/noticias/tomi-lanca-servico-inovador-tirar-filas-das-lojas-do-cidadao/612/>, 2018. Accessed Out. 2024.
- [16] B. Sujith, “Crime detection and avoidance in atm: a new framework,” *International Journal of Computer Science and Information Technologies*, vol. 5, no. 5, pp. 6068–6071, 2014.
- [17] K. Bessa, “Assaltos nos atm preocupa untentes em cacuaco,” <https://namiradocrime.info/show/7567>, 2023. Accessed Out. 2024.
- [18] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.
- [19] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, “A survey of convolutional neural networks: Analysis, applications, and prospects,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022.
- [20] C. G. García, D. Meana-Llorián, B. C. P. G-Bustelo, J. M. C. Lovelle, and N. Garcia-Fernandez, “Midgar: Detection of people through computer vision in the internet of things scenarios to improve the security in smart cities, smart towns, and smart homes,” *Future Generation Computer Systems*, vol. 76, pp. 301–313, 2017.
- [21] U. de Stanford, “Cs329s: Machine learning systems design.” <https://stanford-cs329s.github.io/syllabus.html>, 2022. Acesso em: nov. 2024.
- [22] D. Weichert, P. Link, A. Stoll, S. Rüping, S. Ihlenfeldt, and S. Wrobel, “A review of machine learning for the optimization of production processes,” *The International Journal of Advanced Manufacturing Technology*, vol. 104, no. 5, pp. 1889–1902, 2019.
- [23] R. Agarwal, “Implementing batch and real-time ml systems for scalable user engagement,” *International Journal of Realtime Systems and Machine Learning*, 2025.
- [24] Q. Qi and F. Tao, “A smart manufacturing service system based on edge computing, fog computing, and cloud computing,” *IEEE Access*, 2019.
- [25] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, “Convergence of edge computing and deep learning: A comprehensive survey,” *IEEE Communications Surveys Tutorials*, 2020.
- [26] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, “Multiple object tracking: A literature review,” *Artificial intelligence*, vol. 293, p. 103448, 2021.

- [27] B. Mirzaei, H. Nezamabadi-Pour, A. Raoof, and R. Derakhshani, “Small object detection and tracking: a comprehensive review,” *Sensors*, vol. 23, no. 15, p. 6887, 2023.
- [28] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.
- [29] A. Wang, “Yolov10,” <https://github.com/THU-MIG/yolov10>, 2023.
- [30] S. Oh, Y. Kwon, and J. Lee, “Optimizing real-time object detection in a multi-neural processing unit system,” *Sensors*, vol. 25, no. 5, p. 1376, 2025.
- [31] B. Yogameena and C. Nagananthini, “Computer vision based crowd disaster avoidance system: A survey,” *International journal of disaster risk reduction*, vol. 22, pp. 95–129, 2017.
- [32] T. Ko, “A survey on behavior analysis in video surveillance for homeland security applications,” in *2008 37th IEEE Applied Imagery Pattern Recognition Workshop*, pp. 1–8, 2008.
- [33] L. Wang, Y. Qiao, and X. Tang, “Video action detection with relational dynamic-poselets,” in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*, pp. 565–580, Springer, 2014.
- [34] H. Idrees, M. Shah, and R. Surette, “Enhancing camera surveillance using computer vision: a research note,” *Policing: An International Journal*, vol. 41, no. 2, pp. 292–307, 2018.
- [35] W. Sultani, C. Chen, and M. Shah, “Real-world anomaly detection in surveillance videos,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6479–6488, 2018.
- [36] M. E. Kundegorski, S. Akçay, M. Devereux, A. Mouton, and T. P. Breckon, “On using feature descriptors as visual words for object detection within x-ray baggage security screening,” 2016.
- [37] S. Akcay, M. E. Kundegorski, C. G. Willcocks, and T. P. Breckon, “Using deep convolutional neural network architectures for object classification and detection within x-ray baggage security imagery,” *IEEE transactions on information forensics and security*, vol. 13, no. 9, pp. 2203–2215, 2018.
- [38] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, vol. 28, 2015.
- [39] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, “A survey of modern deep learning based object detection models,” *Digital Signal Processing*, vol. 126, p. 103514, 2022.
- [40] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*, pp. 21–37, Springer, 2016.
- [41] J. Glenn, “Yolov8,” <https://github.com/ultralytics/ultralytics/tree/main>, 2023.
- [42] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

- [43] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, “Searching for mobilenetv3,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.
- [44] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.
- [45] J. Redmon and A. Farhadi, “Yolo9000: better, faster, stronger,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.
- [46] Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, and J. Chen, “Detrs beat yolos on real-time object detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16965–16974, 2024.
- [47] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*, pp. 740–755, Springer, 2014.
- [48] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, “Fairmot: On the fairness of detection and re-identification in multiple object tracking,” *International journal of computer vision*, vol. 129, pp. 3069–3087, 2021.
- [49] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in *2016 IEEE international conference on image processing (ICIP)*, pp. 3464–3468, IEEE, 2016.
- [50] M.-H. Guo, T.-X. Xu, J.-J. Liu, Z.-N. Liu, P.-T. Jiang, T.-J. Mu, S.-H. Zhang, R. R. Martin, M.-M. Cheng, and S.-M. Hu, “Attention mechanisms in computer vision: A survey,” *Computational visual media*, vol. 8, no. 3, pp. 331–368, 2022.
- [51] T. Meinhhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, “Trackformer: Multi-object tracking with transformers,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8844–8854, 2022.
- [52] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, “Bytetrack: Multi-object tracking by associating every detection box,” in *European conference on computer vision*, pp. 1–21, Springer, 2022.
- [53] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, “Bot-sort: Robust associations multi-pedestrian tracking,” *arXiv preprint arXiv:2206.14651*, 2022.
- [54] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*, vol. 1, pp. 886–893, Ieee, 2005.
- [55] M. Song, D. Tao, Z. Liu, X. Li, and M. Zhou, “Image ratio features for facial expression recognition application,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 3, pp. 779–788, 2009.

- [56] S. Vignesh, M. Savithadevi, M. Sridevi, and R. Sridhar, “A novel facial emotion recognition model using segmentation vgg-19 architecture,” *International Journal of Information Technology*, vol. 15, no. 4, pp. 1777–1787, 2023.
- [57] W. S. Noble, “What is a support vector machine?,” *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006.
- [58] G. Bradski, “Learning opencv: Computer vision with the opencv library,” *O'REILLY google schola*, vol. 2, pp. 334–352, 2008.
- [59] S. I. Serengil, “Deepface: A lightweight face recognition and facial attribute analysis framework.” <https://github.com/serengil/deepface>, 2024. Acesso em: 13 de novembro de 2024.
- [60] G. Van Rossum, F. L. Drake, *et al.*, *Python reference manual*, vol. 111. Centrum voor Wiskunde en Informatica Amsterdam, 1995.
- [61] S. Ramírez, “Fastapi - the modern web framework for python,” *FastAPI Documentation*, 2024. Acessado em: 7 de dezembro de 2024.
- [62] J. Assalim, “Weapons and knives detector with yolov8.” <https://github.com/JoaoAssalim/Weapons-and-Knives-Detector-with-YOLOv8>, 2023. Acedido em 2025-04-21.
- [63] BecayeSoft, “Guns detection yolov8.” <https://github.com/BecayeSoft/Guns-Detection-YOLOv8>, 2023. Acedido em 2025-04-21.
- [64] GingerBrains, “Object detection with yolov8.” <https://github.com/GingerBrains/object-detection>, 2023. Acedido em 2025-06-21.
- [65] JacobM184, “Efficientnet for gun detection.” <https://github.com/JacobM184/EfficientNet-for-Gun-detection>, 2023. Acedido em 2025-04-21.
- [66] J. Shenk, “fer: Facial expression recognition.” <https://github.com/JustinShenk/fer>, 2023. Acesso em: 2025-04-19.
- [67] Musawer14, “fight\_detection\_yolov8.” [https://huggingface.co/Musawer14/fight\\_detection\\_yolov8](https://huggingface.co/Musawer14/fight_detection_yolov8), 2024. Acedido em 2025-04-21.
- [68] M. Ala'raj and M. F. Abbod, “Classifiers consensus system approach for credit scoring,” *Knowledge-Based Systems*, vol. 104, pp. 89–105, 2016.
- [69] H. G. Jabbar, “Advanced threat detection using soft and hard voting techniques in ensemble learning,” *Journal of Robotics and Control (JRC)*, vol. 5, no. 4, pp. 1104–1116, 2024.
- [70] M. T. Bhatti, M. G. Khan, M. Aslam, and M. J. Fiaz, “Weapon detection in real-time cctv videos using deep learning,” *Ieee Access*, vol. 9, pp. 34366–34382, 2021.

- [71] M. T. Abdullah and J. H. ALameri, “A multi-weapon detection using synthetic dataset and yolov5,” in *2022 Fifth College of Science International Conference of Recent Trends in Information Technology (CSCTIT)*, pp. 100–104, IEEE, 2022.
- [72] Roboflow, “Roboflow: Organize, annotate, and train computer vision datasets.” <https://roboflow.com>, 2024. Acessado em fevereiro de 2025.
- [73] weapon detection cctv, “Weapon detection cctv v3 dataset.” <https://universe.roboflow.com/weapon-detection-cctv/weapon-detection-cctv-v3-dataset>, 2023.
- [74] Roboflow Community, “Yolo weapon detection dataset.” <https://universe.roboflow.com/weapon-detect-qbsiw/yolo-weapon-detection>. Acedido em julho de 2025.
- [75] Roboflow Community, “Suspicious detection dataset.” <https://universe.roboflow.com/suspicious-movement/suspicious-detection>.
- [76] U. Roboflow, “Silah computer vision project.” <https://universe.roboflow.com/deneme-jnb8a/silah-h8axi>, 2023.
- [77] U. Roboflow, “Knife detection computer vision project.” <https://universe.roboflow.com/ai-0jtbr/knife-detection-hgvy2>, 2023.
- [78] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, *et al.*, “Challenges in representation learning: A report on three machine learning contests,” in *Neural information processing: 20th international conference, ICONIP 2013, daegu, korea, november 3-7, 2013. Proceedings, Part III* 20, pp. 117–124, Springer, 2013.
- [79] S. Li, W. Deng, and J. Du, “Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2852–2861, 2017.
- [80] H. Akoglu, “User’s guide to correlation coefficients,” *Turkish journal of emergency medicine*, vol. 18, no. 3, pp. 91–93, 2018.