# Bridging the Gap in Facial Expression Recognition: Real-Time Use and Human-in-the-Loop Data Collection

**Paulo Ricardo**[1], **Tiago de Melo**[1], **Elloá B. Guedes**[1]

[1]Universidade do Estado do Amazonas, Manaus, 69050-020, Brasil

{prfds.eng21,tmelo,ebgcosta}@uea.edu.br

***Abstract.*** *Traditional evaluation of facial expression recognition (FER) solutions fails to reflect the challenges of their deployment in real-world scenarios. To mitigate this limitation, this paper proposes a real-time, web-based FER system (YOLOv8n, WebRTC, FastAPI) integrated with a human-in-the-loop mechanism that converts classification errors into YOLO-formatted labeled samples, compiling hard examples for future retraining. Empirical evaluation confirmed the gap between benchmark assessments of an FER solution and practical use scenarios: the throughput decreased from 56 FPS to 25 FPS (GPU) and 7 FPS (CPU) within the complete pipeline, while also revealing sensitivity to pose and capture quality. We conclude that this approach presents a practical pathway for continuous data collection, supporting the future enhancement of the model.*

***Resumo.*** *A avaliação tradicional de soluções de reconhecimento de expressões faciais (REF) não reflete os desafios de sua utilização em situações reais. Para mitigar essa limitação, este artigo propõe um sistema web de REF em tempo real (YOLOv8n, WebRTC, FastAPI) integrado a um mecanismo* human-in-the-loop *que converte os erros de classificação em amostras rotuladas no formato YOLO, compilando exemplos difíceis para futuros re-treinos. A avaliação empírica confirmou a lacuna entre avaliação de uma solução REF em* benchmark *em contraste com um cenário de uso prático, em que a vazão de 56 FPS reduziu para 25 FPS (GPU) e 7 FPS (CPU) no fluxo completo, revelando também sensibilidade à pose e à qualidade da captura. O sistema proposto viabiliza a coleta contínua de dados e apoia o aprimoramento futuro do modelo.*

## 1. Introdução

O reconhecimento de expressões faciais (REF) é uma área central em Visão Computacional e Computação Afetiva, com aplicações que compreendem a interação humano-computador, o uso em sistemas assistivos, o auxílio na segurança e o monitoramento da atenção e bem-estar, dentre outras [Alshammari and Alshammari 2024, Taware and Thakare 2023]. A possibilidade de sistemas computacionais interpretarem sinais emocionais humanos tem motivado o desenvolvimento de sistemas interativos mais sensíveis ao contexto, especialmente quando o reconhecimento ocorre em tempo real, em vídeo capturado em condições reais de uso.

Apesar dos avanços recentes, há uma lacuna pouco explorada entre o desempenho medido em *benchmarks* de FER versus o uso dessas soluções em sistemas de tempo real. Modelos inteligentes, especialmente aqueles baseados em Aprendizado Profundo, são tipicamente avaliados segundo métricas calculadas sobre um conjunto de teste fixo, como *mean Average Precision* (mAP), que fornece uma estimativa controlada de acurácia da

detecção da face humana e da classificação da expressão facial correspondente, mas que não captura fatores que só se manifestam perante o uso contínuo: variações de iluminação, pose de cabeça, oclusões parciais, qualidade da câmera e a própria subjetividade na reprodução de expressões [Liao et al. 2023, Mollahosseini et al. 2017, Lu et al. 2025]. Em cenários não controlados (*in-the-wild*), a capacidade de aferição da métrica de desempenho elencada no conjunto de teste pode ficar aquém da experiência real de uso, a qual é definitivamente relevante e raramente é quantificada.

Entre as arquiteturas de detecção baseadas em Aprendizado Profundo capazes de operar em tempo real, a família *You Only Look Once* (YOLO) [Redmon et al. 2016] consolidou-se como referência por realizar detecção e classificação em uma única passagem (*single stage*) pela rede, unificando a detecção facial e a classificação da expressão com um baixo custo computacional. No entanto, dispor de um modelo rápido e acurado em teste não é suficiente. Para evoluir um sistema de REF, é preciso observar como o modelo se comporta perante o uso em condições realísticas e, sobretudo, capturar sistematicamente seus erros. Esse último ponto esbarra em um obstáculo prático bem conhecido, que consiste na intensa exigência de recursos humanos, financeiros e de tempo para rotulação manual de volumes de dados cada vez mais crescentes [Wu 2024]. Consequentemente, surge a necessidade de desenvolver sistemas que consigam contribuir para a construção de novas rotulações para dados, auxiliando na construção de novos conjuntos de dados em tempo de execução.

Diante disso, este trabalho apresenta um sistema web de detecção de expressões faciais em tempo real que realiza inferências a partir do monitoramento da entrada de uma *webcam*, que atua como um sensor de captura de vídeo. Além disso, o sistema incorpora um mecanismo de *feedback* com humano no laço (*human-in-the-loop*), por meio do qual o usuário confirma ou corrige as predições durante o uso. Cada predição marcada como incorreta é convertida automaticamente em um exemplo rotulado no formato YOLO, como uma imagem, rótulo com a classe corrigida, coordenadas normalizadas e metadados de rastreabilidade, formando um conjunto de candidatos a exemplos difíceis (*hard examples*) para re-treino futuro. Um painel analítico (*dashboard*) consolida os registros e permite acompanhar o comportamento do modelo em uso interativo. O objetivo principal deste trabalho é desenvolver e avaliar funcionalmente esse sistema web de REF em tempo real, com ênfase no mecanismo human-in-the-loop para coleta de exemplos difíceis. Como análise complementar, investiga-se a diferença entre o desempenho isolado do modelo e o comportamento observado quando ele é integrado ao pipeline completo da aplicação.

As principais contribuições deste trabalho são: ($i$) um sistema web de REF em tempo real com arquitetura cliente-servidor baseada em WebRTC e WebSocket, integrando um modelo da família YOLO ao fluxo contínuo de captura, inferência e exibição; ($ii$) um *workflow* baseado no paradigma *human-in-the-loop* que transforma erros do modelo em exemplos rotulados no formato YOLO, apoiando a coleta de amostras corrigidas em uso real para curadoria e aprimoramento futuro; e ($iii$) evidência empírica da lacuna entre *benchmark* e uso real, considerando o desempenho em GPU e CPU, o efeito da qualidade da câmera e da pose, e a análise dos erros de detecção registrados durante o uso.

Para apresentar o que se propõe, o artigo está organizado da seguinte forma: a Seção 2 discute trabalhos relacionados; a Seção 3 apresenta o sistema proposto e a seleção do modelo; a Seção 4 relata a avaliação e os resultados; e a Seção 5 conclui o trabalho e

aponta direções futuras.

## 2. Trabalhos Relacionados

Esta seção apresenta os trabalhos relacionados que embasam a pesquisa. Inicialmente, são elencados os desafios do REF em cenários não controlados e a aplicação de arquiteturas da família YOLO para inferência em tempo real. Em seguida, aborda-se a infraestrutura de sistemas web necessária para a análise contínua em vídeos. A partir dessa trajetória, o texto culmina na identificação das lacunas na literatura.

### 2.1. Reconhecimento de Expressões Faciais em Cenários Não Controlados

Os primeiros métodos de REF empregavam a extração manual de características, como *Local Binary Patterns* e *Histogram of Oriented Gradients*, seguidas de classificadores tradicionais [Kumari et al. 2015]. Embora eficazes em ambientes controlados, o desempenho desses métodos degradava-se mediante variações de iluminação, pose e oclusões. A transição para o Aprendizado Profundo permitiu aprender representações discriminativas diretamente dos dados, superando boa parte dessas limitações [Liao et al. 2023]. Ainda assim, a robustez em cenários reais permanece dependente da diversidade e da escala das bases de dados; bases amplas e realistas, como a AffectNet [Mollahosseini et al. 2017], foram decisivas para treinar modelos mais generalizáveis. Mesmo com esses avanços, o REF *in-the-wild* continua desafiador, com limitações de desempenho associadas a variações de iluminação, ângulo da cabeça, oclusões e diferenças individuais na manifestação emocional [Liao et al. 2023, Lu et al. 2025, Shao and Qian 2019].

### 2.2. Arquiteturas YOLO Aplicadas ao REF

A família YOLO vem sendo crescentemente explorada para REF por unir detecção e classificação em tempo real. Trabalhos recentes aplicam o YOLOv8 à detecção de expressões emocionais [Alshammari and Alshammari 2024], combinam YOLOv5 com mecanismos de atenção para reconhecimento em ambientes educacionais [Zhong et al. 2023], propõem adaptações do YOLOv8 com ganhos de precisão [Li et al. 2024] e integram YOLO a *Vision Transformers* para REF baseado em vídeo [Sareen and Seeja 2025]. Revisões abrangentes [Jegham et al. 2025] descrevem a evolução das versões recentes, evidenciando melhorias arquiteturais que mantêm desempenho adequado para aplicações em tempo real. A maioria desses estudos, contudo, concentra-se na avaliação de métricas sobre conjuntos de teste, sem investigar o comportamento do modelo em uso interativo contínuo.

### 2.3. Sistemas Web para Análise de Vídeo em Tempo Real

A operação de modelos de visão em tempo real sobre o navegador demanda a coordenação de captura, processamento, inferência e visualização. Arquiteturas distribuídas e orientadas a borda para análise de fluxos de vídeo organizam esse processamento em *pipelines* e disponibilizam interfaces baseadas em HTTP e WebRTC para consumo dos dados [Sassu et al. 2021]. *Frameworks* de serviço em borda demonstram a viabilidade de análise inteligente de vídeo com baixa latência próxima à fonte de captura [Liu et al. 2018]. O WebRTC, em particular, foi projetado para comunicação em tempo real em navegadores e dispositivos móveis, oferecendo baixa latência e suporte à transmissão contínua de mídia [Mahmoud and Abozariba 2025], sendo uma alternativa relevante para sistemas interativos baseados em vídeo.

### 2.4. Lacuna Abordada

Embora a literatura sugira arquiteturas YOLO eficazes para REF e infraestruturas robustas para vídeo em tempo real, são escassos os trabalhos que $(i)$ integrem essas duas frentes em um sistema funcional; $(ii)$ caracterizem empiricamente a distância entre o desempenho de *benchmark* e o uso real; e $(iii)$ aproveitem o próprio uso do sistema para coletar e rotular, com o apoio do usuário, os casos em que o modelo falha. Essas lacunas são o que o presente trabalho busca mitigar.

## 3. Solução Proposta

A solução proposta no escopo deste trabalho consiste em um sistema estruturado em uma arquitetura cliente-servidor, composta por um módulo de *frontend*, um módulo de *backend*, um modelo de inferência YOLOv8n e uma camada de persistência local. Essa organização separa, no cliente, a interação com o usuário e, no servidor, as rotinas de processamento associadas ao modelo. A Figura 1 apresenta o fluxo geral, desde a captura pela *webcam* até a consolidação dos *feedbacks* no *dashboard*.

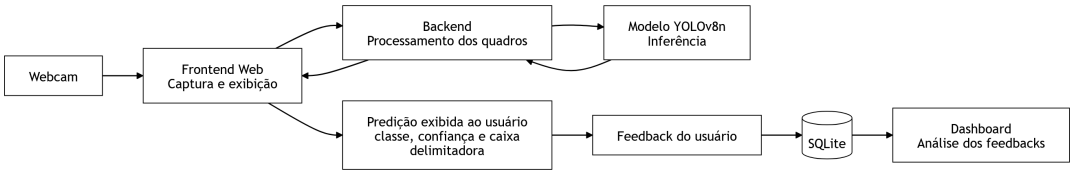



**Figura 1. Arquitetura geral do sistema proposto para reconhecimento de expressões faciais em tempo real, destacando o fluxo entre câmera, *frontend*, *backend*, modelo YOLOv8n e *dashboard* de análise.**

### 3.1. Seleção e Integração do Modelo

O modelo adotado neste trabalho foi definido a partir de um estudo comparativo prévio que avaliou quatro versões recentes da família YOLO, a citar: YOLOv8, YOLOv10, YOLOv11 e YOLOv12, todas na variante *nano* em razão do menor número de parâmetros, sob um protocolo experimental unificado, utilizando o conjunto público "*9 Facial Expressions for YOLO*" em condições não controladas [Ricardo and de Melo 2026]. A Tabela 1 reproduz os resultados do estudo em questão, que serviram como critério de seleção do modelo no escopo deste trabalho.

**Tabela 1. Resumo do comparativo entre versões YOLO (variante *nano*, mesmo protocolo experimental) perante *dataset* FER. Fonte: [Ricardo and de Melo 2026].**

| Modelo | Parâmetros | GFLOPs | mAP@50 | mAP@50–95 | FPS | Treino |
|---|---|---|---|---|---|---|
| YOLOv8n | ∼3,01M | 8,9 | 0,837 | 0,681 | 56,17 | ∼8,99 h |
| YOLOv10n | ∼2,78M | 8,7 | 0,827 | 0,668 | 73,85 | ∼10,70 h |
| YOLOv11n | ∼2,64M | 6,6 | 0,825 | 0,673 | 62,42 | ∼8,79 h |
| YOLOv12n | ∼2,60M | 6,7 | 0,839 | 0,686 | 47,11 | ∼14,63 h |

As diferenças preditivas entre as versões avaliadas foram pequenas. Embora o YOLOv12n tenha obtido o maior mAP@50–95 e o YOLOv10n tenha apresentado a

maior taxa de FPS no teste isolado, o YOLOv8n foi adotado por apresentar um equilíbrio adequado entre acurácia, velocidade, estabilidade de implementação e viabilidade de integração ao protótipo. Além disso, sua variante *nano* possui baixo custo computacional, uma característica relevante para aplicações em tempo real e para cenários com restrições de hardware, como dispositivos de computação de borda.

A escolha do YOLOv8n também se justifica pela maturidade do ecossistema Ultralytics, pela disponibilidade de ferramentas consolidadas para treinamento, inferência e exportação de modelos, e pela compatibilidade direta com o formato de anotações YOLO utilizado no conjunto de dados e no mecanismo de coleta proposto. Essa compatibilidade simplifica tanto a integração inicial quanto o reaproveitamento futuro das amostras corrigidas pelo usuário.

A arquitetura do sistema, contudo, não é restrita ao YOLOv8n. O módulo de inferência foi estruturado de forma modular, podendo ser substituído por outros detectores que retornem caixas delimitadoras, classes e níveis de confiança em formato compatível com o fluxo da aplicação. Essa flexibilidade permite avaliar futuramente modelos mais recentes da família YOLO ou arquiteturas baseadas em *Vision Transformers*, desde que atendam aos requisitos de latência, custo computacional e integração exigidos por aplicações em tempo real.

### 3.2. Inferência em Tempo Real

O *frontend* captura o vídeo da *webcam* diretamente no navegador, exibe a cena analisada e exibe as detecções sobre a imagem por meio de um *canvas* sobreposto à área de vídeo, apresentando as caixas delimitadoras, rótulos de classe e nível de confiança. A interface foi desenvolvida com React, TypeScript e Vite, com estilização em Tailwind CSS; os gráficos do *dashboard* utilizam a biblioteca Recharts.

O *backend*, implementado com FastAPI, recebe os quadros, pré-processa as imagens e executa a inferência. A transmissão do fluxo de vídeo do cliente para o servidor ocorre por WebRTC, enquanto o retorno das detecções para a interface utiliza WebSocket; essa combinação separa o envio contínuo dos quadros da transmissão dos resultados, favorecendo a comunicação em tempo real [Mahmoud and Abozariba 2025, Sassu et al. 2021]. O processamento dos quadros apoia-se nas bibliotecas OpenCV e NumPy, e a execução do modelo é feita pela biblioteca Ultralytics YOLO [Jocher et al. 2023], responsável por carregar os pesos pré-treinados do modelo e também pela realização das inferências, tendo o PyTorch como base de execução. Para cada detecção, o sistema retorna a classe prevista, a confiança e as coordenadas da caixa delimitadora, que são ajustadas à escala de exibição e enviadas ao *frontend*. A interface, então, atualiza continuamente o vídeo com as detecções e informações complementares, como confiança, taxa de quadros por segundo, latência estimada e dispositivo de execução.

### 3.3. Mecanismo de *Feedback* com Humano no Laço

Além da inferência, o sistema incorpora um mecanismo de *feedback* manual que permite ao usuário avaliar as predições durante o uso. Quando uma detecção é exibida, o usuário pode congelar o quadro e, em uma janela de confirmação, indicar se a predição está correta, incorreta ou se não sabe avaliá-la, vide Figura 2. Quando a predição é marcada como incorreta, a interface apresenta uma segunda etapa, na qual o usuário seleciona a classe correta dentre as nove categorias disponíveis: (*angry*, *contempt*, *disgust*, *fear*, *happy*, *natural*, *sad*, *sleepy* e *surprised*). O sistema também permite registrar situações em que o

modelo não produz nenhuma detecção útil, por meio da opção “não detectou”, tratada separadamente dos casos de incerteza.

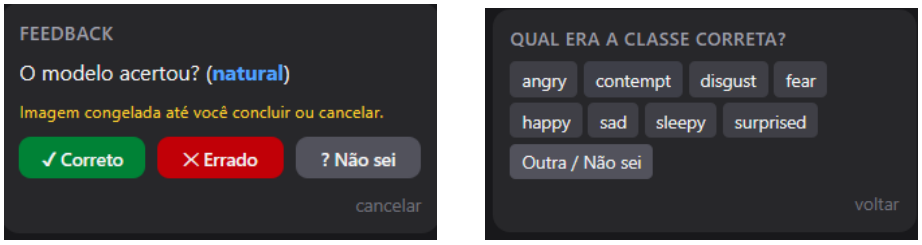



**Figura 2. Etapas do mecanismo de *feedback*: à esquerda, confirmação da predição realizada pelo modelo; à direita, seleção da classe correta quando a predição é marcada como incorreta.**

Esse mecanismo torna o usuário parte do laço de avaliação, registrando não apenas confirmações de acerto, mas também correções de classe e falhas de detecção. Essas informações são as que alimentam tanto a análise do *dashboard* quanto a coleta automática de dados descrita a seguir.

### 3.4. Geração de Dados Anotados com Apoio do Usuário

Um diferencial do sistema proposto consiste em transformar os erros do modelo em novos dados de treinamento. Quando o usuário marca uma predição como incorreta e informa a classe correta, o *backend* salva o quadro correspondente juntamente com um arquivo de rótulo no formato YOLO, contendo o identificador da classe corrigida e as coordenadas normalizadas (centro, largura e altura) da caixa delimitadora. Esse formato segue exatamente a estrutura utilizada no treinamento dos modelos, de modo que as amostras coletadas possam ser reaproveitadas diretamente para fins de re-treino ou de ajuste fino.

Adicionalmente, o sistema registra metadados em um arquivo `metadata.jsonl`, incluindo o identificador da amostra, os caminhos da imagem e do rótulo, a classe prevista, a classe corrigida, a confiança da predição, as dimensões do quadro, as coordenadas da caixa delimitadora e os indicadores de validação e de elegibilidade da amostra para reaproveitamento. Esses metadados permitem rastrear a origem de cada exemplo e filtrar amostras inadequadas, como imagens com baixa nitidez, problemas de iluminação ou possíveis duplicatas.

É importante, entretanto, delimitar o escopo desse mecanismo. Como o rótulo YOLO exige uma caixa delimitadora, somente os erros de classificação em que há uma detecção com caixa, mas a classe prevista é corrigida pelo usuário, geram amostras rotuladas (detecções Verdadeiro Positivo com classificação incorreta). Os erros do tipo Falso Negativo, em que não há caixa associada, são apenas registrados nos metadados e não produzem rótulos no formato YOLO, não sendo, portanto, incorporados ao conjunto de treino. Dessa forma, o conjunto resultante reúne amostras candidatas a exemplos difíceis (*hard examples*) que, após curadoria, podem apoiar o re-treino voltado às limitações observadas em uso real. Há que se observar, ainda, os benefícios e limitações desse re-treino no contexto particular de cada aplicação em que a solução proposta está sendo adotada.

### 3.5. *Dashboard* de Análise

Para apoiar a interpretação dos registros coletados, o sistema oferece um *dashboard* que consolida as avaliações em indicadores e visualizações. A interface exibe o total de

predições avaliadas e os percentuais de predições corretas, incorretas, *feedbacks* indefinidos e casos sem detecção, além de gráficos de distribuição dos *feedbacks* e de erros por classe prevista. Um recurso adicional é a matriz de confusão construída a partir do *feedback* do usuário. Diferentemente da matriz obtida sobre o conjunto de teste, ela reflete o uso real, com a diagonal principal representando predições confirmadas como corretas e os valores fora da diagonal indicando correções manuais. O *dashboard* também lista registros recentes e permite exportar os dados em formato CSV para análise externa. Para fins de revisão e reprodutibilidade, o código-fonte e as instruções de execução do sistema estão disponíveis em um repositório[1].

### 3.6. Aspectos Éticos e de Privacidade

Como o sistema utiliza imagens faciais capturadas por *webcam*, foram considerados cuidados com a privacidade. A aplicação não grava o fluxo contínuo de vídeo nem armazena automaticamente todas as imagens capturadas. O armazenamento ocorre apenas quando o usuário marca uma predição como incorreta, situação em que o quadro, o rótulo corrigido e os metadados associados são salvos para análise posterior. O acesso à *webcam* ocorre mediante a autorização explícita pelo usuário no navegador. No que tange a este trabalho, a avaliação foi conduzida pelos próprios pesquisadores, sem coleta de dados faciais de terceiros, e as imagens geradas não serão disponibilizadas publicamente sem consentimento específico. Não foram implementados mecanismos de autenticação, de acesso a múltiplos usuários ou para sincronização em nuvem, por não serem necessários aos objetivos desta fase. Em estudos futuros com participantes, será necessário adotar termo de consentimento informado, política de retenção e exclusão das imagens e, conforme aplicável, apreciação por comitê de ética em pesquisa.

## 4. Avaliação e Resultados

A avaliação teve caráter funcional e qualitativo, com o objetivo de verificar o fluxo completo do sistema em execução, como captura, comunicação cliente-servidor, inferência, exibição, registro de *feedbacks* e consolidação no *dashboard*, e caracterizar o comportamento do modelo em uso real. Os testes foram conduzidos em ambiente local, em um notebook com processador Intel Core i5, 16 GB de RAM e GPU dedicada NVIDIA GeForce RTX 3050 Laptop com 6 GB VRAM, utilizando a *webcam* integrada (720p) e os navegadores Firefox e Chrome. Foram registrados 51 *feedbacks* pelo mecanismo de avaliação. Ressalta-se que esse volume, obtido em um cenário local de uso, não constitui validação estatística com múltiplos usuários. Os percentuais a seguir devem ser interpretados como indicativos do funcionamento do mecanismo de coleta e do *dashboard*, e não como medida conclusiva de desempenho do modelo.

Quanto ao procedimento, os *feedbacks* foram produzidos por um único usuário integrante da equipe de pesquisa, que reproduziu voluntariamente diante da câmera as nove categorias de expressão e avaliou cada predição apresentada pelo sistema. Como o usuário sabia qual expressão pretendia reproduzir, a classe de referência é *autodeclarada*, e não uma anotação independente; por isso, os acertos e erros registrados refletem a concordância entre a predição do modelo e a intenção do usuário, sem validação externa. A detecção considerou um único rosto por quadro, com limiares de confiança e de *IoU* configuráveis na interface, e a taxa de quadros por segundo foi reportada como média

[1] `https://github.com/tmelo-uea/eniac2026-fer-realtime-hitl`

móvel observada durante a execução contínua. A Tabela 2 resume a avaliação funcional dos módulos.

**Tabela 2. Avaliação funcional dos módulos do sistema.**

| Módulo | Resultado Observado |
|---|---|
| Captura por *webcam* | A câmera integrada foi acessada pelo navegador e o vídeo foi exibido corretamente na interface. |
| Comunicação cliente-servidor | A comunicação entre *frontend* e *backend* foi estabelecida, permitindo o envio dos quadros e o retorno das predições. |
| Inferência com YOLOv8n | O modelo realizou detecções em tempo real, retornando classe prevista, confiança e coordenadas da caixa delimitadora. |
| Exibição das predições | Caixas delimitadoras, rótulos e níveis de confiança foram exibidos sobre o vídeo capturado. |
| Registro de *feedback* | O sistema permitiu avaliar predições como corretas, incorretas, indefinidas ou sem detecção. |
| Persistência local | Os *feedbacks* foram armazenados localmente em SQLite. |
| *Dashboard* | Os registros foram consolidados em indicadores, gráficos e matriz de confusão baseada nos *feedbacks*. |

Um primeiro indício da distância entre *benchmark* e uso real aparece na vazão. Enquanto a medição isolada de inferência do YOLOv8n alcançou 56,17 FPS (Tabela 1), o sistema completo em execução, considerando captura, comunicação entre módulos, inferência e atualização da interface, apresentou uma média aproximada de 25 FPS em GPU e de apenas 7 FPS em CPU (Tabela 3). A partir da avaliação realizada, permite-se concluir que a métrica de *benchmark* superestimou em mais de duas vezes a vazão observada na aplicação, mesmo com aceleração por GPU, e a execução em CPU compromete a fluidez da interação. Esse resultado reforça que a viabilidade de um sistema de REF em tempo real depende não apenas da arquitetura do modelo, mas de todo o caminho de captura, transmissão e renderização.

**Tabela 3. Desempenho observado do sistema completo em execução local, contrastando com a medição isolada de inferência (56,17 FPS).**

| Dispositivo de inferência | FPS médio observado |
|---|---|
| GPU – NVIDIA GeForce RTX 3050 Laptop 6 GB | Aproximadamente 25 FPS |
| CPU – Intel Core i5-13420H | Aproximadamente 7 FPS |

Durante a execução, a interface exibiu, sobre o vídeo, as caixas delimitadoras das faces detectadas, a classe prevista e a confiança associada, além de métricas em tempo real, como dispositivo de inferência, latência estimada e taxa de quadros por segundo (Figura 3), permitindo acompanhar continuamente o comportamento do sistema.

Das 51 predições avaliadas, 42 foram confirmadas como corretas e 9 marcadas como incorretas, resultando em uma taxa indicativa de acerto de aproximadamente 82,4% e de erro de 17,6%; não houve, nesse conjunto, *feedbacks* indefinidos ou casos sem detecção (falso negativo). Os erros concentraram-se nas classes *sad* e *natural* (três ocorrências cada), seguidas de *angry* (duas) e *happy* (uma), padrão compatível com a maior ambiguidade visual e a baixa intensidade dessas expressões.

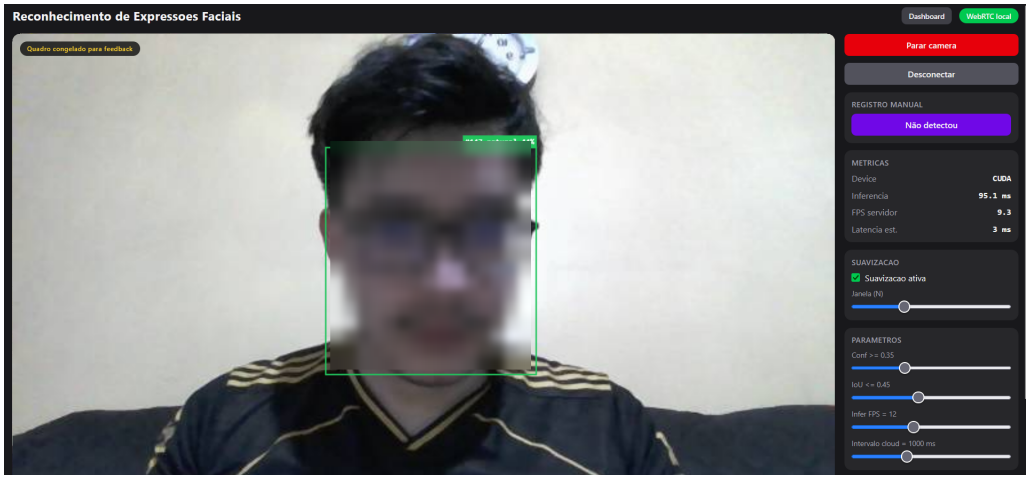



**Figura 3. Exemplo da interface em execução: detecção facial, classe prevista, confiança da predição e métricas de inferência em tempo real.**

A Figura 4 apresenta a matriz de confusão construída a partir dos *feedbacks*. A maior concentração de acertos ocorreu em *natural* (10 confirmações), seguida de *happy* (7), *angry* e *surprised* (6 cada), *sleepy* (5) e *sad* e *disgust* (4 cada). Os erros de classificação envolveram, sobretudo, predições de *sad* corrigidas como *fear* ou *sleepy*, de *natural* corrigidas como *angry*, *sad* ou *surprised*, e de *angry* corrigidas como *sad*. Como algumas classes (por exemplo, *contempt* e *fear*) tiveram poucos registros, a matriz deve ser lida como análise complementar e indicativa do comportamento em uso, e não como validação estatística por classe.

O *dashboard* consolidou esses registros, reunindo filtros por período, classe prevista e cenário, cartões de indicadores gerais e gráficos de distribuição dos *feedbacks* e de erros por classe, além da matriz de confusão da Figura 4. A ferramenta mostrou-se adequada para organizar os dados coletados e evidenciar padrões iniciais de acerto e erro, complementando a avaliação quantitativa realizada sobre o conjunto de teste.

A observação do uso foi especialmente importante por ter revelado fatores que as métricas de teste não capturaram. O mais ilustrativo foi o efeito da qualidade da *webcam*. Por exemplo, expressões como *disgust*, dificilmente reconhecidas pela *webcam* integrada do notebook, passaram a ser corretamente identificadas quando a captura foi feita pela câmera de um *smartphone* (Samsung Galaxy S25), mantendo-se o mesmo *backend* e o mesmo modelo. Embora pontual e não sistemático, esse teste indica que resolução, foco, nitidez e controle de exposição influenciam diretamente o desempenho percebido, sobretudo em expressões sutis. Também se observou que rotações laterais do rosto reduziram a estabilidade das detecções, levando, em alguns casos, à perda de detecção ou à baixa confiança. Esses comportamentos são coerentes com as dificuldades já conhecidas do REF *in-the-wild* [Liao et al. 2023, Lu et al. 2025] e mostram que o desempenho real depende de condições de captura, pose e intensidade da expressão que não se manifestam em um conjunto de teste fixo.

Ao converter predições incorretas em amostras rotuladas no formato YOLO (vide Seção 3.4), o sistema gerou candidatos a exemplos difíceis para as classes mais problemáticas (*sad*, *natural*, *angry*) em um insumo promissor para re-treino ou ajuste fino. Embora essas amostras ainda exijam curadoria, como a verificação de rótulos e o descarte




**Figura 4. Matriz de confusão baseada no *feedback* autodeclarado do usuário, construída durante o uso do sistema, sem validação independente. A diagonal indica predições confirmadas; fora da diagonal, correções manuais de classe.**

de exemplos de baixa qualidade, e o ganho efetivo do re-treino não tenha sido aferido neste estudo, o mecanismo consolida-se como uma solução prática para mitigar o desbalanceamento de classes e a lacuna entre *benchmark* e uso real evidenciada nas seções anteriores.

### 4.1. Discussão e Limitações

Os resultados obtidos reforçam a motivação central do trabalho, em que se constatou experimentalmente que métricas de teste informam pouco sobre o uso real de REF *in-the-wild*. A queda de vazão no fluxo completo, a parte mais robusta da avaliação, por independer do *feedback* autodeclarado, da sensibilidade à qualidade da câmera e à pose, e das confusões observadas em uso, sugere a necessidade de avaliar sistemas de REF para além de *benchmarks*. Ao mesmo tempo, o mecanismo *human-in-the-loop* contemplado na solução proposta oferece um encaminhamento prático para essa lacuna, ao capturar automaticamente os casos de falha como dados rotulados candidatos a re-treino.

As principais limitações dizem respeito à escala e à diversidade da avaliação. Os 51 *feedbacks* foram coletados por um único usuário, em ambiente local, com predominância da *webcam* de um único equipamento. Assim, os percentuais e a matriz de confusão de *feedback* têm valor indicativo, não conclusivo. Avaliações futuras com mais participantes, dispositivos e condições de captura permitiriam quantificar com maior segurança a lacuna observada e o ganho efetivo proporcionado pelo re-treino com os exemplos difíceis coletados.

## 5. Conclusão

Este trabalho apresentou o desenvolvimento de um sistema web de reconhecimento de expressões faciais em tempo real que vai além da inferência sobre o vídeo da *webcam*, incorporando um mecanismo de *feedback* com humano no laço capaz de transformar cada erro de classificação em um exemplo rotulado no formato YOLO, construindo automaticamente um conjunto de candidatos a exemplos difíceis para re-treino futuro. Integrando um modelo YOLOv8n a uma arquitetura cliente-servidor baseada em WebRTC e WebSocket, o sistema permitiu caracterizar a lacuna entre *benchmark* e uso real, no qual a vazão caiu de 56 FPS, medidos de forma isolada, para cerca de 25 FPS em GPU e 7 FPS em CPU no fluxo completo, e o reconhecimento de expressões sutis mostrou-se dependente da qualidade da câmera e da pose da face.

Mais do que demonstrar a viabilidade da aplicação, os resultados iniciais indicam que a avaliação de sistemas de REF não deveria encerrar-se nas métricas de teste e que o próprio uso pode fornecer dados para o aprimoramento futuro do modelo. Reconhece-se como principal limitação deste trabalho a escala reduzida da avaliação. Em trabalhos futuros, pretende-se ampliar a avaliação com múltiplos usuários, dispositivos e modelos, quantificar o ganho de re-treino incremental a partir dos candidatos a exemplos difíceis coletados e investigar estratégias de mitigação de desbalanceamento e de robustez a variações de pose, iluminação e intensidade da expressão.

## Declaração de Uso de Inteligência Artificial

Ferramentas de inteligência artificial generativa (Gemini 3.1 Pro) foram utilizadas como apoio à redação e edição do texto deste artigo (organização, revisão de estilo e clareza). A concepção da pesquisa, a metodologia, os experimentos, a análise dos resultados e as conclusões são de responsabilidade dos autores, que revisaram integralmente o conteúdo e assumem total responsabilidade por sua exatidão e originalidade. Nenhuma ferramenta de IA é considerada autora deste trabalho.

## Referências


Alshammari, A. and Alshammari, M. E. (2024). Emotional facial expression detection using YOLOv8. *Engineering, Technology & Applied Science Research*, 14(5):16619–16623.

Jegham, N., Koh, C. Y., Abdelatti, M. F., and Hendawi, A. (2025). Yolo evolution: A comprehensive benchmark and architectural review of yolov12, yolo11, and their previous versions. Preprint.

Jocher, G., Chaurasia, A., and Qiu, J. (2023). Ultralytics yolov8.

Kumari, J., Rajesh, R., and Pooja, K. M. (2015). Facial expression recognition: A survey. *Procedia Computer Science*, 58:486–491.

Li, W., Wang, J., Zhang, L., Zeng, X., Chen, G., and Long, H. (2024). Research on facial expression recognition based on improved yolov8. In *Proc. SPIE 13230, Third International Conference on Machine Vision, Automatic Identification, and Detection (MVAID 2024)*.

Liao, J., Lin, Y., Ma, T., He, S., Liu, X., and He, G. (2023). Facial expression recognition methods in the wild based on fusion feature of attention mechanism and lbp. *PMC*, 2023:1–21.

Liu, P., Qi, B., and Banerjee, S. (2018). Edgeeye: An edge service framework for real-time intelligent video analytics. In *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, EdgeSys '18, pages 1–6, New York, NY, USA. Association for Computing Machinery.

Lu, H., Xu, J., and Zheng, H. (2025). Explore the facial expression recognition in different complex environment. *Applied and Computational Engineering*, 174:77–85.

Mahmoud, H. and Abozariba, R. (2025). A systematic review on webrtc for potential applications and challenges beyond audio video streaming. *Multimedia Tools and Applications*, 84:2909–2946.

Mollahosseini, A., Hasani, B., and Mahoor, M. H. (2017). Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 10(1):18–31.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Ricardo, P. and de Melo, T. (2026). Reconhecimento de expressoes faciais em tempo real com yolo: Estudo comparativo em cenários nao controlados. In *Seminário Integrado de Software e Hardware (SEMISH)*, pages 998–1003. SBC.

Sareen, V. and Seeja, K. R. (2025). Video-based facial emotion recognition using YOLO and Vision Transformer. In *EPJ Web of Conferences*, volume 328, page 01040, Les Ulis, France. EDP Sciences. Published online: 18 June 2025.

Sassu, A., Saenz-Cogollo, J. F., and Agelli, M. (2021). Deep-framework: A distributed, scalable, and edge-oriented framework for real-time analysis of video streams. *Sensors*, 21(12):4045.

Shao, J. and Qian, Y. (2019). Three convolutional neural network models for facial expression recognition in the wild. *Neurocomputing*, 355:82–92.

Taware, S. and Thakare, A. (2023). Critical analysis on multimodal emotion recognition in meeting the requirements for next generation human computer interactions. *International Journal on Recent and Innovation Trends in Computing and Communication*, 11(9s):523–531.

Wu, T. (2024). Reducing the cost of test data labelling for deep-learning systems: An empirical study. Master's thesis, School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Ontario, Canada. Disponível em `https://doi.org/10.20381/ruor-30788`. Acesso em: 15 de maio de 2026.

Zhong, H., Han, T., Xia, W., Tian, Y., and Wu, L. (2023). Research on real-time teachers' facial expression recognition based on yolov5 and attention mechanisms. *EURASIP Journal on Advances in Signal Processing*, 2023(55):1–17.