

**UNIVERSIDADE DO ESTADO DO AMAZONAS  
ESCOLA SUPERIOR DE TECNOLOGIA**

**PAULO RICARDO FERREIRA DA SILVA**

**Reconhecimento de Expressões Faciais em Tempo Real com YOLO: Estudo  
Comparativo e Desenvolvimento de um Protótipo Web**

Manaus  
2026

**PAULO RICARDO FERREIRA DA SILVA**

**Reconhecimento de Expressões Faciais em Tempo Real com YOLO: Estudo  
Comparativo e Desenvolvimento de um Protótipo Web**

Orientador: Tiago Eugenio de Melo, Dr.

### Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).  
**Sistema Integrado de Bibliotecas da Universidade do Estado do Amazonas.**

|       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| S586r | <p>Silva, Paulo Ricardo Ferreira da</p> <p>Reconhecimento de Expressões Faciais em Tempo Real com YOLO: Estudo Comparativo e Desenvolvimento de um Protótipo Web / Paulo Ricardo Ferreira da Silva. Manaus : [s.n], 2026.</p> <p>70 f.: color.; 21.0 cm.</p> <p>TCC - Graduação em Engenharia de Computação- Universidade do Estado do Amazonas, Manaus, 2026.</p> <p>Orientador: Tiago Eugenio de Melo.</p> <p>1. YOLO. 2. Reconhecimento de expressões faciais. 3. Visão computacional. 4. Tempo real. 5. Protótipo web. I. Tiago Eugenio de Melo (Orient.) II. Universidade do Estado do Amazonas. III. Título</p> <p>CDU(1997)62:004</p> |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



## FOLHA DE APROVAÇÃO

### Reconhecimento de Expressões Faciais em Tempo Real com YOLO: Desenvolvimento e Avaliação de um Protótipo Web

**Paulo Ricardo Ferreira Da Silva**

Trabalho de Conclusão de Curso defendido e aprovado pela banca avaliadora constituída pelos professores:

---

Tiago Eugenio de Melo  
Presidente

---

Elloa Guedes  
Avaliador

---

Mário Bessa  
Avaliador

Manaus, 09 de junho de 2026.

## AGRADECIMENTOS

Agradeço primeiramente a Deus, pela força, saúde e perseverança ao longo desta caminhada.

Aos meus familiares, pelo amor, apoio e incentivo em todos os momentos, especialmente nos períodos de maior dificuldade.

Ao meu orientador, Prof. Dr. Tiago de Melo, pelas orientações, pela disponibilidade e pelas contribuições fundamentais para o desenvolvimento deste trabalho.

Aos professores do curso de Engenharia da Computação, pelos ensinamentos transmitidos ao longo da graduação e pela contribuição à minha formação acadêmica e profissional.

Aos amigos da faculdade, pelo companheirismo, pelas trocas de conhecimento e pelos momentos compartilhados durante essa trajetória.

Aos amigos de fora, pelo apoio, incentivo e presença nos momentos importantes.

A todos que, de alguma forma, contribuíram para a realização deste trabalho, deixo meu sincero agradecimento.

## RESUMO

Este trabalho aborda o reconhecimento automático de expressões faciais em condições não controladas, um desafio relevante para aplicações em interação humano-computador, segurança, saúde e sistemas interativos em tempo real. Neste contexto, investigou-se o uso de modelos baseados na arquitetura *You Only Look Once* (YOLO), capazes de realizar detecção e classificação em uma única etapa, combinando rapidez e precisão. A metodologia foi conduzida em duas etapas complementares. Na primeira, foram treinadas e avaliadas comparativamente quatro versões da família YOLO, especificamente YOLOv8n, YOLOv10n, YOLOv11n e YOLOv12n, utilizando o conjunto de dados público “9 Facial Expressions for YOLO”, composto por imagens em cenários não controlados e nove categorias de expressões faciais. A avaliação considerou métricas como precisão, revocação, mAP, FPS, GFLOPs e matrizes de confusão. Na segunda etapa, o modelo selecionado foi integrado a um protótipo web para reconhecimento de expressões faciais em tempo real, com captura por *webcam*, inferência no *backend*, exibição das predições na interface, registro de *feedbacks* do usuário e visualização dos dados em um *dashboard*. Os resultados mostraram que o YOLOv12n apresentou leve vantagem preditiva nos experimentos realizados, enquanto o YOLOv10n apresentou a maior velocidade de inferência. Entretanto, o YOLOv8n foi selecionado para a etapa aplicada por apresentar o melhor equilíbrio entre acurácia, velocidade e viabilidade prática, alcançando mAP50 de 0,837, mAP50–95 de 0,681 e 56,17 FPS. O protótipo desenvolvido demonstrou funcionamento adequado no fluxo de captura, inferência, exibição das predições, registro de *feedbacks* e análise agregada dos resultados. Observou-se, contudo, que expressões mais sutis ou ambíguas, variações de pose, iluminação e qualidade da câmera ainda influenciam o desempenho percebido do sistema. Assim, o trabalho contribui tanto com uma análise comparativa de arquiteturas YOLO aplicadas ao reconhecimento de expressões faciais quanto com a implementação de um protótipo funcional para inferência em tempo real.

**Palavras-chave:** *YOLO*, reconhecimento de expressões faciais, visão computacional, tempo real, protótipo web.

## ABSTRACT

This work addresses automatic facial expression recognition under uncontrolled conditions, a relevant challenge for applications in human-computer interaction, security, healthcare, and real-time interactive systems. In this context, this study investigated the use of models based on the *You Only Look Once* (YOLO) architecture, which performs detection and classification in a single stage, combining speed and accuracy. The methodology was conducted in two complementary stages. First, four versions of the YOLO family were trained and comparatively evaluated, specifically YOLOv8n, YOLOv10n, YOLOv11n, and YOLOv12n, using the public dataset “*9 Facial Expressions for YOLO*”, composed of images captured in uncontrolled scenarios and covering nine facial expression categories. The evaluation considered metrics such as precision, recall, mAP, FPS, GFLOPs, and confusion matrices. In the second stage, the selected model was integrated into a web prototype for real-time facial expression recognition, including webcam capture, backend inference, prediction display in the interface, user feedback registration, and data visualization through a dashboard. The results indicated a slight predictive advantage of YOLOv12n in the experiments performed, while YOLOv10n presented the highest inference speed. However, YOLOv8n was selected for the applied stage because it provided the best balance between accuracy, speed, and practical feasibility, achieving an mAP<sub>50</sub> of 0.837, an mAP<sub>50–95</sub> of 0.681, and 56.17 FPS. The developed prototype demonstrated adequate functionality in the complete flow of capture, inference, prediction display, feedback registration, and aggregated result analysis. Nevertheless, it was observed that more subtle or ambiguous expressions, pose variations, lighting conditions, and camera quality still influence the perceived performance of the system. Therefore, this work contributes both with a comparative analysis of YOLO architectures applied to facial expression recognition and with the implementation of a functional prototype for real-time inference.

**Keywords:** *YOLO*, facial expression recognition, computer vision, real time, web prototype.

## Lista de Figuras

|    |                                                                                                                                                                                                                                                                                                                                               |    |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1  | Ilustração do funcionamento da arquitetura YOLO. O modelo divide a imagem em uma grade $S \times S$ , onde cada célula prediz $B$ caixas delimitadoras, níveis de confiança e probabilidades de classe. Figura adaptada de Redmon et al. (2016)(REDMON et al., 2016). . . . .                                                                 | 21 |
| 2  | Arquitetura detalhada do YOLOv8, destacando o <i>backbone</i> com múltiplas camadas convolucionais hierárquicas, a <i>Feature Pyramid Network</i> (FPN) que integra informações em diferentes escalas e a <i>head</i> de detecção responsável pelas predições finais. Figura adaptada de Jegham et al. (2025) (JEGHAM et al., 2025) . . . . . | 22 |
| 3  | Proporção dos conjuntos de treino, validação e teste no dataset. . . . .                                                                                                                                                                                                                                                                      | 31 |
| 4  | Amostras do conjunto de dados “9 Facial Expressions for YOLO”, ilustrando variedade de expressões e condições de captura. . . . .                                                                                                                                                                                                             | 32 |
| 5  | Arquitetura geral do protótipo desenvolvido para reconhecimento de expressões faciais em tempo real. . . . .                                                                                                                                                                                                                                  | 37 |
| 6  | Matriz de confusão – YOLOv8n. . . . .                                                                                                                                                                                                                                                                                                         | 46 |
| 7  | Matriz de confusão – YOLOv10n. . . . .                                                                                                                                                                                                                                                                                                        | 48 |
| 8  | Matriz de confusão – YOLOv11n. . . . .                                                                                                                                                                                                                                                                                                        | 50 |
| 9  | Matriz de confusão – YOLOv12n. . . . .                                                                                                                                                                                                                                                                                                        | 52 |
| 10 | Interface principal do protótipo em execução, com detecção facial, classe prevista, confiança da predição e métricas de inferência. . . . .                                                                                                                                                                                                   | 57 |
| 11 | Etapas do mecanismo de feedback do usuário: à esquerda, confirmação da predição realizada pelo modelo; à direita, seleção da classe correta quando a predição é marcada como incorreta. . . . .                                                                                                                                               | 59 |
| 12 | Dashboard de análise dos feedbacks coletados durante o uso do protótipo. . . . .                                                                                                                                                                                                                                                              | 59 |
| 13 | Matriz de confusão construída a partir dos feedbacks registrados pelo usuário. . . . .                                                                                                                                                                                                                                                        | 61 |

## **Lista de Abreviaturas e Siglas**

API *Application Programming Interface*

CNN *Convolutional Neural Network*

CSV *Comma-Separated Values*

FPS *Frames Per Second*

GFLOPs *Giga Floating Point Operations per Second*

GPU *Graphics Processing Unit*

HTTP *Hypertext Transfer Protocol*

IoU *Intersection over Union*

mAP *mean Average Precision*

REF Reconhecimento de Expressões Faciais

WebRTC *Web Real-Time Communication*

YOLO *You Only Look Once*

## SUMÁRIO

|          |                                                                                         |           |
|----------|-----------------------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>INTRODUÇÃO</b>                                                                       | <b>12</b> |
| 1.1      | Justificativa                                                                           | 13        |
| 1.2      | Objetivo Geral                                                                          | 15        |
| 1.3      | Objetivos Específicos                                                                   | 15        |
| 1.4      | Organização do Trabalho                                                                 | 15        |
| <b>2</b> | <b>FUNDAMENTAÇÃO TEÓRICA</b>                                                            | <b>17</b> |
| 2.1      | Reconhecimento de Expressões Faciais (REF)                                              | 17        |
| 2.1.1    | Emoções básicas e classes utilizadas em sistemas modernos de reconheci-<br>mento facial | 18        |
| 2.2      | Aprendizado Profundo ( <i>Deep Learning</i> )                                           | 19        |
| 2.3      | Redes Neurais Convolucionais (CNNs)                                                     | 19        |
| 2.4      | A Arquitetura YOLO ( <i>You Only Look Once</i> )                                        | 20        |
| 2.4.1    | Evolução das versões YOLO utilizadas (v8, v10, v11 e v12)                               | 23        |
| 2.5      | Inferência em Tempo Real em Aplicações Web                                              | 24        |
| <b>3</b> | <b>TRABALHOS RELACIONADOS</b>                                                           | <b>26</b> |
| 3.1      | YOLO                                                                                    | 26        |
| 3.2      | Reconhecimento de Expressões Faciais (REF)                                              | 27        |
| 3.3      | Aplicações web para análise de vídeo em tempo real                                      | 28        |
| <b>4</b> | <b>METODOLOGIA</b>                                                                      | <b>30</b> |
| 4.1      | Visão geral da metodologia                                                              | 30        |
| 4.2      | Etapa 1 - avaliação comparativa das arquiteturas YOLO                                   | 30        |
| 4.2.1    | Conjunto de Dados                                                                       | 30        |
| 4.2.2    | Escolha do dataset                                                                      | 33        |
| 4.2.3    | Arquiteturas Avaliadas                                                                  | 34        |
| 4.2.4    | Protocolo de Treinamento                                                                | 34        |
| 4.2.5    | Ambiente Computacional                                                                  | 35        |
| 4.2.6    | Métricas de Avaliação                                                                   | 35        |
| 4.3      | Etapa 2 – Desenvolvimento do protótipo em tempo real                                    | 36        |
| 4.3.1    | Modelo utilizado no protótipo                                                           | 36        |
| 4.3.2    | Arquitetura geral do protótipo                                                          | 37        |
| 4.3.3    | Tecnologias empregadas                                                                  | 37        |
| 4.3.4    | Fluxo de inferência em tempo real                                                       | 38        |
| 4.3.5    | Mecanismo de feedback do usuário                                                        | 39        |

|          |                                                        |           |
|----------|--------------------------------------------------------|-----------|
| 4.3.6    | Dashboard de análise dos feedbacks . . . . .           | 39        |
| 4.3.7    | Persistência dos dados e registros coletados . . . . . | 40        |
| 4.3.8    | Aspectos éticos e privacidade . . . . .                | 41        |
| 4.3.9    | Ambiente local de execução e testes . . . . .          | 42        |
| 4.3.10   | Disponibilização da ferramenta . . . . .               | 43        |
| 4.3.11   | Procedimento de avaliação do protótipo . . . . .       | 44        |
| <b>5</b> | <b>RESULTADOS . . . . .</b>                            | <b>45</b> |
| 5.1      | Protocolo Experimental . . . . .                       | 45        |
| 5.2      | Métricas e Ambiente . . . . .                          | 45        |
| 5.3      | Resultados dos Modelos . . . . .                       | 45        |
| 5.4      | Comparação Geral entre Modelos . . . . .               | 53        |
| 5.5      | Seleção do modelo para o protótipo . . . . .           | 55        |
| 5.6      | Resultados do protótipo em tempo real . . . . .        | 55        |
| 5.6.1    | Funcionamento geral da aplicação . . . . .             | 56        |
| 5.6.2    | Avaliação funcional dos módulos do protótipo . . . . . | 57        |
| 5.6.3    | Resultados do mecanismo de feedback . . . . .          | 58        |
| 5.6.4    | Análise do dashboard . . . . .                         | 59        |
| 5.6.5    | Limitações observadas . . . . .                        | 62        |
| <b>6</b> | <b>CONCLUSÕES . . . . .</b>                            | <b>64</b> |
| 6.1      | Objetivos alcançados . . . . .                         | 65        |
| 6.2      | Trabalhos futuros . . . . .                            | 66        |
|          | <b>REFERÊNCIAS . . . . .</b>                           | <b>68</b> |

# 1 INTRODUÇÃO

O reconhecimento de expressões faciais é uma área central em visão computacional e computação afetiva, com aplicações que vão desde a interação humano-computador até contextos de segurança e pesquisa psicológica (ALSHAMMARI; ALSHAMMARI, 2024). A transição de abordagens tradicionais, que extraíam elementos pré-definidos da face, como contorno e texturas (KUMARI; RAJESH; POOJA, 2015), para modelos capazes de aprender a reconhecer padrões complexos diretamente a partir de exemplos visuais, abordagem conhecida como aprendizado profundo (*deep learning*), marcou um avanço significativo nesse cenário. No entanto, sua utilização em cenários do mundo real ainda traz alguns desafios, como variações de iluminação, pose da cabeça e oclusões parciais, que exigem alta capacidade de generalização dos modelos (LI; DENG, 2018).

Um dos principais obstáculos para o avanço nessa área de reconhecimento de expressões foi a falta de bases de dados de treinamento que fossem grandes o suficiente (MOLLAHOSSEINI; HASANI; MAHOOR, 2017). Modelos de *deep learning*, por sua natureza, precisam de um volume massivo de exemplos para aprender padrões distintos e evitar o sobreajuste (*overfitting*). A recente disponibilização de *datasets* com dezenas de milhares de imagens em cenários naturais, como o utilizado neste trabalho, representa uma mudança no cenário, possibilitando o treinamento de arquiteturas mais profundas e com maior poder de generalização.

Contudo, a superação do desafio dos dados não é suficiente para garantir a aplicabilidade prática. A implementação de sistemas interativos e de monitoramento precisa que o modelo funcione em tempo real, um requisito que gera grandes restrições computacionais (SHAO; QIAN, 2019). Nesse contexto, modelos de estágio único (*one-stage*), como a arquitetura *You Only Look Once* (YOLO) (REDMON et al., 2016), surgem como uma solução de alta performance, pois analisam a imagem inteira de uma vez, unindo a detecção facial e a classificação de expressões em uma única etapa rápida, diferente dos modelos de múltiplos estágios, que examinavam a imagem em partes separadas antes de classificar.

Diante do exposto, este trabalho propõe o desenvolvimento e a avaliação de um sistema de reconhecimento de expressões faciais que aborda simultaneamente os desafios de generalização, eficiência e aplicabilidade prática. Utilizando a arquitetura YOLO e o *dataset* público “9 Facial Expressions for YOLO”, disponível no Kaggle, diferentes versões da família YOLO são treinadas e avaliadas para classificar nove categorias de expressões faciais: raiva (*angry*), desprezo (*contempt*), nojo (*disgust*), medo (*fear*), felicidade (*happy*), neutro (*natural*), tristeza (*sad*), sonolência (*sleepy*) e surpresa (*surprised*). O objetivo é identificar um modelo capaz de alcançar bom desempenho mesmo em condições desafiadoras, mantendo-se rápido e eficiente o suficiente para aplicações em tempo real.

A partir da análise comparativa entre as versões avaliadas, o modelo com melhor equilíbrio entre desempenho preditivo, velocidade de inferência e custo computacional é selecionado para a etapa aplicada do trabalho. Essa escolha permite avançar da avaliação experimental para a construção de uma solução funcional, verificando se o modelo treinado mantém sua viabilidade quando utilizado fora do ambiente controlado de treinamento e teste.

Nesse sentido, este trabalho também contempla o desenvolvimento de um protótipo web para reconhecimento de expressões faciais em tempo real. O sistema integra a captura de imagens por *webcam*, o envio dos quadros para processamento no *backend*, a execução da inferência com o modelo selecionado e a exibição dos resultados em uma interface web. A aplicação permite visualizar o rosto detectado, a expressão facial reconhecida, o nível de confiança da predição e as caixas delimitadoras geradas pelo modelo.

Além da exibição das predições em tempo real, o protótipo inclui um mecanismo de registro de *feedbacks* do usuário e um *dashboard* para apresentação agregada dessas informações. Esses recursos permitem observar o comportamento do sistema em um fluxo de uso mais próximo de uma aplicação real, contribuindo para identificar padrões de acerto, falhas recorrentes e limitações relacionadas a fatores como iluminação, pose facial, variação das expressões e condições de captura.

Dessa forma, a contribuição deste trabalho está na integração entre uma etapa experimental comparativa e uma etapa prática de desenvolvimento. A primeira permite avaliar diferentes versões da arquitetura YOLO para o reconhecimento de expressões faciais em cenários não controlados; a segunda demonstra a aplicação do modelo selecionado em um protótipo web funcional, voltado à inferência em tempo real e à análise do comportamento do sistema em um cenário funcional de uso.

## 1.1 JUSTIFICATIVA

O crescente interesse na “Computação Afetiva” tem posicionado o Reconhecimento de Expressões Faciais (REF) como uma tecnologia relevante para a próxima geração de sistemas interativos (TAWARE; THAKARE, 2023). A possibilidade de máquinas interpretarem sinais emocionais humanos amplia o potencial de aplicações em diferentes contextos, como plataformas educacionais adaptativas, ferramentas de apoio à saúde, sistemas de segurança, ambientes de monitoramento e experiências de entretenimento mais imersivas. Nesse sentido, o REF contribui para tornar a interação humano-computador mais natural, sensível ao contexto e capaz de responder de forma mais adequada ao comportamento do usuário.

Apesar dos avanços recentes, a aplicação do reconhecimento de expressões faciais em situações reais ainda apresenta desafios significativos. Modelos de *deep learning* tendem a sofrer perdas de desempenho quando submetidos a condições não controladas, como

variações de iluminação, mudanças na pose da cabeça, oclusões parciais da face, diferenças demográficas e expressões de baixa intensidade (LIAO et al., 2023; LU; XU; ZHENG, 2025). Essas condições afetam diretamente a robustez e a precisão dos sistemas, tornando necessário avaliar arquiteturas capazes de manter bom desempenho fora de ambientes controlados.

Diante desse cenário, a primeira justificativa deste trabalho está na necessidade de investigar modelos capazes de equilibrar acurácia e eficiência computacional para o reconhecimento de expressões faciais em tempo real. A arquitetura YOLO mostra-se adequada a esse propósito por realizar detecção e classificação em uma única etapa, reduzindo o tempo de processamento e favorecendo aplicações que exigem respostas rápidas. Assim, a comparação entre diferentes versões da família YOLO permite identificar qual arquitetura apresenta melhor relação entre precisão, velocidade de inferência e custo computacional para a tarefa proposta.

Além disso, a utilização de um conjunto de dados amplo e construído para cenários não controlados contribui para uma avaliação mais próxima das condições encontradas em aplicações reais. Ao considerar múltiplas expressões faciais, incluindo classes mais sutis ou menos representadas, como *contempt*, *natural* e *sleepy*, o estudo permite analisar não apenas o desempenho global dos modelos, mas também suas limitações diante de categorias visualmente ambíguas ou com menor quantidade de exemplos. Essa análise é importante para compreender em quais situações o modelo apresenta maior confiabilidade e em quais casos ainda há maior propensão a erros.

A segunda justificativa está na continuidade prática da pesquisa. Após a etapa comparativa, torna-se necessário verificar se o modelo selecionado mantém sua viabilidade quando integrado a uma aplicação funcional. Dessa forma, o desenvolvimento de um protótipo web permite avaliar o comportamento do YOLOv8n em um fluxo real de uso, envolvendo captura de imagens por *webcam*, envio dos quadros ao *backend*, execução da inferência e apresentação das predições ao usuário em tempo real. Essa etapa aproxima o estudo de uma aplicação prática, indo além da avaliação em ambiente experimental.

Por fim, a inclusão de mecanismos de *feedback* e de um *dashboard* amplia a contribuição do trabalho ao permitir o registro e a visualização das respostas do usuário sobre as predições realizadas. Esses recursos favorecem a análise qualitativa do funcionamento do sistema, possibilitando observar padrões de acerto, falhas recorrentes e limitações percebidas durante o uso. Assim, este trabalho se justifica por unir uma etapa experimental comparativa, voltada à seleção de um modelo eficiente, a uma etapa aplicada, voltada à construção e avaliação de um protótipo web para reconhecimento de expressões faciais em tempo real.

## 1.2 OBJETIVO GERAL

Desenvolver e avaliar um sistema de reconhecimento de expressões faciais em tempo real baseado em modelos da família YOLO, por meio da comparação entre diferentes arquiteturas em um conjunto de dados de larga escala em condições não controladas e da integração do modelo selecionado a um protótipo web funcional.

## 1.3 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste estudo são:

- Revisar estudos sobre reconhecimento de expressões faciais, aprendizado profundo, redes neurais convolucionais e arquiteturas YOLO aplicadas a tarefas de detecção e classificação.
- Explorar o conjunto de dados “9 Facial Expressions for YOLO”, analisando sua estrutura, classes, anotações, divisão dos dados e distribuição das amostras por categoria.
- Implementar e treinar diferentes versões da arquitetura YOLO para detectar rostos e classificar múltiplas expressões faciais em condições não controladas.
- Avaliar quantitativa e qualitativamente os modelos treinados, considerando métricas de acurácia, velocidade de inferência, custo computacional, matrizes de confusão e desempenho por classe.
- Comparar as versões YOLO avaliadas e selecionar o modelo com melhor equilíbrio entre precisão, velocidade e viabilidade para aplicações em tempo real.
- Integrar o modelo selecionado a um protótipo web composto por *frontend* e *backend*, permitindo a captura de quadros por *webcam*, o envio das imagens para inferência e a exibição das predições ao usuário.
- Desenvolver uma interface para visualização em tempo real do rosto detectado, da expressão facial reconhecida, do nível de confiança e das caixas delimitadoras geradas pelo modelo.
- Implementar um mecanismo de registro de *feedbacks* do usuário sobre as predições realizadas e um *dashboard* para apresentação agregada dessas informações.
- Avaliar funcionalmente o protótipo em ambiente local, considerando aspectos como captura por *webcam*, comunicação entre *frontend* e *backend*, exibição das predições, registro de *feedbacks*, desempenho da aplicação e limitações observadas.

## 1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado da seguinte forma: o Capítulo 2 apresenta a fundamentação teórica necessária para compreender o estudo proposto, incluindo conceitos

de reconhecimento de expressões faciais, aprendizado profundo, redes neurais convolucionais e a arquitetura YOLO. O Capítulo 3 reúne os principais trabalhos relacionados, destacando abordagens aplicadas ao reconhecimento de expressões faciais e ao uso de arquiteturas YOLO em tarefas de detecção e classificação.

O Capítulo 4 descreve a metodologia adotada, abrangendo a avaliação comparativa das arquiteturas YOLO, o conjunto de dados utilizado, o protocolo de treinamento, as métricas de avaliação e o desenvolvimento do protótipo web em tempo real. O Capítulo 5 apresenta os resultados obtidos, incluindo o desempenho dos modelos avaliados, a comparação entre as arquiteturas, a seleção do modelo utilizado no protótipo e a análise funcional da aplicação desenvolvida. Por fim, o Capítulo 6 apresenta as conclusões do trabalho, os objetivos alcançados, as limitações observadas e possibilidades de trabalhos futuros.

## 2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos fundamentais que sustentam o desenvolvimento deste trabalho. São abordados o reconhecimento de expressões faciais (REF), os princípios do aprendizado profundo, as Redes Neurais Convolucionais (CNNs), a arquitetura YOLO e os conceitos relacionados à inferência em tempo real em aplicações web, que servem como base para a avaliação comparativa dos modelos e para o desenvolvimento do protótipo proposto.

### 2.1 RECONHECIMENTO DE EXPRESSÕES FACIAIS (REF)

O reconhecimento de expressões faciais é um campo de pesquisa que busca identificar automaticamente padrões de expressões faciais associados a categorias emocionais a partir de imagens ou vídeos da face. Essa área tem ganhado destaque por suas diversas aplicações, que vão desde sistemas de segurança e monitoramento até plataformas de ensino, entretenimento e saúde digital (ALSHAMMARI; ALSHAMMARI, 2024).

Os primeiros métodos de REF, como LBP (*Local Binary Patterns*) e HOG (*Histogram of Oriented Gradients*), utilizavam técnicas tradicionais de processamento de imagem, exigindo que o pesquisador definisse explicitamente como extrair bordas, texturas e formas, que depois eram classificadas por algoritmos como SVM ou Random Forest (KUMARI; RAJESH; POOJA, 2015). Embora esses métodos apresentassem bons resultados em ambientes controlados, seu desempenho caía significativamente em situações reais, com variações de iluminação, ângulo da cabeça, oclusões (como óculos e mãos) e diferenças culturais ou demográficas (LI; DENG, 2018).

Com o avanço do aprendizado profundo, os modelos passaram a aprender diretamente os padrões faciais a partir dos dados, superando boa parte das limitações dos métodos manuais. Ainda assim, a qualidade e a diversidade das bases de dados permanecem fatores decisivos. Durante muito tempo, a escassez de grandes conjuntos de imagens anotadas limitava o desempenho desses modelos. A criação de bases amplas e realistas, como a AffectNet, representou um avanço importante, permitindo treinar redes mais robustas e capazes de generalizar para diferentes contextos (MOLLAHOSSEINI; HASANI; MAHOOR, 2017).

Mais recentemente, o uso de arquiteturas de detecção e classificação simultâneas — como o YOLO — vem ganhando espaço no REF. Essas abordagens unificam em uma única etapa a detecção do rosto e a classificação da expressão facial, reduzindo o tempo de processamento e tornando possível o reconhecimento em tempo real. Assim, o REF passa de uma tarefa puramente analítica para uma aplicação prática, capaz de operar de forma eficiente em ambientes não controlados, como vídeos ao vivo e interações homem-máquina.

Neste trabalho, o termo expressão facial refere-se ao padrão visual observado na face e associado a uma classe do conjunto de dados, não implicando a inferência direta ou definitiva do estado emocional real do indivíduo. Assim, as classes utilizadas representam categorias visuais de expressões faciais, como *angry*, *happy*, *sad* e *natural*, e não uma confirmação psicológica do estado emocional da pessoa analisada.

### 2.1.1 Emoções básicas e classes utilizadas em sistemas modernos de reconhecimento facial

O estudo das expressões faciais foi amplamente influenciado pelos trabalhos de Paul Ekman, que propôs a existência de um conjunto de seis emoções básicas universais: raiva, nojo, medo, tristeza, felicidade e surpresa (EKMAN, 1992). Essas emoções seriam reconhecidas de maneira consistente em diferentes culturas, sustentando a hipótese de que certos padrões musculares faciais correspondem a estados emocionais biologicamente determinados. Posteriormente, alguns autores passaram a incluir também a expressão de desprezo, que apresenta ativação muscular relativamente estável entre diferentes grupos culturais (MATSUMOTO, 1992).

Embora o modelo clássico forneça a base conceitual da área, aplicações contemporâneas de reconhecimento automático de expressões faciais utilizam conjuntos de classes mais amplos. Em particular, sistemas baseados em visão computacional frequentemente incluem a classe neutra, que não representa uma emoção, mas atua como referência para distinguir a ausência de ativação emocional (PANTIC; ROTHKRANTZ, 2000). Também é comum que bases de dados modernas adicionem expressões funcionais ou estados comportamentais não contemplados na teoria psicológica original, mas relevantes no contexto de reconhecimento automático, como cansaço, dor, esforço ou sonolência.

O conjunto de dados utilizado neste trabalho segue essa tendência, apresentando nove categorias: *angry*, *contempt*, *disgust*, *fear*, *happy*, *natural*, *sad*, *sleepy* e *surprised*. Observa-se que parte dessas classes corresponde diretamente às emoções básicas descritas por Ekman, enquanto outras refletem adaptações feitas para fins práticos em sistemas computacionais. A inclusão das classes *natural* e *sleepy*, por exemplo, não está prevista no modelo psicológico tradicional, mas é recorrente em bases desenvolvidas para algoritmos de aprendizado profundo, que se beneficiam da diversidade de estados faciais presentes em cenários não controlados (LI; DENG, 2018; MOLLAHOSSEINI; HASANI; MAHOOR, 2017).

Essa divergência entre o modelo teórico e os conjuntos de classes adotados em bases modernas não representa uma inconsistência, mas uma ampliação do escopo do reconhecimento automático. Enquanto a psicologia busca identificar emoções universais, a visão computacional prioriza a variabilidade visual e a capacidade dos sistemas de distinguir padrões faciais relevantes para aplicações reais. Assim, o uso das nove classes presen-

tes no dataset é coerente com a literatura contemporânea de *deep learning* aplicada ao reconhecimento de expressões faciais.

## 2.2 APRENDIZADO PROFUNDO (*DEEP LEARNING*)

O aprendizado profundo revolucionou a forma como os sistemas de visão computacional são desenvolvidos, ao permitir que máquinas aprendam representações complexas e hierárquicas diretamente dos dados (LECUN; BENGIO; HINTON, 2015). Diferente das abordagens tradicionais, em que as características eram extraídas manualmente, o aprendizado profundo utiliza redes neurais com múltiplas camadas para aprender automaticamente representações abstratas — desde padrões simples, como bordas e cores, até conceitos de alto nível, como formas e objetos completos.

O princípio central do aprendizado profundo é o treinamento de redes neurais compostas por diversas camadas não lineares, conhecidas como *deep neural networks*. Cada camada aprende uma transformação progressiva dos dados, refinando a informação até atingir uma representação útil para a tarefa desejada (GOODFELLOW; BENGIO; COURVILLE, 2016). Essa capacidade de aprendizado hierárquico tornou possível o avanço em áreas como reconhecimento facial, tradução automática e diagnóstico médico baseado em imagens.

Em aplicações de visão computacional, o aprendizado profundo mostrou-se especialmente eficaz, pois as redes são capazes de identificar padrões invariantes a iluminação, rotação ou escala — algo difícil de alcançar com técnicas manuais. Por exemplo, em tarefas de reconhecimento facial, as primeiras camadas de uma rede tendem a aprender características simples, como contornos e gradientes, enquanto as camadas mais profundas aprendem estruturas globais da face e expressões complexas relacionadas à emoção.

Além disso, o surgimento de técnicas como o *transfer learning* e o *fine-tuning* ampliou a aplicabilidade do aprendizado profundo em domínios com poucos dados. Nesses casos, o modelo é inicializado com pesos pré-treinados em bases extensas, como a ImageNet, e ajustado para o novo conjunto de dados, aproveitando o conhecimento prévio adquirido (PLESTED; GEDEON, 2022). Essa estratégia reduz o tempo de treinamento e melhora o desempenho em tarefas específicas, como a classificação de expressões faciais. Ferramentas modernas de aprendizado profundo, como PyTorch e TensorFlow, tornaram esse processo acessível e eficiente, viabilizando experimentos de alta complexidade em hardware acessível, como GPUs comerciais.

## 2.3 REDES NEURAIIS CONVOLUCIONAIS (CNNs)

As Redes Neurais Convolucionais (CNNs) são uma das arquiteturas mais importantes dentro do aprendizado profundo, especialmente em aplicações de visão computacional (ALBAWI; MOHAMMED; AL-ZAWI, 2017). Inspiradas no funcionamento do córtex

visual humano, as CNNs utilizam camadas convolucionais para extrair automaticamente padrões espaciais de uma imagem, aprendendo representações progressivamente mais complexas a cada camada (RAWAT; WANG, 2017).

De modo geral, uma CNN é composta por:

- **Camadas convolucionais:** responsáveis por detectar padrões locais, como bordas, texturas e formas simples;
- **Camadas de *pooling*:** reduzem a dimensionalidade das representações, preservando informações relevantes e tornando o modelo mais eficiente;
- **Camadas totalmente conectadas (*fully connected*):** realizam a classificação final com base nas características extraídas.

Além disso, funções de ativação não lineares, como a ReLU (*Rectified Linear Unit*), e técnicas de regularização, como o *dropout*, ajudam a melhorar a convergência e reduzir o sobreajuste. O uso de normalização de lotes (*batch normalization*) também é comum para estabilizar o processo de aprendizado (ALBAWI; MOHAMMED; AL-ZAWI, 2017).

Essas propriedades tornaram as CNNs a base para diversas arquiteturas modernas, como AlexNet, VGG, ResNet e EfficientNet, que influenciaram diretamente a evolução de detectores como o YOLO. Nas tarefas de reconhecimento facial, as CNNs são responsáveis por extrair representações robustas de regiões específicas do rosto, permitindo que sutis variações musculares associadas a emoções sejam reconhecidas de forma automática. Dessa maneira, elas funcionam como o núcleo de modelos de detecção e classificação simultâneos, sendo essenciais para o desempenho de sistemas modernos de reconhecimento facial baseados em YOLO.

## 2.4 A ARQUITETURA YOLO (*YOU ONLY LOOK ONCE*)

Entre os vários modelos de detecção de objetos, a arquitetura YOLO se destaca por combinar rapidez e precisão em uma única etapa de processamento. Enquanto modelos anteriores, como R-CNN e Faster R-CNN, realizavam a detecção em dois estágios, primeiro gerando regiões de interesse e depois classificando, o YOLO unifica todo o processo, tratando a detecção como um problema de regressão direta (REDMON et al., 2016).

Nessa abordagem, a imagem é dividida em uma grade  $S \times S$ , e cada célula é responsável por prever as coordenadas das caixas delimitadoras, os níveis de confiança e as probabilidades de classe. Essa estrutura permite que o modelo analise a imagem completa de uma só vez, o que o torna extremamente rápido e adequado para aplicações em tempo real.

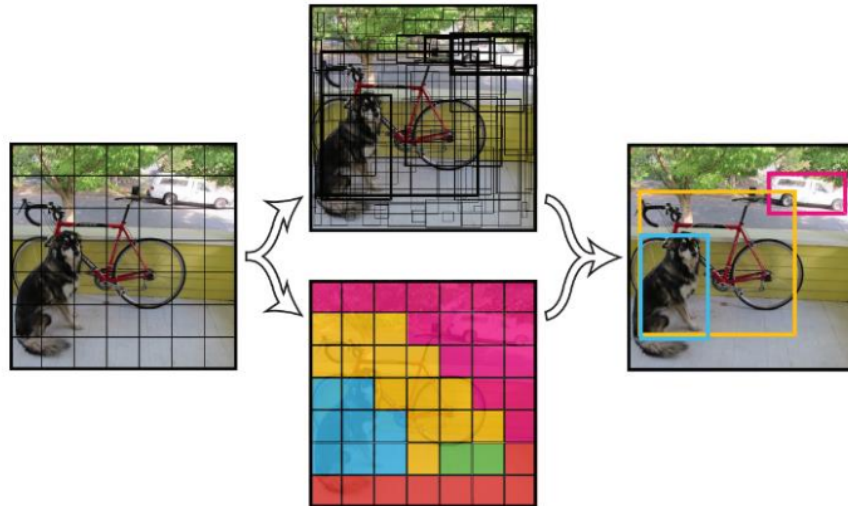


Figura 1 – Ilustração do funcionamento da arquitetura YOLO. O modelo divide a imagem em uma grade  $S \times S$ , onde cada célula prediz  $B$  caixas delimitadoras, níveis de confiança e probabilidades de classe. Figura adaptada de Redmon et al. (2016)(REDMON et al., 2016).

A arquitetura moderna do YOLO é organizada em três blocos principais:

- **Backbone:** extrai características visuais da imagem por meio de camadas convolucionais profundas (como o CSPDarknet53);
- **Neck:** combina características em diferentes escalas, utilizando estruturas como FPN (Feature Pyramid Network) e PANet, que permitem a detecção de objetos pequenos, médios e grandes;
- **Head:** gera as predições finais — coordenadas, confiança e classes.

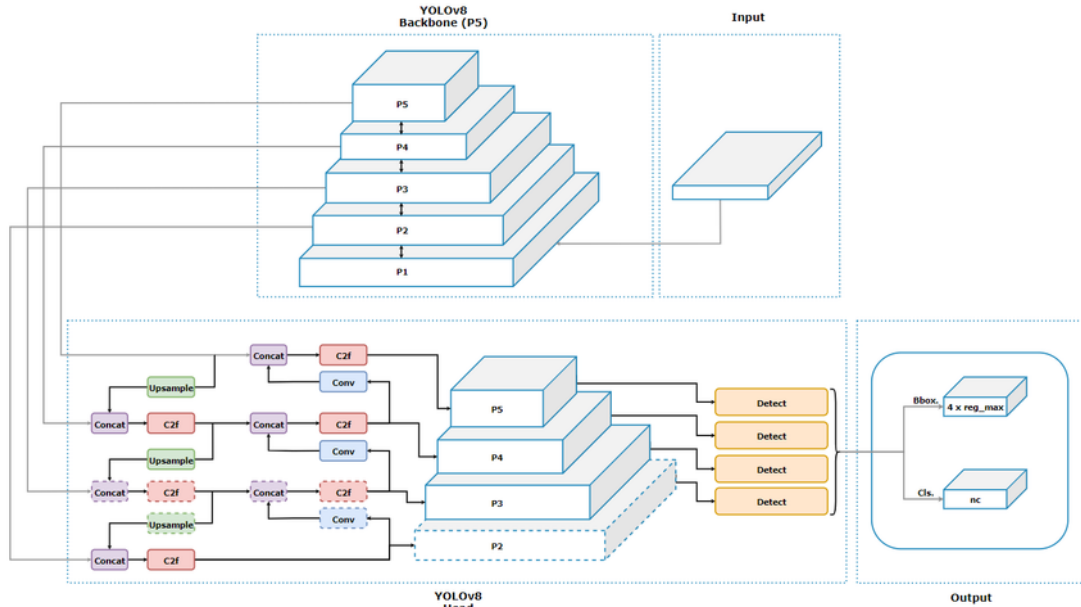


Figura 2 – Arquitetura detalhada do YOLOv8, destacando o *backbone* com múltiplas camadas convolucionais hierárquicas, a *Feature Pyramid Network* (FPN) que integra informações em diferentes escalas e a *head* de detecção responsável pelas predições finais. Figura adaptada de Jegham et al. (2025) (JEGHAM et al., 2025)

Apesar de seu alto desempenho, o YOLO apresenta limitações. Redmon et al. (2016) destacam que a restrição de uma célula prever apenas algumas caixas e classes pode dificultar a detecção de objetos pequenos ou muito próximos. Além disso, as camadas de grande redução espacial podem levar à perda de detalhes finos, reduzindo a precisão em regiões pequenas.

Versões recentes, como YOLOv8, incorporaram aprimoramentos estruturais que mitigam essas limitações, incluindo detecção em múltiplas escalas, blocos de atenção (como SE e CBAM) e camadas de melhor conectividade, tornando o modelo mais robusto e preciso (JEGHAM et al., 2025).

Comparativamente, modelos como Faster R-CNN e SSD (*Single Shot Multibox Detector*) também alcançam alta precisão, porém com maior custo computacional e menor velocidade de inferência. O YOLO, por outro lado, equilibra desempenho e eficiência, sendo mais adequado para aplicações em tempo real, como vigilância, interação humano-computador e reconhecimento de expressões faciais.

Neste trabalho, o YOLO é utilizado para detectar a face e, simultaneamente, classificar a expressão facial correspondente. Essa abordagem oferece um equilíbrio entre desempenho e eficiência, sendo ideal para aplicações em tempo real em ambientes não controlados.

### 2.4.1 Evolução das versões YOLO utilizadas (v8, v10, v11 e v12)

A evolução recente da família YOLO tem sido marcada por revisões arquiteturais significativas, muitas das quais foram sistematizadas por Jegham et al. (2025). As versões avaliadas neste trabalho — YOLOv8, YOLOv10, YOLOv11 e YOLOv12 — representam diferentes estágios dessa progressão, cada uma introduzindo mecanismos próprios para melhorar precisão, eficiência e estabilidade.

O YOLOv8 inaugura uma linha de modelos mais leves e flexíveis dentro da Ultralytics. Essa versão substitui os blocos tradicionais do CSPDarknet pelo módulo C2f, que aumenta a reutilização de características e melhora o fluxo de informação no *backbone*. Além disso, adota uma *head* de detecção totalmente *anchor-free*, tendência iniciada no YOLOv3u, simplificando o processo de predição e reduzindo operações redundantes. Essas mudanças tornaram o YOLOv8 um modelo equilibrado, combinando boa precisão com velocidade elevada (JOCHER; CHAURASIA; QIU, 2023).

O YOLOv10, desenvolvido por pesquisadores da Universidade de Tsinghua, introduz um conjunto de inovações centradas na eficiência e na eliminação do pós-processamento tradicional. Seu principal avanço é o treinamento sem *Non-Maximum Suppression* (NMS), viabilizado pela estratégia de atribuição dupla composta por uma *one-to-many head* durante o treinamento e uma *one-to-one head* na inferência. Essa abordagem reduz a latência, torna o modelo mais determinístico e melhora o limite de velocidade da arquitetura. A versão também incorpora um *backbone* CSPNet aprimorado, convoluções de kernel grande, módulos de autoatenção parcial e um mecanismo de *downsampling* espacial-canal desacoplado, resultando em maior eficiência sem aumento expressivo de GFLOPs (WANG et al., 2024).

O YOLOv11 aprofunda algumas das mudanças estruturais iniciadas no YOLOv8, introduzindo o módulo C2PSA (*Cross-Stage Partial with Self-Attention*), que combina caminhos paralelos de convolução com mecanismos de autoatenção. Além disso, substitui o bloco C2f pelo C3k2, que utiliza duas convoluções menores para melhorar a eficiência e reduzir redundâncias. Essas alterações ampliam a capacidade do modelo de capturar dependências contextuais, especialmente úteis para objetos pequenos ou parcialmente ocluídos, mantendo o custo computacional moderado (JOCHER; QIU, 2024).

Por fim, o YOLOv12 representa o estágio evolutivo mais avançado entre as versões consideradas. Essa arquitetura adota um design centrado em atenção, introduzindo o módulo *Area Attention* (A2), que amplia o campo receptivo sem aumento expressivo de custo computacional. Incorpora também os blocos *Residual Efficient Layer Aggregation Networks* (R-ELAN), que otimizam a agregação de características e aumentam a estabilidade do treinamento. Outros refinamentos incluem o uso de *FlashAttention*, a redução da razão MLP e a remoção de *positional encodings*, contribuindo para um modelo mais

rápido e eficiente, mantendo desempenho elevado em IoUs mais rigorosos (TIAN; YE; DOERMANN, 2025).

Em síntese, as quatro versões apresentam filosofias de projeto distintas: enquanto o YOLOv8 privilegia simplicidade e equilíbrio, o YOLOv10 busca eficiência e latência mínima, o YOLOv11 introduz mecanismos híbridos de convolução e atenção, e o YOLOv12 aprofunda o uso de blocos de atenção para maximizar a precisão em tarefas desafiadoras. Essas diferenças arquiteturais ajudam a explicar os comportamentos observados nos experimentos deste trabalho.

## 2.5 INFERÊNCIA EM TEMPO REAL EM APLICAÇÕES WEB

A aplicação de modelos de aprendizado profundo em sistemas baseados em vídeo exige que a inferência seja realizada com baixa latência, de modo que os resultados sejam apresentados ao usuário em tempo próximo ao da captura da imagem. Em tarefas como o reconhecimento de expressões faciais por *webcam*, esse requisito torna-se especialmente importante, pois atrasos elevados podem comprometer a percepção de continuidade da aplicação e reduzir sua utilidade prática. Nesse contexto, aplicações de análise de vídeo em tempo real precisam equilibrar desempenho preditivo, custo computacional, taxa de processamento e capacidade de resposta do sistema (LIU; QI; BANERJEE, 2018; SASSU; SAENZ-COGOLLO; AGELLI, 2021).

Liu, Qi e Banerjee (LIU; QI; BANERJEE, 2018) destacam que a execução de inferência baseada em redes neurais profundas sobre vídeo em tempo real pode gerar desafios relacionados à latência, consumo de largura de banda, confiabilidade e privacidade, especialmente quando o processamento depende exclusivamente de servidores remotos. Embora o presente trabalho não proponha uma arquitetura de computação de borda, esses desafios também são relevantes para aplicações web interativas, nas quais quadros capturados pela câmera precisam ser enviados para processamento e os resultados devem retornar rapidamente à interface do usuário.

Em arquiteturas web, esse tipo de aplicação costuma ser organizado em uma estrutura cliente-servidor. O cliente, executado no navegador, é responsável pela interação com o usuário, pela captura da imagem da câmera e pela apresentação visual dos resultados. O servidor, por sua vez, concentra as etapas de maior custo computacional, como o recebimento dos quadros, o pré-processamento das imagens, o carregamento do modelo de aprendizado profundo e a execução da inferência. Essa separação permite que a interface permaneça responsiva, enquanto o processamento do modelo ocorre em um ambiente mais adequado para a execução da rede neural.

Sassu, Saenz-Cogollo e Agelli (SASSU; SAENZ-COGOLLO; AGELLI, 2021) também ressaltam que sistemas de análise de vídeo em tempo real baseados em aprendizado profundo exigem a coordenação de diferentes módulos de software, incluindo captura de

vídeo, processamento, extração de informações, monitoramento de desempenho e disponibilização dos resultados para aplicações clientes. Os autores apresentam uma arquitetura com suporte a múltiplos fluxos de vídeo, pipelines de processamento e interfaces web baseadas em HTTP e WebRTC, evidenciando a importância de mecanismos padronizados para consumir tanto os dados extraídos quanto o fluxo de vídeo em aplicações executadas em navegadores.

Nesse contexto, o WebRTC representa uma alternativa relevante para aplicações web que exigem comunicação em tempo real. Segundo Mahmoud e Abozariba (MAHMOUD; ABOZARIBA, 2025), essa tecnologia foi projetada para permitir comunicação em tempo real por meio de navegadores web e dispositivos móveis, oferecendo baixa latência, suporte a comunicação ponto a ponto e recursos de segurança. Sua arquitetura envolve mecanismos como sinalização, *PeerConnection*, ICE, STUN e TURN, que permitem estabelecer conexões entre clientes mesmo em cenários com restrições de rede. Dessa forma, o WebRTC pode ser utilizado em aplicações que demandam transmissão contínua de mídia ou dados, como sistemas interativos baseados em vídeo.

Outro aspecto importante em aplicações de inferência em tempo real é a forma como os resultados são disponibilizados à interface. Em sistemas interativos, não basta processar uma imagem isolada; é necessário manter um fluxo contínuo de captura, inferência e atualização visual. Para isso, podem ser utilizados protocolos voltados à transmissão em tempo real, como o WebRTC, e canais de comunicação persistentes, como WebSocket, permitindo que informações como classe prevista, confiança da predição e coordenadas da caixa delimitadora sejam enviadas continuamente ao cliente. Essa organização favorece a construção de interfaces capazes de exibir os resultados sobre o vídeo capturado, mantendo a experiência de uso próxima ao tempo real.

Além da inferência, aplicações interativas podem incorporar mecanismos de feedback do usuário, permitindo registrar se uma predição foi considerada correta, incorreta ou incerta. Esses registros não substituem a avaliação quantitativa realizada com conjuntos de teste previamente anotados, mas funcionam como uma fonte complementar de análise em cenários reais de uso. Quando organizados em dashboards, esses dados permitem visualizar padrões de acerto, erro e confusão entre classes, contribuindo para identificar limitações do modelo e possíveis exemplos difíceis para reavaliação futura.

### 3 TRABALHOS RELACIONADOS

#### 3.1 YOLO

A arquitetura YOLO (*You Only Look Once*), apresentada inicialmente por Redmon et al. (2016) (REDMON et al., 2016), consolidou-se como uma das mais importantes famílias de modelos para detecção de objetos em visão computacional. Seu diferencial é realizar a detecção em uma única passada pela rede, permitindo inferência em alta velocidade sem perdas significativas de precisão. Essa filosofia influenciou diretamente a evolução de diversas versões subsequentes, como YOLOv3, YOLOv5, YOLOv7, YOLOv8 e, mais recentemente, YOLOv11 e YOLOv12, destacadas em (JEGHAM et al., 2025). Ao longo dessas gerações, melhorias estruturais foram introduzidas, incluindo módulos de atenção, OTAs otimizados, arquiteturas mais leves e uso de novas funções de perda.

Além de seu impacto teórico, o YOLO tem sido amplamente aplicado em diferentes domínios. Na agricultura, modelos das famílias YOLOv5 e YOLOv8 têm sido empregados para classificação de pragas, detecção de frutos e monitoramento automatizado de lavouras, mesmo sob condições adversas de iluminação e movimento, conforme apontado por Kanna et al. (2024) (KANNA et al., 2024). Esses sistemas demonstram que o YOLO é capaz de oferecer uma boa relação entre custo computacional e precisão, viabilizando aplicações embarcadas em drones e sensores agrícolas.

No setor industrial, os avanços do YOLO têm permitido inspeções automatizadas com alto nível de confiabilidade. Ali et al. (2024) (ALI; NAEEM et al., 2024) ressaltam que versões recentes — especialmente YOLOv7 e YOLOv8 — conseguem identificar falhas minúsculas em produtos, contribuindo para processos de garantia de qualidade em linhas de montagem. A detecção em tempo real é um diferencial essencial nesses ambientes, pois reduz perdas e permite intervenções operacionais rápidas. Além disso, a literatura discute como melhorias como decoupled heads e otimizadores eficientes aumentaram a robustez do modelo para cenários industriais com ruído visual.

Outra área que se beneficia significativamente do YOLO é a medicina. De acordo com Pinto-Coelho (2023) (PINTO-COELHO, 2023), áreas como radiologia, dermatologia e análises histopatológicas têm adotado o YOLO para identificação de tumores, lesões e anomalias diversas. Mesmo em tarefas altamente sensíveis, como detecção de câncer, o modelo tem alcançado desempenho comparável ao de especialistas humanos, assegurando rapidez e baixo custo de processamento. Esses resultados evidenciam o potencial do YOLO para apoiar diagnósticos e melhorar o fluxo clínico.

Aplicações em segurança também se destacam amplamente. O trabalho YOLO-IHD, proposto por Kucukayan et al. (2024) (KUCUKAYAN; YAVUZ et al., 2024), demonstra como adaptações do YOLO possibilitam reconhecimento humano eficiente em drones indoor, garantindo estabilidade mesmo em ambientes fechados e com obstáculos. A robustez

a movimentação, sombra e oclusões torna a arquitetura ideal para vigilância inteligente, rastreamento de indivíduos e detecção de comportamentos suspeitos.

Finalmente, o YOLO vem ganhando grande relevância em tarefas relacionadas a aspectos humanos, incluindo reconhecimento de expressões faciais, emoções e microexpressões. Pesquisas como as de Zhong et al. (2023) (ZHONG et al., 2023), Alshammari (2024) (ALSHAMMARI; ALSHAMMARI, 2024) e Li et al. (2024) (LI; ZHANG; CHEN, 2024) demonstram que versões como YOLOv5 e YOLOv8 apresentam excelente equilíbrio entre precisão e desempenho em tempo real, mesmo sob variações de pose, iluminação e ruído. Esses estudos se alinham diretamente ao objetivo desta monografia, reforçando o YOLO como escolha adequada para sistemas em tempo real de reconhecimento de expressões faciais.

### 3.2 RECONHECIMENTO DE EXPRESSÕES FACIAIS (REF)

O reconhecimento de expressões faciais (REF) tem evoluído rapidamente, impulsionado pela expansão das redes neurais profundas e pela disponibilidade de grandes bases de dados em ambientes não controlados. Pesquisas tradicionais revisadas por Li e Deng (2018) (LI; DENG, 2018) indicam que métodos clássicos baseados em descritores manuais — como Local Binary Patterns (LBP), HOG e Eigenfaces — foram gradualmente superados por arquiteturas baseadas em CNNs, que extraem automaticamente padrões complexos relacionados às expressões humanas. Esse salto tecnológico permitiu maior robustez a variações de iluminação, oclusões e diferenças culturais nas expressões.

Além das CNNs tradicionais, técnicas de atenção e fusão multimodal têm ganhado destaque na literatura recente. Liao et al. (2023) (LIAO et al., 2023) exploram a combinação entre mecanismos de atenção e texturas LBP, demonstrando que a fusão de características locais e globais aumenta a precisão em ambientes desafiadores. Da mesma forma, estudos como o de Taware e Thakare (2023) (TAWARE; THAKARE, 2023) apontam que abordagens multimodais baseadas em voz, postura corporal ou contexto ambiental podem complementar o reconhecimento facial em cenários reais.

Outro eixo fundamental do avanço do REF é a disponibilidade de bases de dados amplas e diversificadas. AffectNet (MOLLAHOSSEINI; HASANI; MAHOOR, 2017), por exemplo, tornou-se um dos marcos da área ao oferecer milhões de imagens rotuladas com expressões variadas e níveis de valência e excitação. Esse tipo de dataset foi essencial para treinar modelos mais robustos e para reduzir o viés cultural presente em bases anteriores mais limitadas. Trabalhos posteriores vêm utilizando essas bases para explorar classificações mais complexas e detectar emoções sutis.

O YOLO, tradicionalmente aplicado à detecção de objetos, também passou a ser incorporado como backbone em sistemas de REF, especialmente em aplicações que exigem processamento em tempo real. Zhong et al. (2023) (ZHONG et al., 2023) demonstram um

sistema baseado em YOLOv5 com módulos de atenção que permite identificar expressões de professores em salas de aula, favorecendo análise comportamental e suporte pedagógico. Alshammari (2024) (ALSHAMMARI; ALSHAMMARI, 2024), por sua vez, avalia o YOLOv8 e mostra que a arquitetura mantém alto desempenho mesmo com expressões de baixa intensidade.

Além disso, adaptações mais recentes incluem o uso de vídeo e modelos híbridos. O estudo EPJ (2025) (SAREEN; SEEJA, 2025) propõe a fusão entre YOLO e Vision Transformers (ViT) para reconhecer emoções em sequências temporais, conseguindo capturar microexpressões e variações dinâmicas faciais. Em complemento, Li et al. (2024) (LI; ZHANG; CHEN, 2024) apresentam uma versão otimizada do YOLOv8 voltada ao REF, indicando que aprimoramentos como convoluções reestruturadas e módulos de atenção expandem a capacidade do modelo para tarefas mais sensíveis.

Outra linha de pesquisa relevante envolve aplicações reais de reconhecimento de expressões faciais em interações sociais complexas, como entrevistas de emprego. Naim et al. (2015) (NAIM et al., 2015) apresentam um dos estudos mais abrangentes sobre análise multimodal de desempenho em entrevistas, utilizando um conjunto de 138 vídeos gravados em cenários realistas. O trabalho combina extração automática de expressões faciais, incluindo sorrisos, movimentos de sobrancelhas, abertura dos olhos e distâncias labiais, com informações prosódicas e linguísticas para prever traços sociais como engajamento, entusiasmo e simpatia. Os autores demonstram que expressões faciais, mesmo capturadas em ambientes não controlados, são determinantes para a percepção de competências sociais, alcançando correlações superiores a 0.75 na predição de traços como excitação e engajamento. Esse estudo reforça que o REF possui papel central em aplicações práticas de avaliação comportamental, evidenciando a necessidade de sistemas capazes de operar em tempo real e sob condições naturais, o mesmo tipo de robustez buscado neste trabalho ao empregar arquiteturas YOLO.

Dessa forma, a literatura indica que sistemas de REF baseados em YOLO apresentam vantagens relevantes para aplicações práticas, especialmente pela rapidez, portabilidade e bom custo computacional. Esses fatores tornam a arquitetura atrativa para sistemas de reconhecimento de expressões faciais em tempo real, servindo como base para a etapa aplicada desenvolvida neste trabalho.

### 3.3 APLICAÇÕES WEB PARA ANÁLISE DE VÍDEO EM TEMPO REAL

Além dos avanços em modelos de reconhecimento de expressões faciais, outro aspecto relevante para este trabalho é a integração de modelos de aprendizado profundo a aplicações capazes de operar sobre fluxos de vídeo em tempo real. Sistemas desse tipo precisam lidar não apenas com a acurácia dos modelos, mas também com requisitos de latência, transmissão de dados, organização dos módulos de processamento e apresentação

dos resultados ao usuário.

Liu, Qi e Banerjee (LIU; QI; BANERJEE, 2018) apresentam o EdgeEye, um framework voltado à análise inteligente de vídeo em ambientes de borda. Os autores destacam que a execução de inferência baseada em redes neurais profundas sobre vídeo pode gerar desafios relacionados ao consumo de largura de banda, latência, confiabilidade e privacidade, especialmente quando o processamento depende de servidores remotos. Embora o foco do trabalho esteja em computação de borda, sua contribuição é relevante por evidenciar a necessidade de aproximar o processamento da fonte de dados e de disponibilizar os resultados da análise com baixa latência.

Sassu, Saenz-Cogollo e Agelli (SASSU; SAENZ-COGOLLO; AGELLI, 2021) propõem o Deep-Framework, uma arquitetura distribuída e escalável para análise de fluxos de vídeo baseada em aprendizado profundo. O sistema organiza o processamento em pipelines, permitindo a integração de modelos de detecção e extração de características, além de disponibilizar interfaces baseadas em HTTP e WebRTC para consumo dos dados em aplicações clientes. Esse tipo de abordagem reforça a importância de arquiteturas modulares para aplicações de vídeo em tempo real, nas quais diferentes componentes precisam atuar de forma coordenada, desde a captura até a visualização dos resultados.

No contexto de comunicação em aplicações web, Mahmoud e Abozariba (MAHMOUD; ABOZARIBA, 2025) apresentam uma revisão sistemática sobre o WebRTC, destacando sua aplicação em cenários que exigem comunicação em tempo real, baixa latência e suporte à transmissão de mídia ou dados em navegadores. Os autores também discutem desafios associados à conectividade, segurança, interoperabilidade e escalabilidade, aspectos relevantes para sistemas interativos baseados em vídeo.

Esses trabalhos indicam que a construção de aplicações de análise de vídeo em tempo real envolve desafios que vão além do treinamento do modelo. É necessário considerar também a forma como o vídeo é capturado, transmitido, processado e apresentado ao usuário. Nesse sentido, o presente trabalho aproxima-se dessas abordagens ao integrar um modelo de reconhecimento de expressões faciais a um protótipo web funcional. Contudo, diferencia-se por concentrar-se na aplicação de um modelo YOLOv8n previamente selecionado para reconhecimento de expressões faciais, incorporando ainda um mecanismo de feedback manual e um dashboard para análise dos registros coletados durante o uso.

## 4 METODOLOGIA

### 4.1 VISÃO GERAL DA METODOLOGIA

Este trabalho foi desenvolvido em duas etapas complementares. Na primeira, realizou-se uma avaliação comparativa entre diferentes versões da arquitetura YOLO, com o objetivo de identificar o modelo mais adequado para a tarefa de reconhecimento de expressões faciais em tempo real. Na segunda etapa, o modelo selecionado foi incorporado a um protótipo funcional, permitindo sua avaliação funcional em um cenário local de uso mais próximo de uma aplicação prática. Essa organização metodológica possibilitou tanto a comparação controlada entre arquiteturas quanto a análise prática do sistema final.

### 4.2 ETAPA 1 - AVALIAÇÃO COMPARATIVA DAS ARQUITETURAS YOLO

#### 4.2.1 Conjunto de Dados

Para o desenvolvimento deste estudo foi utilizado o conjunto de dados “9 Facial Expressions for YOLO”<sup>1</sup>, disponibilizado publicamente na plataforma Kaggle. Esse dataset foi especificamente preparado para tarefas de detecção e classificação de expressões faciais, contendo mais de 65 mil imagens em condições *in-the-wild*, ou seja, capturadas em ambientes não controlados, com variações de iluminação, pose, oclusões e diversidade demográfica.

As imagens encontram-se nos formatos JPG ou PNG e incluem anotações no padrão YOLO, contendo as coordenadas das caixas delimitadoras e o rótulo associado a uma das nove expressões: *angry* (raiva), *contempt* (desprezo), *disgust* (nojo), *fear* (medo), *happy* (feliz), *natural* (neutra), *sad* (triste), *sleepy* (sonolento) e *surprised* (surpreso). A divisão em conjuntos de treino, validação e teste é fornecida originalmente pelo autor do dataset, eliminando a necessidade de dividir manualmente o material disponível.

No formato disponibilizado, o dataset é composto por 64.866 imagens para treino (94,99%), 1.720 imagens para validação (2,52%) e 1.700 imagens para teste (2,49%). Embora essas proporções sejam menos convencionais quando comparadas às divisões clássicas (como 70/15/15 ou 80/10/10), a escolha por manter essa estrutura visa preservar a consistência com o protocolo do dataset original. Além disso, esse número de amostras nos conjuntos de validação e teste é suficiente para permitir uma avaliação estável e confiável dos modelos, sem prejuízo das métricas de desempenho.

---

<sup>1</sup>Projeto disponível em: <https://www.kaggle.com/datasets/aklimarimi/8-facial-expressions-for-yolo>.

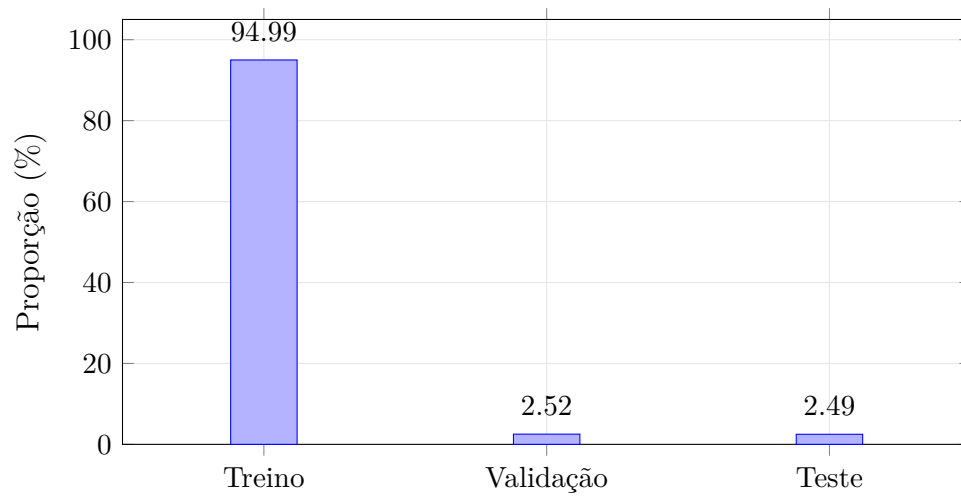


Figura 3 – Proporção dos conjuntos de treino, validação e teste no dataset.

O conjunto foi construído a partir de múltiplas bases hospedadas no Roboflow Universe, previamente filtradas para remoção de duplicatas e imagens inválidas. Essa diversidade de fontes contribui para um grau elevado de variabilidade, favorecendo a avaliação dos modelos em cenários realistas. A Figura 4 apresenta exemplos representativos das imagens incluídas no dataset.

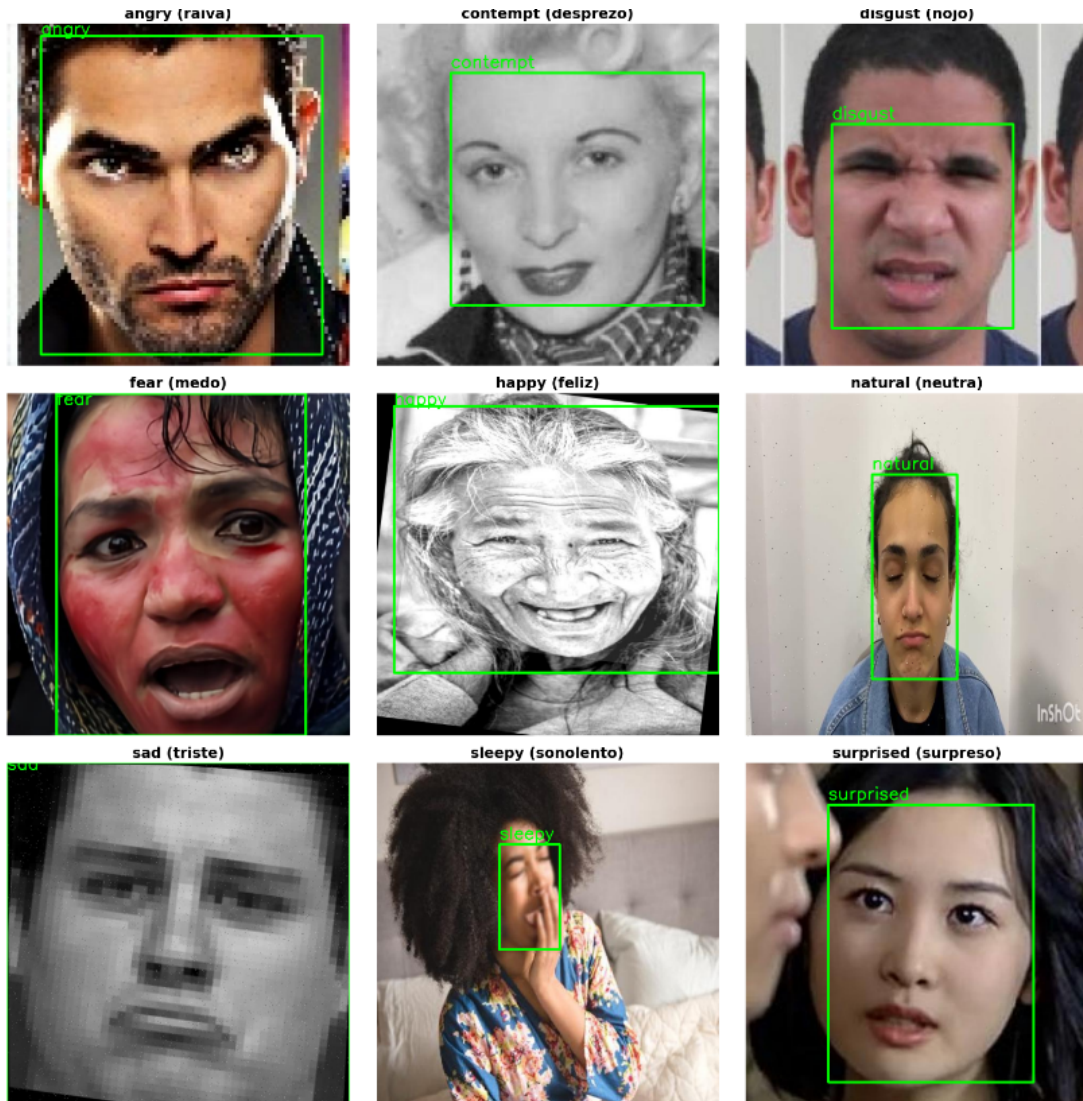


Figura 4 – Amostras do conjunto de dados “9 Facial Expressions for YOLO”, ilustrando variedade de expressões e condições de captura.

Além das características visuais, também foi analisada a distribuição das anotações por classe, uma vez que o desbalanceamento entre categorias pode influenciar significativamente o desempenho dos modelos. A Tabela 1 apresenta o número total de instâncias por expressão facial no conjunto de dados.

Observa-se uma forte assimetria: classes como *happy (feliz)*, *angry (raiva)* e *sad (triste)* possuem mais de 10 mil instâncias, enquanto categorias de menor intensidade, como *contempt (desprezo)* e *sleepy (sonolento)*, apresentam menos de 3 mil e 1,1 mil exemplos, respectivamente. Esse desbalanceamento tende a favorecer as classes majoritárias durante o treinamento, contribuindo para taxas mais altas de confusão nas categorias minoritárias — um comportamento que se reflete nas matrizes de confusão e nas métricas por classe discutidas na seção de Resultados.

Tabela 1 – Distribuição de anotações por classe no dataset.

| Classe | Expressão                   | Total de Anotações |
|--------|-----------------------------|--------------------|
| 0      | <i>angry</i> (raiva)        | 11195              |
| 1      | <i>contempt</i> (desprezo)  | 2543               |
| 2      | <i>disgust</i> (nojo)       | 4296               |
| 3      | <i>fear</i> (medo)          | 5358               |
| 4      | <i>happy</i> (feliz)        | 13831              |
| 5      | <i>natural</i> (neutra)     | 5663               |
| 6      | <i>sad</i> (triste)         | 11969              |
| 7      | <i>sleepy</i> (sonolento)   | 1051               |
| 8      | <i>surprised</i> (surpreso) | 8960               |

#### 4.2.2 Escolha do dataset

A escolha do conjunto de dados “*9 Facial Expressions for YOLO*” deve-se a um conjunto de características que o tornam adequado aos objetivos deste trabalho. Primeiramente, o *dataset* foi construído especificamente para modelos da família YOLO, fornecendo anotações no formato nativo da arquitetura, com as imagens e as suas respectivas coordenadas das caixas delimitadoras, o que elimina a necessidade de conversões. Isso garante maior consistência com o protocolo experimental e favorece uma comparação justa entre as diferentes versões avaliadas.

Outro motivo relevante é a diversidade visual do conjunto, composto por mais de 65 mil imagens coletadas em condições *in-the-wild*, com grande variação de iluminação, ângulos, expressões de baixa intensidade e oclusões parciais. Essas características aproximam o cenário de teste das condições reais de uso, tornando a avaliação mais representativa e alinhada ao propósito do trabalho, que envolve desempenho em ambientes não controlados.

Além disso, o dataset contempla nove categorias de expressões faciais, incluindo expressões pouco exploradas em bases tradicionais, como *contempt* e *sleepy*. A presença dessas classes amplia o desafio da tarefa e permite avaliar a capacidade de generalização das arquiteturas YOLO não apenas em expressões prototípicas (como *happy* ou *fear*), mas também em expressões mais sutis, diretamente relacionadas à variabilidade do comportamento humano.

Por fim, o dataset apresenta divisão pré-definida entre treino, validação e teste, elaborada pelo próprio autor. Manter essa divisão garante reprodutibilidade, padronização e alinhamento com o uso esperado da base, além de evitar vieses decorrentes de uma divisão manual alternativa.

Dessa forma, a escolha do “*9 Facial Expressions for YOLO*” se dá por sua adequação técnica ao escopo deste trabalho: diversidade realista, proporção adequada, anotação compatível com a arquitetura YOLO e variedade de expressões que permitem uma análise completa do desempenho das versões avaliadas.

#### 4.2.3 Arquiteturas Avaliadas

Foram avaliadas quatro versões recentes da família YOLO: YOLOv8n, YOLOv10n, YOLOv11n e YOLOv12n. Todas as versões pertencem à categoria *nano* (n), caracterizada por modelos compactos, baixo custo computacional e adequação a aplicações em tempo real.

A escolha por comparar variantes *nano* se deve ao objetivo central deste trabalho: analisar o equilíbrio entre precisão e velocidade em cenários nos quais a eficiência é crítica. Além disso, modelos menores reduzem o tempo de treinamento e possibilitam comparação sistemática sob o mesmo orçamento computacional.

As diferenças estruturais entre as versões incluem avanços no backbone, novos blocos convolucionais, aprimoramentos na estratégia de aprendizado e ajustes no balanceamento entre profundidade e largura da rede. Avaliar essas etapas evolutivas permite compreender se as melhorias propostas em cada versão refletem ganhos concretos na tarefa de reconhecimento de expressões faciais.

#### 4.2.4 Protocolo de Treinamento

Todos os modelos foram treinados utilizando a biblioteca Ultralytics YOLO, garantindo consistência entre as execuções. Para assegurar comparação justa, foi adotado o mesmo protocolo experimental para todas as versões:

- **Épocas:** 45
- **Resolução:** `imgsz = 640`
- **Batch size:** 16
- **Otimização:** AdamW
- **Learning rate inicial:** 0,001
- **Learning rate final (lrf):** 0,01
- **Warmup:** 3 épocas
- **Weight decay:** 0,01
- **Semente aleatória (seed):** 42

As mesmas técnicas de *data augmentation* foram aplicadas a todas as arquiteturas, com o objetivo de aumentar a robustez do treinamento, preservando características faciais relevantes. Foram utilizadas:

- rotação limitada (`degrees=10`)
- translação leve (`translate=0.05`)
- escala moderada (`scale=0.10`)
- espelhamento horizontal (`fliplr=0.5`)
- ajustes de matiz, saturação e brilho (`hsv_h=0.015`, `hsv_s=0.5`, `hsv_v=0.3`)
- mosaico com intensidade reduzida (`mosaic=0.10`)
- técnicas potencialmente distorcivas, como *mixup*, *flipud* e *shear*, foram desativadas

Esse conjunto de augmentations foi escolhido com o objetivo ampliar a variabilidade das imagens sem distorcer elementos essenciais das expressões faciais, garantindo que o modelo aprendesse padrões realistas e visualmente consistentes.

#### 4.2.5 Ambiente Computacional

Os experimentos foram conduzidos na plataforma Kaggle Notebooks, utilizando GPU NVIDIA Tesla T4, com 15 GB de memória. O ambiente incluía:

- Python 3.10.14
- PyTorch 2.4.0
- Ultralytics 8.3.x

A execução em ambiente controlado garante que todos os modelos tenham sido treinados sob as mesmas condições de hardware e software, assegurando a comparabilidade dos resultados.

#### 4.2.6 Métricas de Avaliação

A avaliação dos modelos foi realizada considerando parâmetros de acurácia e eficiência. A métrica geométrica básica utilizada foi a Intersection over Union (IoU), definida pela Equação 4.1:

$$\text{IoU} = \frac{\text{área}(B_{\text{pred}} \cap B_{\text{real}})}{\text{área}(B_{\text{pred}} \cup B_{\text{real}})} \quad (4.1)$$

A partir da IoU, são derivadas métricas amplamente utilizadas na literatura:

- **Precisão (*Precision*)** — proporção de predições corretas entre todas as predições positivas.
- **Revocação (*Recall*)** — proporção de objetos corretamente detectados entre todos os objetos reais.
- **AP (*Average Precision*)** — área sob a curva precisão–revocação para cada classe.

- **mAP** — média das APs de todas as classes.
- **mAP50–95** — média das APs calculadas para IoUs de 0,50 a 0,95, em passos de 0,05.

Além das métricas de acurácia, foram avaliados aspectos relacionados ao desempenho computacional:

- **Tempo de inferência** (ms/imagem)
- **FPS** (*frames per second*)
- **GFLOPs**, indicador do custo computacional da arquitetura

Para medir o FPS, cada modelo executou 100 inferências com `batch=1`, utilizando uma imagem fixa do conjunto de validação, desconsiderando o tempo de carregamento do modelo. Esse procedimento fornece uma estimativa consistente da latência real em cenários de uso contínuo.

### 4.3 ETAPA 2 – DESENVOLVIMENTO DO PROTÓTIPO EM TEMPO REAL

Com base nos resultados obtidos na etapa comparativa, especialmente na identificação do YOLOv8n como o modelo com melhor equilíbrio entre precisão, velocidade de inferência e custo computacional, foi desenvolvida a etapa aplicada do trabalho, voltada à implementação prática do sistema em tempo real. Enquanto a etapa comparativa teve como foco a análise controlada entre arquiteturas, esta etapa buscou incorporar o modelo selecionado a uma aplicação funcional, capaz de operar de forma contínua e interativa em um cenário local de uso mais próximo de uma aplicação prática.

O desenvolvimento do protótipo teve como objetivo verificar o comportamento do modelo fora do ambiente experimental, incorporando mecanismos de captura de vídeo, processamento contínuo dos quadros, exibição visual das detecções e coleta de feedback do usuário. Dessa forma, a etapa não se limita à execução da inferência, mas envolve também a integração entre os componentes da aplicação, definindo o fluxo completo entre aquisição da imagem, processamento no backend, apresentação dos resultados ao usuário final e análise posterior dos registros coletados.

#### 4.3.1 Modelo utilizado no protótipo

O modelo utilizado no protótipo foi o YOLOv8n, selecionado a partir da avaliação comparativa realizada na primeira etapa do trabalho. A escolha desse modelo considerou o equilíbrio entre desempenho preditivo, velocidade de inferência e custo computacional, critérios relevantes para aplicações em tempo real. O arquivo de pesos utilizado corresponde ao melhor modelo obtido durante o treinamento dessa arquitetura.

### 4.3.2 Arquitetura geral do protótipo

O protótipo foi estruturado em uma arquitetura cliente-servidor, composta por um módulo de frontend, um módulo de backend, o modelo de inferência YOLOv8n e uma camada de persistência local dos dados. Essa organização foi adotada com o objetivo de separar as responsabilidades da aplicação, mantendo no cliente a interação com o usuário e, no servidor, as rotinas de processamento associadas ao modelo.

O frontend é responsável pela captura do vídeo da webcam diretamente no navegador, pela exibição da cena analisada, pelo desenho das detecções sobre a imagem e pela interação do usuário com o sistema. Nessa camada, o usuário pode iniciar ou encerrar a conexão, acompanhar as predições realizadas pelo modelo, visualizar métricas de execução e registrar feedbacks sobre os resultados apresentados.

O backend é responsável pelo recebimento dos quadros de vídeo, pré-processamento das imagens, execução da inferência com o YOLOv8n e envio dos resultados processados de volta para a interface. Para cada detecção realizada, o sistema retorna a classe prevista, o nível de confiança e as coordenadas da caixa delimitadora correspondente. Essas informações são utilizadas pelo frontend para atualizar a visualização em tempo real.

Além dos módulos de inferência e visualização, o protótipo incorpora uma camada de persistência local para armazenamento dos feedbacks enviados pelo usuário. Esses registros são posteriormente utilizados para alimentar o dashboard de análise, permitindo observar a distribuição de acertos, erros, feedbacks indefinidos, casos sem detecção e confusões entre classes.

A Figura 5 apresenta a arquitetura geral do protótipo, destacando o fluxo de comunicação entre a câmera, o frontend, o backend, o modelo YOLOv8n e o dashboard de análise.

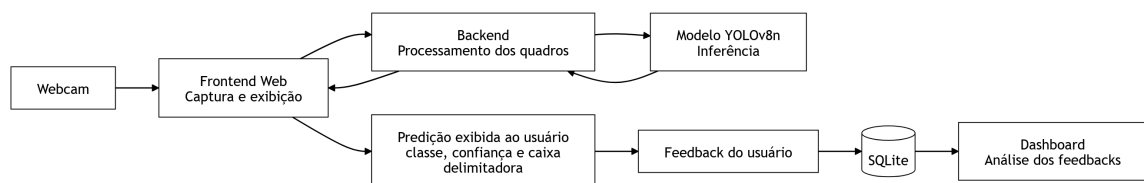


Figura 5 – Arquitetura geral do protótipo desenvolvido para reconhecimento de expressões faciais em tempo real.

### 4.3.3 Tecnologias empregadas

A interface do protótipo foi desenvolvida com React, TypeScript e Vite, permitindo a construção de uma aplicação web modular e responsiva. Para a estilização dos componentes visuais, foi utilizado Tailwind CSS, possibilitando a criação de uma interface leve e adaptável a diferentes tamanhos de tela. A exibição dos gráficos no dashboard foi realizada com a biblioteca Recharts.

Para a captura do vídeo, foram utilizados recursos nativos do navegador, permitindo o acesso à webcam do usuário mediante autorização. A exibição das detecções sobre a imagem foi realizada por meio de um *canvas* sobreposto à área de vídeo, possibilitando o desenho das caixas delimitadoras, rótulos de classe e informações de confiança em tempo real.

No backend, a aplicação foi implementada com FastAPI, adotando uma estrutura adequada para disponibilização de rotas HTTP e comunicação assíncrona. A transmissão do fluxo de vídeo entre cliente e servidor foi realizada por meio de WebRTC, enquanto o retorno das detecções processadas para a interface ocorreu por WebSocket. Essa combinação permite separar o envio contínuo dos quadros da transmissão dos resultados de inferência, favorecendo uma comunicação mais adequada ao contexto de tempo real.

O processamento dos quadros foi realizado com o apoio das bibliotecas OpenCV e NumPy, utilizadas para conversão, manipulação e preparação das imagens antes da inferência. A execução do modelo foi feita por meio da biblioteca Ultralytics YOLO, responsável pelo carregamento do arquivo de pesos `best.pt` e pela realização das predições. O PyTorch foi utilizado como base de execução do modelo, permitindo o uso do dispositivo computacional disponível no ambiente de execução.

Para armazenamento dos registros de feedback, foi utilizado SQLite, uma vez que o protótipo não exige, nesta etapa, uma infraestrutura de banco de dados distribuída ou multiusuário. Essa escolha é suficiente para armazenar localmente as avaliações realizadas pelo usuário e alimentar as consultas utilizadas no dashboard.

#### 4.3.4 Fluxo de inferência em tempo real

O funcionamento do protótipo ocorre em etapas sequenciais. Inicialmente, o usuário acessa a interface web e autoriza o uso da webcam. Em seguida, o vídeo capturado é exibido localmente no navegador, permitindo a visualização contínua da cena que será analisada pelo sistema.

Após a ativação da conexão entre cliente e servidor, o fluxo de vídeo é transmitido ao backend por meio de WebRTC. No servidor, os quadros recebidos são convertidos para um formato compatível com as bibliotecas de visão computacional e submetidos a uma etapa de pré-processamento. Essa etapa inclui a adequação da imagem à entrada esperada pelo modelo, preservando as informações necessárias para a detecção facial e classificação da expressão.

Na sequência, o YOLOv8n realiza a inferência sobre os quadros processados, retornando as detecções encontradas. Para cada detecção, são obtidas a classe prevista, a confiança associada e as coordenadas da caixa delimitadora correspondente. Após essa etapa, as coordenadas são ajustadas para a escala de exibição do vídeo e organizadas em

uma estrutura apropriada para transmissão ao frontend.

Por fim, a interface recebe as detecções por WebSocket e atualiza continuamente a exibição do vídeo com as caixas delimitadoras, os rótulos de classe e informações complementares, como confiança da predição, taxa de quadros por segundo, latência estimada e dispositivo de execução. Esse fluxo permite que o sistema opere de forma contínua, aproximando o comportamento da aplicação de um cenário interativo de uso em tempo real.

#### 4.3.5 Mecanismo de feedback do usuário

Além da inferência em tempo real, o protótipo incorpora um mecanismo de feedback manual, por meio do qual o usuário pode avaliar as predições realizadas pelo modelo. Essa funcionalidade foi adicionada com o objetivo de permitir uma análise prática do comportamento do sistema durante o uso, registrando não apenas as predições corretas, mas também os erros, incertezas e casos em que nenhuma detecção útil foi produzida.

Quando há uma detecção ativa, o usuário pode selecionar a opção de avaliação da predição. Nesse momento, o sistema apresenta um modal de feedback contendo a classe prevista pelo modelo e a confiança associada. O usuário pode então indicar se a predição está correta, incorreta ou se não sabe avaliar o resultado. Nos casos em que a predição é marcada como incorreta, a interface permite selecionar a classe correta entre as nove categorias utilizadas no treinamento: *angry*, *contempt*, *disgust*, *fear*, *happy*, *natural*, *sad*, *sleepy* e *surprised*.

O protótipo também permite registrar situações em que o modelo não detecta nenhuma face ou expressão de forma útil. Nesses casos, o usuário pode utilizar a opção “Não detectou”, informando, quando possível, qual expressão estava tentando realizar. Esse tipo de registro é tratado separadamente dos feedbacks indefinidos, pois representa uma falha de detecção ou ausência de resposta útil do modelo, e não apenas uma incerteza do usuário em relação à predição apresentada.

Os feedbacks coletados são armazenados localmente e incluem informações como classe prevista, confiança, tipo de feedback e, quando aplicável, classe corrigida ou esperada. Esses dados formam a base para a análise posterior apresentada no dashboard, permitindo identificar padrões de erro, classes mais confundidas e situações em que o sistema não retornou detecções.

#### 4.3.6 Dashboard de análise dos feedbacks

Para apoiar a interpretação dos registros coletados durante o uso do protótipo, foi desenvolvido um dashboard de análise dos feedbacks. Essa interface consolida as informações armazenadas pelo sistema e apresenta indicadores quantitativos sobre o comportamento

do modelo em uso interativo.

O dashboard exibe o total de predições avaliadas, a quantidade e o percentual de predições corretas, incorretas, feedbacks indefinidos e casos sem detecção. Essas informações permitem organizar os registros coletados durante os testes do protótipo, complementando as métricas obtidas na etapa experimental anterior.

Além dos indicadores resumidos, o dashboard apresenta gráficos para análise da distribuição dos feedbacks e dos erros por classe prevista. A distribuição de feedbacks permite observar a proporção entre acertos, erros, incertezas e ausência de detecção, enquanto a análise de erros por classe prevista evidencia quais categorias foram mais frequentemente associadas a predições incorretas.

Outro recurso incorporado ao dashboard é a matriz de confusão baseada no feedback do usuário. Diferentemente da matriz de confusão construída a partir do conjunto de teste do dataset, essa matriz é formada a partir das avaliações realizadas durante o uso do protótipo. Assim, a diagonal principal representa predições confirmadas como corretas pelo usuário, enquanto valores fora da diagonal indicam confusões entre a classe prevista pelo modelo e a classe corrigida pelo usuário. Casos de feedback indefinido e sem detecção não são incluídos nessa matriz, pois não fornecem simultaneamente uma classe prevista e uma classe corrigida.

O dashboard também pode apresentar registros recentes, contendo informações como data e hora, classe prevista, confiança, tipo de feedback e classe corrigida ou esperada. Esse recurso permite verificar individualmente as amostras coletadas e compreender a origem dos dados agregados nos gráficos. Adicionalmente, a exportação dos registros em formato CSV possibilita a análise posterior em ferramentas externas.

#### 4.3.7 Persistência dos dados e registros coletados

Os dados produzidos durante o uso do protótipo são armazenados localmente, com a finalidade de registrar os feedbacks fornecidos pelo usuário e disponibilizar essas informações para consultas posteriores no dashboard. Cada registro contém os principais metadados associados à avaliação, como classe prevista, confiança da predição, tipo de feedback, classe corrigida ou esperada, quando aplicável, data e hora do registro.

Além dos metadados utilizados no dashboard, o sistema salva a imagem correspondente ao quadro avaliado quando a predição é marcada como incorreta. Esse armazenamento tem como finalidade permitir análise posterior dos erros do modelo e formar um conjunto de exemplos difíceis, que pode apoiar estudos futuros, como reavaliação ou possível ajuste fino da rede. O protótipo não realiza gravação contínua de vídeo nem salva automaticamente todas as imagens capturadas pela webcam.

Para cada imagem salva em casos de feedback incorreto, também é gerado um arquivo

de rótulo no formato YOLO, contendo a classe corrigida e as coordenadas normalizadas da caixa delimitadora. Esse formato segue a estrutura utilizada no treinamento dos modelos, composta pelo identificador da classe e pelos valores normalizados de centro da caixa, largura e altura. Dessa forma, as amostras coletadas podem ser organizadas de maneira compatível com o formato esperado pela arquitetura YOLO.

Adicionalmente, o sistema registra metadados em arquivo `metadata.jsonl`, incluindo informações como identificador da amostra, caminhos da imagem e do rótulo, classe prevista, classe corrigida, confiança da predição, dimensões do quadro, coordenadas da caixa delimitadora, indicadores de validação da imagem e marcações relacionadas à elegibilidade da amostra para treinamento. Esses metadados permitem rastrear a origem dos exemplos coletados e filtrar amostras inadequadas, como imagens com baixa nitidez, problemas de iluminação ou possíveis duplicações.

A adoção de uma base local nesta etapa tem como objetivo validar o funcionamento da aplicação e analisar o comportamento do modelo em um ambiente de uso controlado pelo pesquisador. Não foram implementados, nesta fase, mecanismos de autenticação, múltiplos usuários ou sincronização em nuvem, pois tais recursos não são necessários para os objetivos centrais do trabalho.

Os registros coletados permitem observar, de forma complementar às métricas tradicionais de avaliação, como o modelo se comporta durante a execução em tempo real. Dessa forma, torna-se possível identificar classes com maior tendência de erro, situações em que o sistema não produz detecções úteis e exemplos que podem ser considerados difíceis para o modelo.

#### 4.3.8 Aspectos éticos e privacidade

Como o protótipo desenvolvido utiliza imagens faciais capturadas por webcam, foram considerados cuidados relacionados à privacidade e ao uso dos dados processados. Durante a execução da aplicação, os quadros de vídeo são utilizados para a realização da inferência pelo modelo YOLOv8n. O sistema não armazena automaticamente todas as imagens capturadas, nem grava o fluxo contínuo de vídeo da webcam ou sessões completas de uso.

O armazenamento de imagens ocorre apenas em situações específicas associadas ao mecanismo de feedback. Quando o usuário marca uma predição como incorreta, o backend salva a imagem correspondente ao quadro avaliado, juntamente com o arquivo de rótulo no formato YOLO e os metadados associados à amostra. Esse procedimento tem como objetivo registrar exemplos difíceis ou casos de erro do modelo, permitindo análise posterior e possível formação de um conjunto complementar de amostras para estudos futuros.

A base local e os arquivos gerados armazenam informações como classe prevista,

confiança da predição, classe corrigida ou esperada, coordenadas da caixa delimitadora, caminhos dos arquivos e indicadores de validação da amostra. Esses dados são utilizados para apoiar a análise do comportamento do modelo e não correspondem à gravação contínua da webcam.

Os testes funcionais foram realizados em ambiente local, pelo próprio autor do trabalho, com finalidade acadêmica e de verificação do funcionamento do protótipo. O acesso à webcam ocorre mediante autorização do navegador, e as imagens utilizadas na monografia para ilustrar a interface correspondem a capturas autorizadas pelo próprio autor, empregadas exclusivamente para demonstrar o funcionamento do sistema.

Como limitação desta etapa, o protótipo não implementa mecanismos avançados de anonimização, controle de acesso multiusuário ou gerenciamento formal de consentimento, uma vez que os testes foram conduzidos localmente e pelo próprio autor. Em uma evolução futura da aplicação, especialmente em testes com múltiplos usuários, seria necessário incorporar termos de consentimento, políticas de retenção e exclusão das imagens coletadas, além de controles adicionais de segurança e privacidade.

#### 4.3.9 Ambiente local de execução e testes

Os testes do protótipo foram realizados em ambiente local, utilizando um notebook Acer Nitro V 15, modelo Nitro ANV15-51. O computador utilizado possuía sistema operacional Microsoft Windows 11 Pro de 64 bits, processador 13th Gen Intel(R) Core(TM) i5-13420H, 16 GB de memória RAM, placa de vídeo dedicada NVIDIA GeForce RTX 3050 6GB Laptop GPU e GPU integrada Intel UHD Graphics.

A captura de vídeo foi realizada por meio da webcam integrada do equipamento, com resolução HD 720p. A aplicação foi testada nos navegadores Mozilla Firefox e Google Chrome, com o *frontend* e o *backend* executados localmente no mesmo equipamento. A inferência foi realizada com o modelo YOLOv8n carregado no *backend* por meio da biblioteca Ultralytics YOLO, utilizando a GPU dedicada NVIDIA GeForce RTX 3050 6GB Laptop GPU, versão para notebook, com suporte do PyTorch.

Os testes funcionais foram conduzidos pelo próprio autor do trabalho, em ambiente local e com finalidade acadêmica, totalizando 51 *feedbacks* registrados no mecanismo de avaliação do protótipo. Esses registros foram utilizados para verificar o funcionamento do armazenamento local, do *dashboard* e da matriz de confusão baseada nos *feedbacks*. Dessa forma, a avaliação não teve como objetivo representar uma validação estatística com múltiplos usuários, mas sim verificar o fluxo completo da aplicação em execução local.

Essa configuração permitiu avaliar o funcionamento do protótipo em um cenário controlado de uso, observando aspectos como resposta da interface, funcionamento da

captura por *webcam*, comunicação entre *frontend* e *backend*, exibição dos resultados da inferência e registro dos *feedbacks* do usuário.

Tabela 2 – Configuração do ambiente utilizado nos testes do protótipo.

| Componente                          | Configuração                                                           |
|-------------------------------------|------------------------------------------------------------------------|
| Equipamento                         | Acer Nitro V 15, modelo Nitro ANV15-51                                 |
| Sistema operacional                 | Microsoft Windows 11 Pro, 64 bits                                      |
| Processador                         | 13th Gen Intel(R) Core(TM) i5-13420H                                   |
| Memória RAM                         | 16 GB                                                                  |
| GPU dedicada                        | NVIDIA GeForce RTX 3050 6GB Laptop GPU                                 |
| GPU integrada                       | Intel(R) UHD Graphics                                                  |
| Câmera                              | Webcam integrada HD 720p                                               |
| Navegadores testados                | Mozilla Firefox e Google Chrome                                        |
| Modelo de inferência                | YOLOv8n                                                                |
| Dispositivos de inferência testados | GPU dedicada NVIDIA GeForce RTX 3050 Laptop e CPU Intel Core i5-13420H |
| Usuários dos testes                 | 1 usuário (autor do trabalho)                                          |
| Feedbacks registrados               | 51 registros                                                           |
| Ambiente de execução                | Local                                                                  |

#### 4.3.10 Disponibilização da ferramenta

O código-fonte do protótipo desenvolvido foi disponibilizado em repositório público, permitindo a consulta da implementação e a reprodução da aplicação em ambiente local. O repositório contém os módulos de *frontend* e *backend*, além dos arquivos necessários para execução do sistema e das instruções básicas de uso.

A aplicação não foi hospedada publicamente nesta etapa, sendo executada localmente durante os testes funcionais. Para reprodução, é necessário iniciar o *backend*, responsável pelo carregamento do modelo YOLOv8n e pela execução das inferências, e em seguida iniciar o *frontend*, responsável pela interface web, captura da webcam, exibição das predições e registro dos *feedbacks*.

O *backend* requer ambiente Python com as bibliotecas FastAPI, Ultralytics, PyTorch, OpenCV e NumPy, além do arquivo de pesos **best.pt** correspondente ao modelo YOLOv8n treinado. O *frontend* requer ambiente Node.js, com instalação das dependências do projeto React, TypeScript e Vite. O arquivo de pesos do modelo, **best.pt**, foi incluído no diretório do *backend* utilizado para o carregamento da inferência.

O repositório contém instruções de execução local no arquivo **README.md**, incluindo

a instalação das dependências do *backend* e do *frontend*, os comandos para iniciar os dois módulos da aplicação e a localização do arquivo de pesos `best.pt`. De forma geral, a execução envolve instalar as dependências Python do *backend*, iniciar o servidor FastAPI, instalar as dependências Node.js do *frontend* e executar o servidor de desenvolvimento do Vite.

O repositório do projeto está disponível em: <https://github.com/pauloricf/tcc2>.

#### 4.3.11 Procedimento de avaliação do protótipo

A avaliação do protótipo foi realizada por meio de testes funcionais em ambiente local, com o objetivo de verificar a integração entre os módulos da aplicação e observar o comportamento do sistema durante a inferência em tempo real. Para isso, o backend foi executado localmente com o modelo YOLOv8n carregado, enquanto o frontend foi acessado pelo navegador no mesmo equipamento utilizado para os testes.

Inicialmente, foi verificado o funcionamento da captura de vídeo pela webcam integrada, observando se a imagem era exibida corretamente na interface e se a conexão entre cliente e servidor era estabelecida. Em seguida, foram realizados testes de inferência em tempo real, nos quais diferentes expressões faciais foram realizadas diante da câmera, permitindo observar o retorno das predições, a exibição das caixas delimitadoras, a classe reconhecida e o nível de confiança associado.

Também foi avaliado o funcionamento do mecanismo de feedback do usuário. Durante os testes, as predições exibidas pelo sistema foram avaliadas manualmente, podendo ser marcadas como corretas, incorretas, indefinidas ou como casos sem detecção. Quando a predição era considerada incorreta, registrava-se a classe esperada, permitindo a formação de registros úteis para análise posterior.

Após a coleta dos feedbacks, foi verificado se os dados eram armazenados corretamente na base local e se as informações eram apresentadas no dashboard. Essa etapa incluiu a conferência dos indicadores agregados, da distribuição dos feedbacks, dos erros por classe prevista, da matriz de confusão baseada nas avaliações do usuário e dos registros recentes exibidos pela interface.

A avaliação teve caráter funcional e qualitativo, buscando verificar se o protótipo executava corretamente o fluxo completo proposto: captura da imagem, envio ao backend, inferência com o modelo YOLOv8n, retorno das predições, exibição visual ao usuário, registro dos feedbacks e consolidação dos dados no dashboard. Assim, o foco dessa etapa não foi substituir a avaliação quantitativa realizada com o conjunto de teste, mas complementar os resultados experimentais por meio da análise prática do sistema em execução.

## 5 RESULTADOS

Esta seção apresenta os resultados obtidos com cada uma das arquiteturas avaliadas, seguindo o protocolo descrito na Metodologia. São reportados os desempenhos globais, métricas por classe, velocidades de inferência e análises comparativas diretas entre as versões testadas.

### 5.1 PROTOCOLO EXPERIMENTAL

Todos os modelos foram treinados seguindo exatamente o mesmo protocolo descrito na Seção Metodologia, incluindo os mesmos hiperparâmetros, augmentations, divisão do dataset e resolução de entrada. Esse procedimento assegura uma comparação justa entre as versões YOLO avaliadas, pois elimina a influência de fatores externos ao modelo.

### 5.2 MÉTRICAS E AMBIENTE

Os experimentos foram conduzidos em uma GPU NVIDIA T4 disponível na plataforma Kaggle, utilizando resolução de 640 px. Os modelos foram avaliados pelas métricas *precision* (precisão), *recall* (revocação), *mAP50*, *mAP50-95*, além de medidas de eficiência, como número de parâmetros, *GFLOPs* e velocidade de inferência em *FPS*.

### 5.3 RESULTADOS DOS MODELOS

O modelo YOLOv8n possui 168 camadas, aproximadamente 3,01 milhões de parâmetros e 8,9 GFLOPs. Foi treinado por 45 épocas, resultando em um tempo total de aproximadamente 8,99 h na GPU T4. A Tabela 3 apresenta o desempenho global do modelo no conjunto de teste.

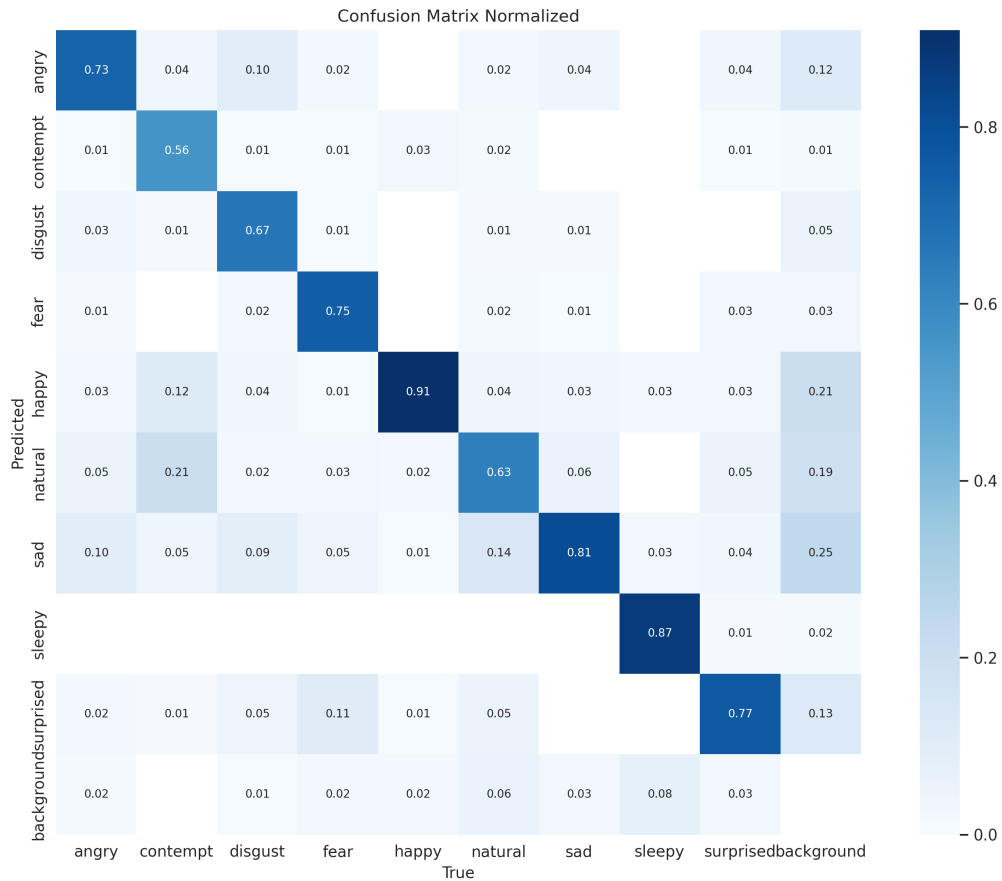
Tabela 3 – Desempenho global no conjunto de teste (*YOLOv8n*).

| Métrica | P     | R     | mAP0,50 | mAP0,50:0,95 |
|---------|-------|-------|---------|--------------|
| Valor   | 0,775 | 0,792 | 0,837   | 0,681        |

As métricas por classe estão disponíveis na Tabela 4, incluindo a quantidade de instâncias por categoria.

Tabela 4 – Métricas por classe no teste (*YOLOv8n*).

| Classe                      | Instâncias | P     | R     | mAP50 | mAP50–95 |
|-----------------------------|------------|-------|-------|-------|----------|
| <i>angry</i> (raiva)        | 297        | 0.807 | 0.825 | 0.879 | 0.728    |
| <i>contempt</i> (desprezo)  | 68         | 0.655 | 0.698 | 0.727 | 0.614    |
| <i>disgust</i> (nojo)       | 98         | 0.712 | 0.731 | 0.789 | 0.728    |
| <i>fear</i> (medo)          | 144        | 0.834 | 0.826 | 0.879 | 0.749    |
| <i>happy</i> (feliz)        | 407        | 0.900 | 0.936 | 0.947 | 0.789    |
| <i>natural</i> (neutra)     | 136        | 0.628 | 0.684 | 0.745 | 0.539    |
| <i>sad</i> (triste)         | 298        | 0.817 | 0.775 | 0.852 | 0.642    |
| <i>sleepy</i> (sonolento)   | 33         | 0.795 | 0.822 | 0.820 | 0.564    |
| <i>surprised</i> (surpreso) | 219        | 0.824 | 0.826 | 0.892 | 0.773    |

Figura 6 – Matriz de confusão – *YOLOv8n*.

A velocidade de inferência foi medida com *batch*=1 e 100 repetições, resultando em 17,80 ms por imagem (56,17 FPS).

Os resultados do *YOLOv8n* indicam um desempenho equilibrado entre precisão e revocação, com mAP50 de 0,837 e mAP50–95 de 0,681. O modelo obteve acertos altos

em classes bem definidas, como *happy*, *fear* e *surprised*, mas apresentou dificuldades em categorias menos representadas ou mais sutis, especialmente *contempt* e *natural*. A velocidade de 56,17 FPS reforça seu potencial para aplicações em tempo real, mantendo boa acurácia mesmo com arquitetura compacta.

O modelo YOLOv10n apresenta 223 camadas, cerca de 2,78 milhões de parâmetros e 8,7 GFLOPs. Após 45 épocas de treinamento, obteve um tempo total de aproximadamente 10,7 h. A Tabela 5 apresenta o desempenho global.

Tabela 5 – Desempenho global no conjunto de teste (*YOLOv10n*).

| Métrica | P     | R     | mAP50 | mAP50–95 |
|---------|-------|-------|-------|----------|
| Valor   | 0,791 | 0,776 | 0,827 | 0,668    |

As métricas por classe seguem na Tabela 6.

Tabela 6 – Métricas por classe no teste (*YOLOv10n*).

| Classe                      | Instâncias | P     | R     | mAP50 | mAP50–95 |
|-----------------------------|------------|-------|-------|-------|----------|
| <i>angry</i> (raiva)        | 297        | 0.817 | 0.805 | 0.869 | 0.709    |
| <i>contempt</i> (desprezo)  | 68         | 0.721 | 0.691 | 0.736 | 0.626    |
| <i>disgust</i> (nojo)       | 98         | 0.741 | 0.701 | 0.784 | 0.708    |
| <i>fear</i> (medo)          | 144        | 0.835 | 0.833 | 0.884 | 0.761    |
| <i>happy</i> (feliz)        | 407        | 0.886 | 0.909 | 0.944 | 0.772    |
| <i>natural</i> (neutra)     | 136        | 0.644 | 0.654 | 0.704 | 0.512    |
| <i>sad</i> (triste)         | 298        | 0.784 | 0.742 | 0.802 | 0.622    |
| <i>sleepy</i> (sonolento)   | 33         | 0.876 | 0.858 | 0.852 | 0.577    |
| <i>surprised</i> (surpreso) | 219        | 0.816 | 0.789 | 0.867 | 0.724    |

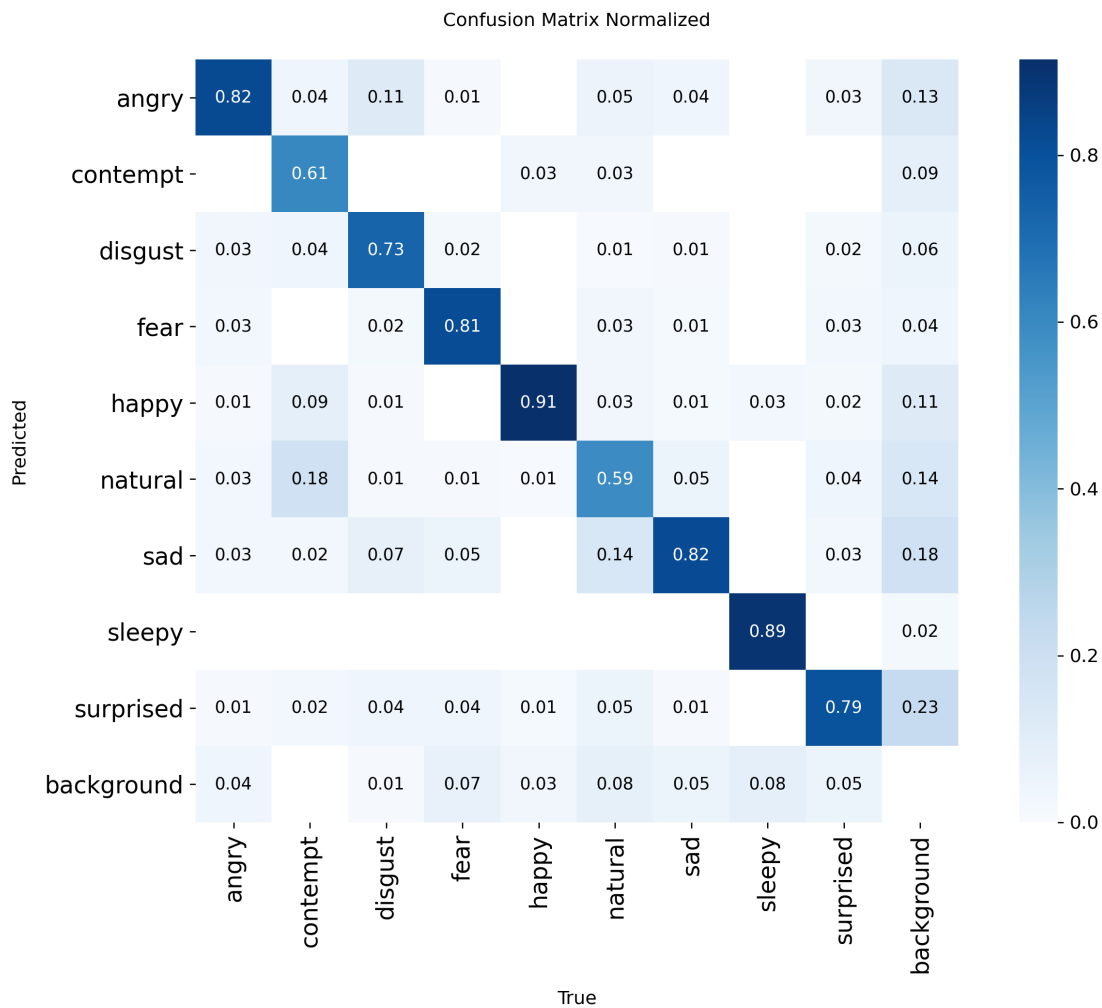


Figura 7 – Matriz de confusão – *YOLOv10n*.

A velocidade de inferência foi medida com *batch*=1 e 100 repetições, resultando em 13,54 ms por imagem (73,85 FPS).

O *YOLOv10n* apresentou métricas próximas às do *YOLOv8n*, com leve queda em mAP50–95 (0,668), embora com precisão global ligeiramente superior. As classes mais frequentes, como *happy* e *fear*, mantiveram desempenho elevado, mas houve maior variabilidade em expressões sutis, como *natural*. Seu principal destaque é a velocidade: 73,85 FPS a maior entre todos os modelos avaliados, o que o torna o mais eficiente em termos de tempo de inferência.

O modelo *YOLOv11n* possui 181 camadas, cerca de 2,64 milhões de parâmetros e 6,6 GFLOPs, com um tempo total de treinamento de aproximadamente 8,79 h. A Tabela 7 apresenta o desempenho global.

Tabela 7 – Desempenho global no conjunto de teste (*YOLOv11n*).

| Métrica | P     | R     | mAP50 | mAP50–95 |
|---------|-------|-------|-------|----------|
| Valor   | 0,751 | 0,790 | 0,825 | 0,673    |

A Tabela 8 apresenta as métricas por classe.

Tabela 8 – Métricas por classe no teste (*YOLOv11n*).

| Classe                      | Instâncias | P     | R     | mAP50 | mAP50–95 |
|-----------------------------|------------|-------|-------|-------|----------|
| <i>angry</i> (raiva)        | 297        | 0.805 | 0.798 | 0.855 | 0.706    |
| <i>contempt</i> (desprezo)  | 68         | 0.728 | 0.708 | 0.752 | 0.630    |
| <i>disgust</i> (nojo)       | 98         | 0.655 | 0.718 | 0.772 | 0.713    |
| <i>fear</i> (medo)          | 144        | 0.805 | 0.819 | 0.885 | 0.756    |
| <i>happy</i> (feliz)        | 407        | 0.879 | 0.926 | 0.948 | 0.783    |
| <i>natural</i> (neutra)     | 136        | 0.667 | 0.691 | 0.718 | 0.531    |
| <i>sad</i> (triste)         | 298        | 0.770 | 0.774 | 0.828 | 0.636    |
| <i>sleepy</i> (sonolento)   | 33         | 0.634 | 0.879 | 0.793 | 0.553    |
| <i>surprised</i> (surpreso) | 219        | 0.813 | 0.799 | 0.874 | 0.750    |

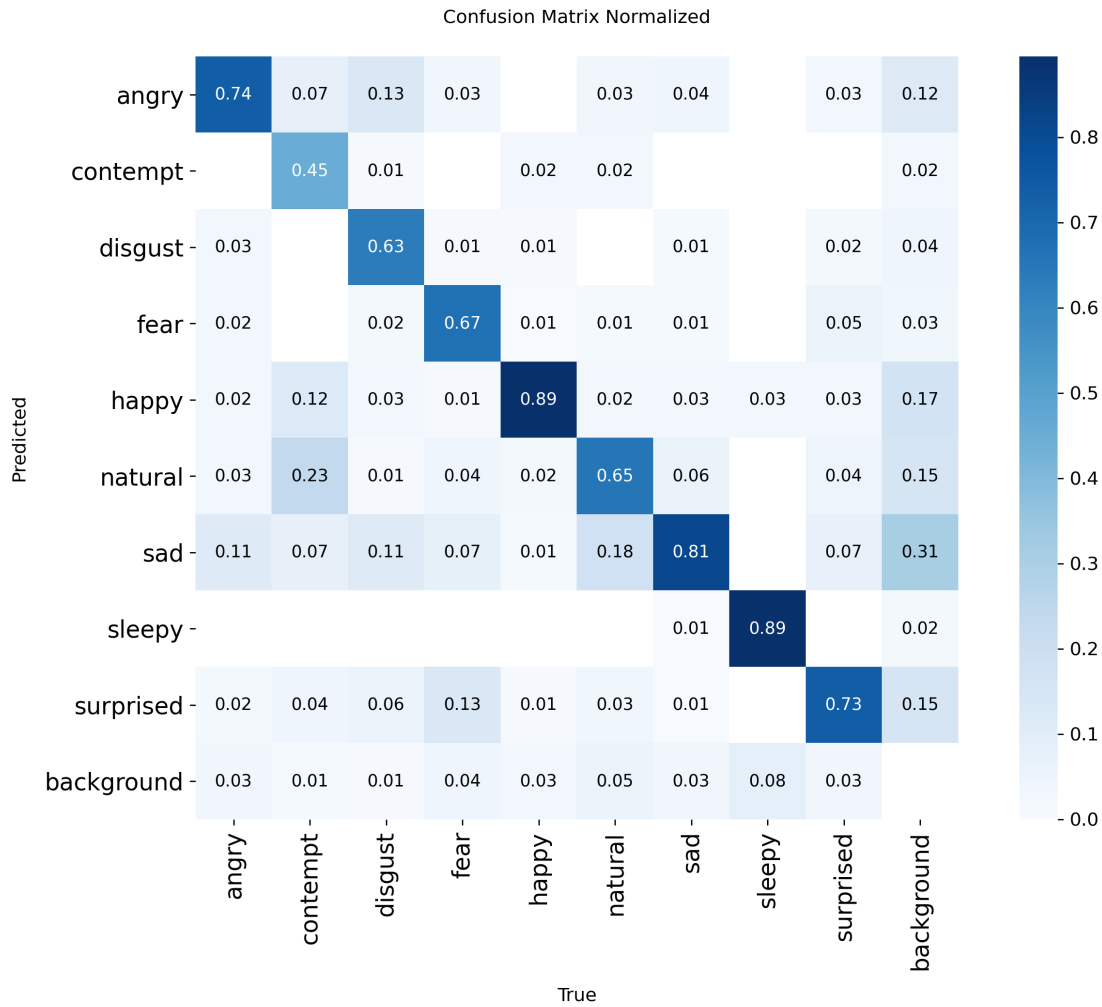


Figura 8 – Matriz de confusão – *YOLOv11n*.

A velocidade de inferência foi medida com *batch*=1 e 100 repetições, resultando em 16,02 ms por imagem (62,42 FPS).

Os resultados do *YOLOv11n* mostram desempenho intermediário, com mAP50 de 0,825 e mAP50–95 de 0,673, valores ligeiramente inferiores aos modelos anteriores. Embora tenha mantido boa performance em classes como *happy* e *surprised*, houve queda perceptível em *disgust* e *natural*. Sua velocidade de inferência (62,42 FPS) permanece adequada para aplicações em tempo real, mas o equilíbrio entre acurácia e custo computacional não se destacou em relação ao *YOLOv8n* ou *YOLOv10n*.

Por fim, o modelo *YOLOv12n* apresenta 272 camadas, 2,60 milhões de parâmetros e 6,7 GFLOPs. Foi treinado por 45 épocas em um tempo total de aproximadamente 14,63 h. A Tabela 9 apresenta o desempenho global.

Tabela 9 – Desempenho global no conjunto de teste (*YOLOv12n*).

| Métrica | P     | R     | mAP50 | mAP50–95 |
|---------|-------|-------|-------|----------|
| Valor   | 0,785 | 0,814 | 0,839 | 0,686    |

Tabela 10 – Métricas por classe no teste (*YOLOv12n*).

| Classe                      | Instâncias | P     | R     | mAP50 | mAP50–95 |
|-----------------------------|------------|-------|-------|-------|----------|
| <i>angry</i> (raiva)        | 297        | 0.818 | 0.838 | 0.880 | 0.739    |
| <i>contempt</i> (desprezo)  | 68         | 0.682 | 0.693 | 0.726 | 0.612    |
| <i>disgust</i> (nojo)       | 98         | 0.684 | 0.796 | 0.809 | 0.746    |
| <i>fear</i> (medo)          | 144        | 0.862 | 0.847 | 0.895 | 0.765    |
| <i>happy</i> (feliz)        | 407        | 0.898 | 0.934 | 0.958 | 0.791    |
| <i>natural</i> (neutra)     | 136        | 0.662 | 0.721 | 0.734 | 0.538    |
| <i>sad</i> (triste)         | 298        | 0.771 | 0.789 | 0.829 | 0.633    |
| <i>sleepy</i> (sonolento)   | 33         | 0.879 | 0.879 | 0.843 | 0.590    |
| <i>surprised</i> (surpreso) | 219        | 0.812 | 0.831 | 0.880 | 0.758    |

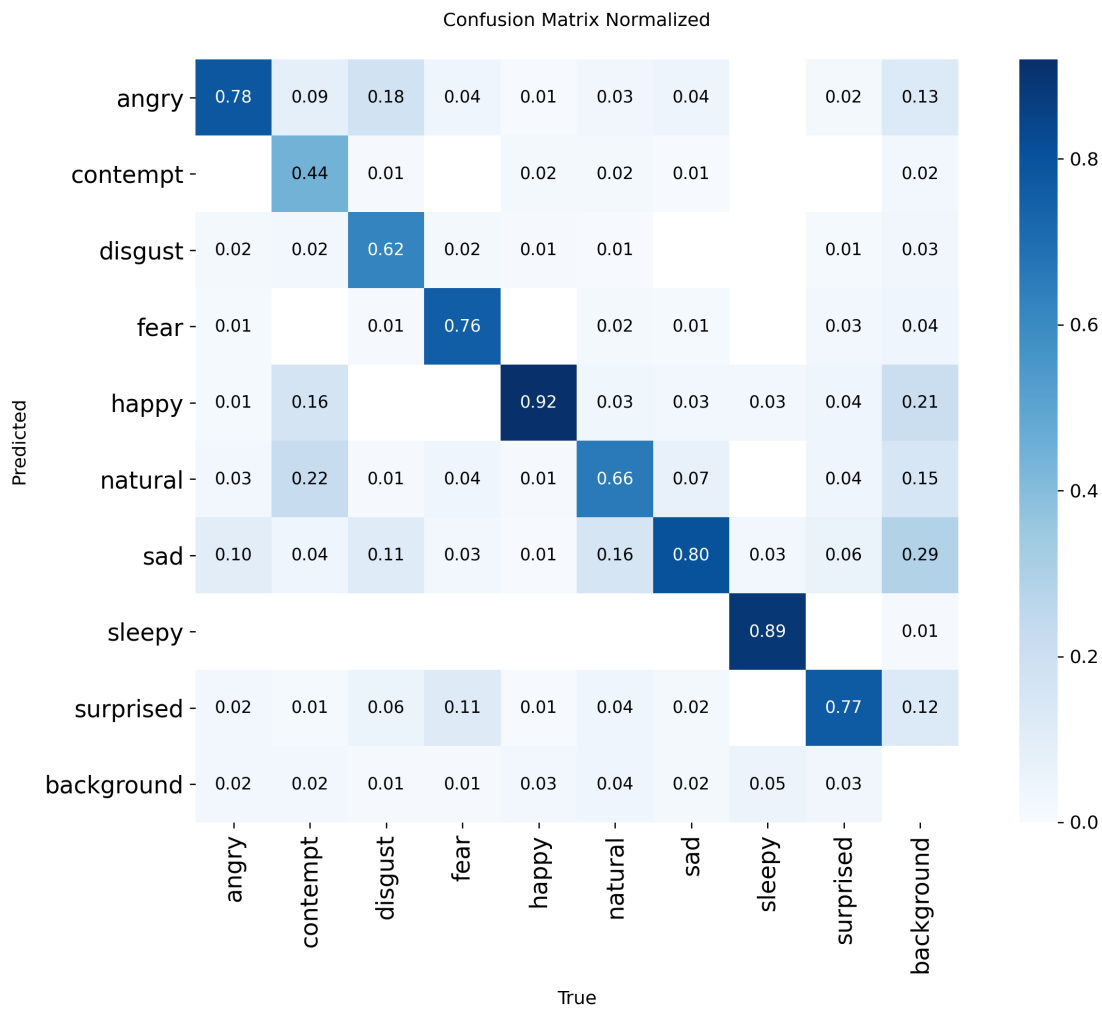


Figura 9 – Matriz de confusão – *YOLOv12n*.

A velocidade de inferência foi medida com *batch*=1 e 100 repetições, resultando em 21,22 ms por imagem (47,11 FPS).

O *YOLOv12n* apresentou leve vantagem preditiva entre os modelos avaliados, com mAP50 de 0,839 e mAP50-95 de 0,686. As métricas por classe mostram elevada consistência em quase todas as categorias, especialmente *fear*, *happy* e *surprised*. Apesar disso, seu tempo de inferência (47,11 FPS) foi o mais baixo, devido ao aumento da complexidade arquitetural. Assim, o modelo apresentou leve ganho preditivo nos experimentos realizados, mas com maior custo temporal de inferência, tornando sua aplicação dependente dos requisitos de desempenho em tempo real.

## 5.4 COMPARAÇÃO GERAL ENTRE MODELOS

A Tabela 11 apresenta os resultados obtidos com as quatro versões avaliadas. De modo geral, os modelos apresentaram comportamentos semelhantes, com diferenças sutis em precisão, velocidade e custo computacional. Entre eles, o YOLOv12n apresentou leve vantagem preditiva nos experimentos realizados, registrando o maior  $mAP50-95$  (0,686) e também o maior  $mAP50$ . No entanto, como esses ganhos são relativamente pequenos, é possível que parte dessa diferença seja decorrente da variabilidade natural do dataset ou de efeitos estocásticos do processo de treinamento. Assim, embora o YOLOv12n demonstre uma tendência de melhora, não se pode afirmar que suas modificações estruturais recentes resultem em avanços substanciais para esta tarefa específica.

Tabela 11 – Comparativo entre versões (mesmo protocolo experimental).

| Modelo   | Parâmetros | GFLOPs | P     | R     | mAP50 | mAP50-95 | FPS   | Tempo de Treino |
|----------|------------|--------|-------|-------|-------|----------|-------|-----------------|
| YOLOv8n  | ~3,01M     | 8,9    | 0,775 | 0,792 | 0,837 | 0,681    | 56,17 | ~8,99h          |
| YOLOv10n | ~2,78M     | 8,7    | 0,791 | 0,776 | 0,827 | 0,668    | 73,85 | ~10,7h          |
| YOLOv11n | ~2,64M     | 6,6    | 0,751 | 0,790 | 0,825 | 0,673    | 62,42 | ~8,79h          |
| YOLOv12n | ~2,60M     | 6,7    | 0,785 | 0,814 | 0,839 | 0,686    | 47,11 | ~14,63h         |

Ao analisar o custo computacional das arquiteturas, representado pelos valores de GFLOPs, observa-se que modelos com menor número de operações, como YOLOv11n (6,6 GFLOPs) e YOLOv12n (6,7 GFLOPs), tendem a exigir menos processamento por imagem, favorecendo implementações em dispositivos com restrições de hardware. Em contraste, YOLOv8n e YOLOv10n apresentam custos maiores (8,9 e 8,7 GFLOPs), refletindo arquiteturas relativamente mais pesadas.

Entretanto, a relação entre GFLOPs, velocidade (FPS) e qualidade das predições não é linear. O YOLOv10n, por exemplo, possui um dos maiores custos computacionais entre os modelos analisados, mas ainda assim foi o mais rápido, alcançando 73,85 FPS. Esse comportamento indica que a eficiência da arquitetura interna exerce impacto significativo sobre a latência, indo além do simples número de operações. Já o YOLOv12n, apesar de um dos menores GFLOPs, apresentou a menor velocidade (47,11 FPS), o que sugere a presença de mecanismos internos mais profundos ou menos paralelizáveis.

Essa análise reforça que os GFLOPs fornecem apenas uma perspectiva complementar e não determinística sobre o desempenho real das arquiteturas, ajudando a compreender os *trade-offs* entre custo computacional, velocidade e qualidade preditiva.

Quando a comparação se concentra no desempenho preditivo, nota-se que o YOLOv8n apresentou métricas muito próximas às do YOLOv12n. A diferença de apenas 0,005 no  $mAP50-95$  sugere que, para este conjunto de dados, as versões mais recentes não introduziram ganhos substanciais na tarefa. Assim, mesmo sendo uma geração anterior,

o YOLOv8n permanece altamente competitivo e bem adaptado ao reconhecimento de expressões faciais.

Os modelos intermediários, YOLOv10n e YOLOv11n, exibiram desempenhos estáveis, porém ligeiramente inferiores aos demais em termos de precisão. Apesar de possuírem menos parâmetros, essa redução de complexidade não se converteu em ganhos perceptíveis de acurácia. Em outras palavras, embora sejam arquiteturas mais leves, não apresentaram vantagens claras no desempenho preditivo.

Quanto à eficiência, as diferenças tornam-se mais marcantes. O YOLOv10n foi o modelo mais rápido, seguido pelo YOLOv11n. O YOLOv8n também exibiu boa velocidade (56,17 FPS), suficiente para aplicações em tempo real. Já o YOLOv12n, apesar de apresentar o melhor mAP, foi o mais lento, desempenho compatível com sua maior complexidade arquitetural.

As matrizes de confusão ajudam a contextualizar esses resultados. Expressões visualmente bem definidas, como *happy*, *fear* e *surprised*, apresentaram altas taxas de acerto em todas as versões. Por outro lado, expressões de baixa intensidade ou maior ambiguidade, como *contempt*, *natural* e *sad*, foram confundidas com maior frequência. A recorrência desse padrão entre todas as arquiteturas sugere que a dificuldade está mais associada às características do dataset do que a limitações específicas de um modelo.

Além da ambiguidade visual entre algumas expressões, o desbalanceamento do conjunto de dados também deve ser considerado na interpretação dos resultados. Algumas classes apresentam quantidade significativamente maior de instâncias no conjunto de teste, como *happy*, *sad* e *angry*, enquanto classes como *sleepy* e *contempt* possuem número reduzido de exemplos. Essa diferença pode afetar a estabilidade das métricas por classe, pois pequenas variações nas predições têm impacto proporcionalmente maior em categorias com poucas amostras. Dessa forma, os resultados por classe devem ser interpretados com cautela, especialmente nas expressões menos representadas.

Em síntese:

- O YOLOv12n apresentou leve vantagem preditiva nos experimentos realizados
- O YOLOv10n foi o modelo mais rápido.
- O YOLOv8n demonstrou o melhor equilíbrio entre precisão e velocidade.
- O YOLOv11n apresentou desempenho intermediário, sem destaque em precisão ou latência.

Esses resultados mostram que a escolha da arquitetura ideal não deve se basear exclusivamente na versão mais recente. Como as diferenças entre os modelos são relativamente pequenas, a decisão depende diretamente dos requisitos da aplicação, especialmente em cenários que exigem inferência em tempo real e equilíbrio entre acurácia e eficiência computacional.

## 5.5 SELEÇÃO DO MODELO PARA O PROTÓTIPO

A escolha do modelo utilizado no protótipo considerou não apenas o melhor valor isolado de uma métrica, mas o equilíbrio entre desempenho preditivo, velocidade de inferência, estabilidade e viabilidade prática de integração à aplicação web. Embora o YOLOv12n tenha apresentado leve vantagem nos experimentos realizados, com mAP50-95 de 0,686 contra 0,681 do YOLOv8n, essa diferença foi pequena e não representou um ganho substancial para a tarefa avaliada. Além disso, o YOLOv12n apresentou menor velocidade de inferência, alcançando 47,11 FPS, enquanto o YOLOv8n atingiu 56,17 FPS.

O YOLOv10n, por sua vez, foi o modelo mais rápido, com 73,85 FPS. Entretanto, apresentou desempenho preditivo inferior ao YOLOv8n, com mAP50 de 0,827 e mAP50-95 de 0,668. Dessa forma, apesar de sua vantagem em velocidade, a redução nas métricas de detecção tornou sua escolha menos adequada para o protótipo, cujo objetivo era manter um equilíbrio entre acurácia e tempo de resposta.

Nesse contexto, o YOLOv8n foi selecionado por apresentar desempenho preditivo muito próximo ao melhor resultado obtido, velocidade suficiente para aplicações em tempo real e menor complexidade prática para integração ao protótipo desenvolvido. Além disso, por ser uma versão consolidada da biblioteca Ultralytics, sua adoção favoreceu maior estabilidade durante a implementação, carregamento do modelo e execução das inferências no ambiente da aplicação. Assim, a escolha do YOLOv8n não se baseou apenas em uma métrica isolada, mas no conjunto de fatores necessários para a construção de uma solução funcional em tempo real, incluindo mAP, FPS, estabilidade e viabilidade prática de integração.

Considerando esse equilíbrio entre desempenho preditivo e velocidade de inferência, o YOLOv8n foi adotado como modelo base para a etapa aplicada do trabalho. A seguir, são apresentados os resultados obtidos com a integração desse modelo ao protótipo web desenvolvido.

## 5.6 RESULTADOS DO PROTÓTIPO EM TEMPO REAL

Após a seleção do YOLOv8n como modelo mais adequado para continuidade do trabalho, foi realizada a implementação e avaliação funcional do protótipo web de reconhecimento de expressões faciais em tempo real. Nesta etapa, buscou-se verificar se o sistema desenvolvido era capaz de executar o fluxo completo proposto na metodologia, contemplando a captura da imagem por webcam, o envio dos quadros ao backend, a realização da inferência com o modelo YOLOv8n, o retorno das predições à interface, o registro de feedbacks do usuário e a consolidação das informações no dashboard.

Diferentemente da etapa comparativa, cujo foco esteve na avaliação quantitativa dos modelos sobre o conjunto de teste, esta etapa teve caráter funcional e qualitativo. As-

sim, os resultados apresentados a seguir descrevem o comportamento observado durante a execução do protótipo, evidenciando as funcionalidades implementadas, os registros produzidos pelo mecanismo de feedback, a visualização dos dados no dashboard e as principais limitações identificadas durante os testes.

#### 5.6.1 Funcionamento geral da aplicação

Durante os testes locais, o protótipo executou corretamente o fluxo principal de reconhecimento de expressões faciais em tempo real. A interface web permitiu o acesso à webcam integrada do equipamento, exibindo o vídeo capturado no navegador e possibilitando a comunicação com o backend para processamento dos quadros. Após o estabelecimento da conexão, as imagens passaram a ser processadas pelo modelo YOLOv8n, e as predições retornadas foram apresentadas visualmente na própria interface.

A aplicação exibiu sobre a imagem capturada as caixas delimitadoras correspondentes às faces detectadas, acompanhadas da classe prevista e do nível de confiança associado à predição. Além disso, a interface apresentou informações complementares de execução, como dispositivo utilizado para inferência, tempo de processamento, taxa de quadros por segundo do servidor e latência estimada. Esses elementos permitiram acompanhar o comportamento do sistema durante a execução e verificar a integração entre captura, inferência e exibição dos resultados.

Durante os testes funcionais, também foi observada a diferença de desempenho entre a execução com GPU e CPU no protótipo. Utilizando a GPU dedicada NVIDIA GeForce RTX 3050 Laptop 6GB, a aplicação apresentou média aproximada de 25 FPS durante a execução em tempo real. Quando executada na CPU Intel Core i5-13420H, a taxa média observada foi de aproximadamente 7 FPS. Esses valores representam o desempenho do protótipo completo em funcionamento, considerando captura, comunicação entre os módulos, inferência e atualização da interface, diferindo dos FPS reportados na etapa comparativa, que foram medidos em um teste isolado de inferência dos modelos.

Tabela 12 – Desempenho observado do protótipo em execução local.

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

A Figura 10 apresenta a tela principal do protótipo em execução, com a captura da webcam, a detecção facial realizada pelo modelo e a exibição das informações de inferência. No exemplo apresentado, o sistema identificou a expressão *natural*, exibindo a caixa delimitadora sobre a face detectada e o nível de confiança associado à predição.

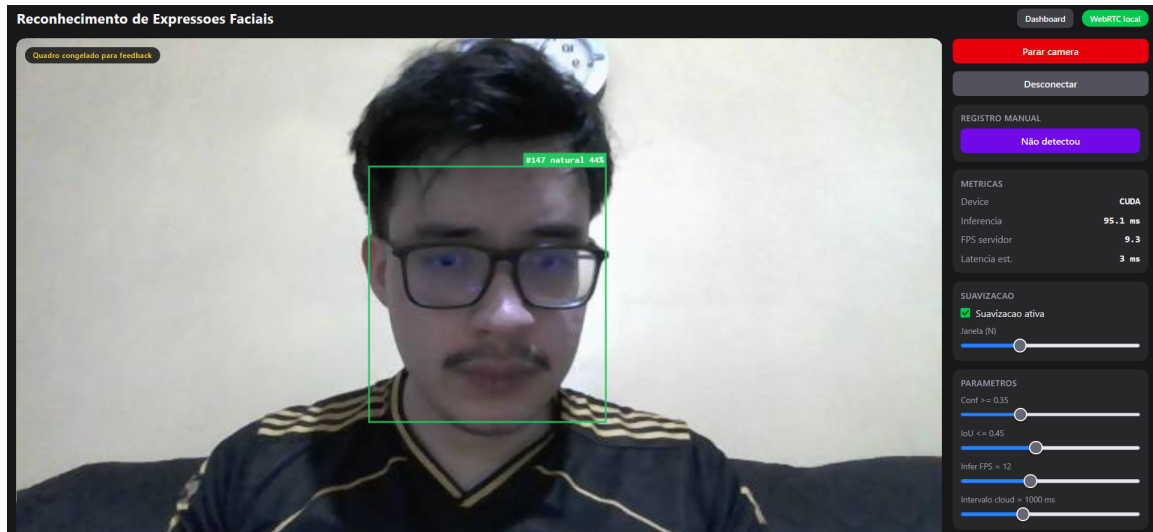


Figura 10 – Interface principal do protótipo em execução, com detecção facial, classe prevista, confiança da predição e métricas de inferência.

### 5.6.2 Avaliação funcional dos módulos do protótipo

A avaliação funcional indicou que os principais módulos previstos para o protótipo foram implementados e integrados corretamente. A captura por webcam foi realizada diretamente no navegador, o backend recebeu os quadros para processamento, o modelo YOLOv8n executou a inferência e os resultados foram retornados à interface para exibição em tempo real.

Além disso, o mecanismo de feedback permitiu registrar avaliações manuais das predições, enquanto a camada de persistência armazenou os registros localmente em SQLite. O dashboard, por sua vez, consolidou essas informações em indicadores e visualizações gráficas, permitindo analisar os feedbacks coletados durante o uso do sistema.

Tabela 13 – Avaliação funcional dos módulos do protótipo.

| 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, nível de confiança e coordenadas da caixa delimitadora. |
| Exibição das predições |                  | As caixas delimitadoras, os rótulos das classes e os níveis de confiança foram exibidos sobre o vídeo capturado.               |
| Registro de feedback   |                  | O sistema permitiu avaliar as predições como corretas, incorretas, indefinidas ou como casos sem detecção.                     |
| Persistência local     |                  | Os feedbacks registrados foram armazenados localmente em SQLite.                                                               |
| Dashboard              |                  | Os registros coletados foram consolidados em indicadores, gráficos e matriz de confusão baseada nos feedbacks.                 |

### 5.6.3 Resultados do mecanismo de feedback

O mecanismo de feedback implementado no protótipo permitiu ao usuário avaliar manualmente as predições realizadas pelo modelo durante a execução em tempo real. Quando uma detecção era exibida na interface, o sistema possibilitava o congelamento do quadro para análise e apresentava um modal de confirmação, no qual o usuário podia indicar se a predição estava correta, incorreta ou se não sabia avaliá-la.

Conforme ilustrado na Figura 11, o primeiro passo do processo consistia na apresentação da classe prevista pelo modelo ao usuário, juntamente com as opções de resposta. Quando a predição era considerada correta, o feedback era registrado diretamente no sistema. Quando a opção “Errado” era selecionada, o protótipo exibia uma segunda etapa, permitindo ao usuário informar qual seria a classe correta entre as categorias disponíveis. Esse procedimento possibilitou registrar não apenas confirmações de acerto, mas também correções de classe, tornando os dados coletados mais úteis para análise posterior no dashboard.

Esse mecanismo mostrou-se funcional durante os testes realizados, permitindo registrar diferentes tipos de retorno do usuário e alimentar a base local de dados com informações como classe prevista, confiança, tipo de feedback e classe corrigida, quando aplicável. Dessa forma, a aplicação passou a incorporar uma etapa de avaliação prática das predições, complementando a análise quantitativa realizada anteriormente com o conjunto de teste do *dataset*.

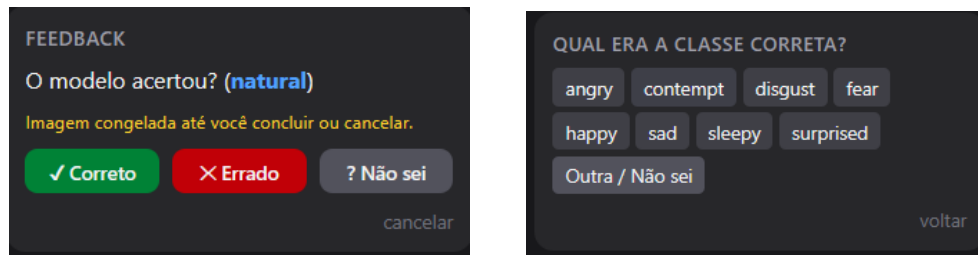


Figura 11 – Etapas do mecanismo de feedback do usuário: à esquerda, confirmação da predição realizada pelo modelo; à direita, seleção da classe correta quando a predição é marcada como incorreta.

#### 5.6.4 Análise do dashboard

O dashboard desenvolvido permitiu consolidar os feedbacks registrados durante os testes funcionais do protótipo, apresentando uma visão agregada do comportamento observado durante o uso da aplicação. Conforme apresentado na Figura 12, a interface reúne filtros por período, classe prevista e cenário, além de cartões com indicadores gerais, resumo analítico e gráficos de distribuição dos feedbacks e de erros por classe prevista.

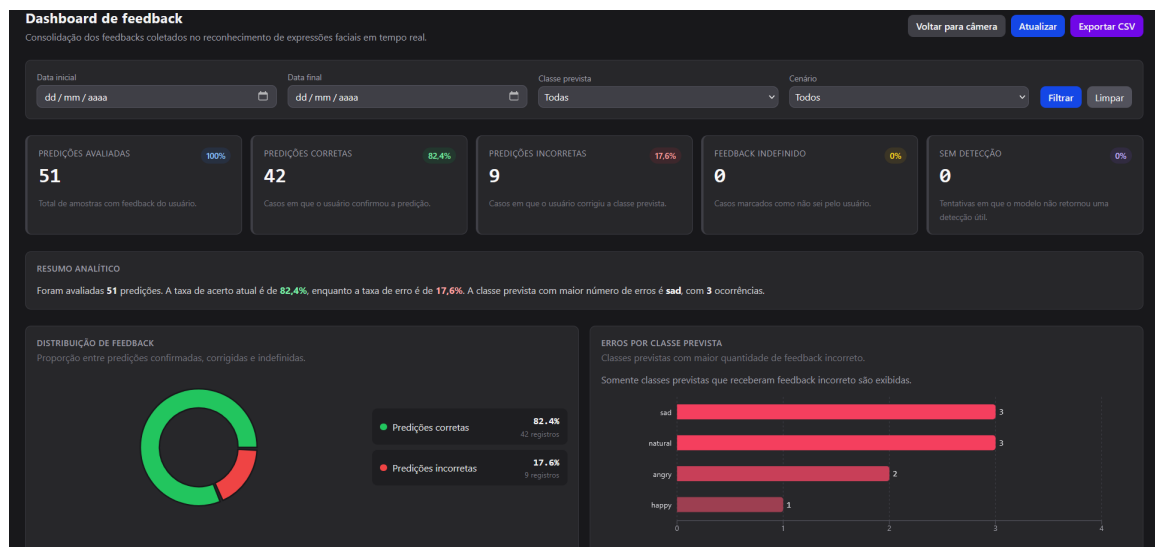


Figura 12 – Dashboard de análise dos feedbacks coletados durante o uso do protótipo.

Durante os testes realizados, foram avaliadas 51 predições registradas pelo mecanismo de feedback. Desse total, 42 foram marcadas como corretas e 9 como incorretas, resultando

em uma taxa indicativa de acerto de aproximadamente 82,4% e taxa de erro de 17,6%. Não foram registrados, nesse conjunto, casos de feedback indefinido ou situações sem detecção. Entretanto, como a quantidade de amostras ainda é reduzida e os registros foram obtidos em um cenário local de uso, esses percentuais não devem ser interpretados como uma medida conclusiva de desempenho do modelo, mas como um indicativo do funcionamento do mecanismo de coleta e da capacidade do dashboard de organizar os dados obtidos durante o uso do sistema.

Além dos indicadores gerais, o dashboard apresenta a distribuição dos feedbacks e os erros agrupados por classe prevista. Nos registros coletados, as classes *sad* e *natural* concentraram o maior número de erros observados, ambas com três ocorrências de feedback incorreto, seguidas por *angry*, com duas ocorrências, e *happy*, com uma ocorrência. Esse comportamento é compatível com as dificuldades já observadas na etapa experimental, em que expressões com maior ambiguidade visual ou baixa intensidade apresentaram maior tendência de confusão.

A Figura 13 apresenta a matriz de confusão construída a partir dos feedbacks registrados pelo usuário. Diferentemente da matriz obtida com o conjunto de teste do *dataset*, essa matriz reflete apenas as avaliações realizadas durante o uso do protótipo. A diagonal principal representa as predições confirmadas como corretas pelo usuário, enquanto os valores fora da diagonal indicam correções informadas manualmente.

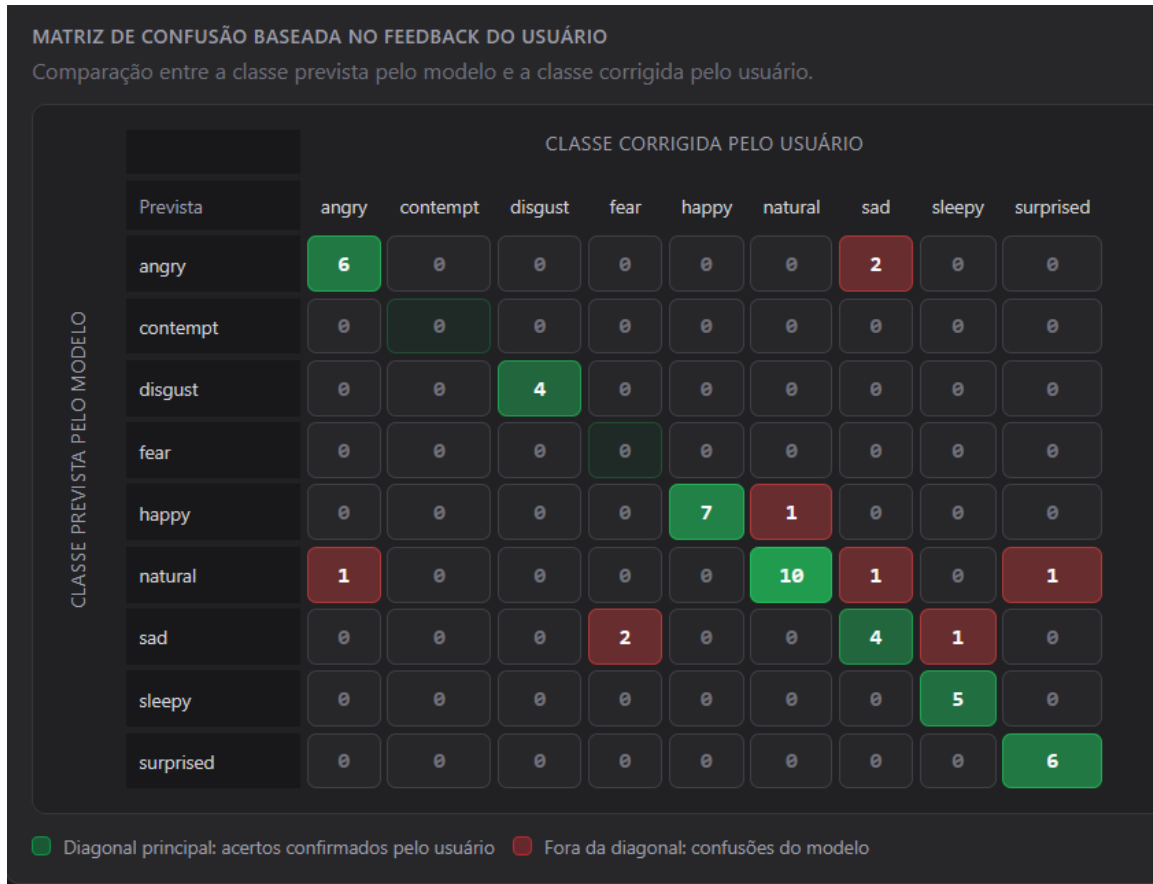


Figura 13 – Matriz de confusão construída a partir dos feedbacks registrados pelo usuário.

No conjunto analisado, observa-se maior concentração de acertos na classe *natural*, com 10 confirmações corretas, seguida pelas classes *happy*, com 7 acertos, *angry* e *surprised*, com 6 acertos cada, *sleepy*, com 5 acertos, e *sad* e *disgust*, com 4 acertos cada. As confusões observadas envolveram principalmente predições de *sad* corrigidas como *fear* ou *sleepy*, predições de *natural* corrigidas como *angry*, *sad* ou *surprised*, predições de *angry* corrigidas como *sad* e uma predição de *happy* corrigida como *natural*.

Esses resultados indicam que, embora o protótipo tenha apresentado funcionamento adequado durante os testes, algumas confusões permanecem em expressões visualmente próximas ou de reprodução subjetiva, especialmente entre *angry*, *sad*, *natural*, *fear* e *sleepy*. Além disso, como algumas classes não tiveram registros suficientes, como *contempt* e *fear*, a matriz deve ser interpretada apenas como uma análise complementar e indicativa do comportamento do sistema em uso, não como uma validação estatística do desempenho por classe.

Dessa forma, o dashboard demonstrou ser útil para complementar a avaliação quantitativa realizada na etapa comparativa, permitindo acompanhar o comportamento do modelo em um cenário de uso interativo. Embora os dados coletados ainda sejam limitados, a funcionalidade mostrou-se adequada para registrar avaliações do usuário, organizar os resultados e evidenciar padrões iniciais de acerto e erro do protótipo.

### 5.6.5 Limitações observadas

Durante os testes funcionais, observou-se que algumas expressões foram reproduzidas e reconhecidas com maior facilidade pelo protótipo, especialmente *happy*, *angry*, *natural* e *sleepy*. Por outro lado, expressões mais sutis ou visualmente ambíguas, como *sad*, *contempt*, *disgust* e *fear*, apresentaram maior dificuldade de reprodução e avaliação durante o uso com webcam. Essa limitação está associada tanto à subjetividade da execução manual das expressões quanto à proximidade visual entre algumas categorias de expressão facial.

Essa dificuldade pôde ser observada especialmente na classe *disgust*. Em execuções realizadas com a webcam integrada do notebook, essa expressão não foi reconhecida com a mesma facilidade observada em outras categorias. Entretanto, em um teste complementar realizado pelo navegador de um smartphone Samsung Galaxy S25, utilizando apenas a câmera do dispositivo como fonte de captura, o sistema retornou corretamente a classe *disgust*. Nesse caso, o processamento permaneceu sendo realizado pelo mesmo *backend* e pelo mesmo modelo YOLOv8n, alterando-se apenas o dispositivo de captura da imagem. Esse comportamento sugere que a qualidade da câmera, incluindo resolução, foco, nitidez e controle de exposição, pode influenciar diretamente o desempenho percebido do protótipo.

Ressalta-se que esse teste teve caráter apenas complementar e pontual, não constituindo uma avaliação sistemática em dispositivos móveis. Ainda assim, ele indica que diferentes câmeras podem produzir variações relevantes na qualidade dos quadros enviados ao modelo, afetando especialmente expressões mais sutis ou de difícil reprodução.

Esse comportamento é coerente com os resultados obtidos na etapa comparativa, na qual as matrizes de confusão indicaram maior dificuldade em classes de baixa intensidade visual ou com maior semelhança entre si. Enquanto expressões mais marcadas, como *happy* e *surprised*, apresentaram desempenho mais consistente nos testes com o conjunto de dados, categorias como *natural*, *sad* e *contempt* demonstraram maior tendência à confusão. Assim, as limitações observadas no protótipo não decorrem apenas da implementação da aplicação, mas também refletem desafios próprios da tarefa de reconhecimento de expressões faciais em condições não controladas.

Outro fator que pode contribuir para essas diferenças é o desbalanceamento do conjunto de dados utilizado no treinamento e na avaliação. Classes com menor número de exemplos, como *sleepy* e *contempt*, tendem a produzir métricas menos estáveis e podem dificultar a generalização do modelo para situações reais de uso. Assim, parte das limitações observadas durante a execução do protótipo também pode estar relacionada à distribuição desigual das amostras entre as classes.

Também foi observado que variações na pose da face, especialmente quando o rosto era rotacionado lateralmente, reduziram a estabilidade das detecções. Em alguns casos, o modelo deixou de detectar a face ou apresentou níveis de confiança baixos, dificultando

a confirmação da classe prevista. Esse resultado indica que o protótipo apresenta melhor funcionamento em condições frontais, com iluminação adequada e enquadramento claro da face, mas ainda possui limitações em situações com maior variação de pose, intensidade da expressão e qualidade da captura.

Outra limitação está relacionada ao tamanho reduzido do conjunto de feedbacks coletados durante os testes funcionais. Como os registros foram obtidos em um cenário local e com quantidade limitada de amostras, os percentuais apresentados no dashboard devem ser interpretados apenas como indicativos iniciais do funcionamento do mecanismo de feedback, e não como uma métrica conclusiva de desempenho do modelo. A avaliação quantitativa principal permanece sendo aquela realizada com o conjunto de teste do *dataset*, enquanto os feedbacks do protótipo funcionam como uma análise complementar do comportamento do sistema em uso interativo.

Além disso, os testes foram realizados principalmente com a webcam integrada de um único notebook, sendo complementados por um teste pontual em navegador móvel. Essa condição limita a generalização das observações para outros dispositivos, resoluções de câmera, condições de iluminação e perfis de usuário.

Dessa forma, embora o protótipo tenha demonstrado funcionamento adequado para o fluxo proposto de inferência em tempo real, registro de feedbacks e visualização em dashboard, os testes evidenciam que a robustez do sistema ainda depende de fatores como pose facial, iluminação, intensidade da expressão, qualidade da câmera e diversidade dos usuários avaliados.

## 6 CONCLUSÕES

Este trabalho teve como objetivo desenvolver, avaliar comparativamente e aplicar modelos baseados na arquitetura YOLO para o reconhecimento de expressões faciais em tempo real. Para isso, a pesquisa foi conduzida em duas etapas complementares: inicialmente, realizou-se uma avaliação comparativa entre diferentes versões da família YOLO em um conjunto de dados de expressões faciais em condições não controladas; em seguida, o modelo selecionado foi integrado a um protótipo web funcional, permitindo a execução de inferências em tempo real, o registro de feedbacks do usuário e a análise dos resultados por meio de um dashboard.

Na etapa comparativa, foram avaliadas as arquiteturas *YOLOv8n*, *YOLOv10n*, *YOLOv11n* e *YOLOv12n*, todas em suas versões *nano*. A utilização do mesmo protocolo de treinamento, das mesmas divisões do conjunto de dados e das mesmas métricas de avaliação permitiu uma comparação justa entre os modelos. Os resultados mostraram que todas as versões apresentaram desempenho compatível com aplicações em tempo real, embora com diferenças em termos de acurácia, velocidade de inferência e custo computacional.

Entre os modelos avaliados, o *YOLOv12n* apresentou leve vantagem preditiva nos experimentos realizados, alcançando os maiores valores de *mAP50* e *mAP50-95*. Entretanto, a vantagem em relação ao *YOLOv8n* foi pequena, enquanto seu tempo de inferência foi superior. O *YOLOv10n*, por sua vez, apresentou a maior velocidade de inferência, mas com desempenho preditivo inferior. Já o *YOLOv11n* demonstrou comportamento intermediário, sem se destacar de forma significativa em precisão ou velocidade. Nesse contexto, o *YOLOv8n* foi selecionado como o modelo mais adequado para a etapa aplicada, por apresentar o melhor equilíbrio entre desempenho preditivo, velocidade e viabilidade prática.

A análise por classe e as matrizes de confusão evidenciaram que o reconhecimento de expressões faciais em cenários não controlados continua sendo uma tarefa desafiadora. Expressões visualmente mais marcadas, como *happy*, *fear* e *surprised*, apresentaram desempenho mais consistente nos modelos avaliados. Por outro lado, categorias mais sutis ou ambíguas, como *contempt*, *natural* e *sad*, demonstraram maior tendência à confusão. Esse comportamento indica que parte das limitações observadas está relacionada não apenas às arquiteturas utilizadas, mas também às características do próprio conjunto de dados, incluindo desbalanceamento entre classes, baixo número de instâncias em categorias como *sleepy* e *contempt*, variações de iluminação, pose facial e intensidade das expressões.

Na etapa aplicada, o modelo *YOLOv8n* foi integrado a um protótipo web de reconhecimento de expressões faciais em tempo real. A aplicação permitiu capturar imagens por *webcam*, enviar os quadros ao *backend*, executar a inferência com o modelo treinado e

exibir na interface a face detectada, a classe prevista, o nível de confiança e as métricas de execução. Dessa forma, foi possível demonstrar a aplicação prática do modelo selecionado em um ambiente interativo, aproximando a pesquisa de um cenário interativo de uso.

Além da inferência em tempo real, o protótipo incorporou um mecanismo de feedback manual, permitindo que o usuário avaliasse as predições como corretas, incorretas, indefinidas ou como casos sem detecção. Os registros coletados foram armazenados localmente e consolidados em um dashboard, que apresentou indicadores gerais, distribuição dos feedbacks, erros por classe prevista, matriz de confusão baseada nas avaliações do usuário e registros recentes. Embora a quantidade de feedbacks coletados tenha sido reduzida, os resultados demonstraram que o mecanismo implementado é funcional e pode contribuir para análises futuras do comportamento do modelo em uso interativo.

Os testes funcionais também evidenciaram limitações importantes. O protótipo apresentou melhor funcionamento em condições frontais, com iluminação adequada e enquadramento claro da face. Variações na pose, especialmente rotações laterais do rosto, reduziram a estabilidade das detecções. Além disso, algumas expressões foram mais difíceis de reproduzir e avaliar manualmente, como *disgust*, *contempt*, *fear* e *sad*. Observou-se ainda que a qualidade da câmera influencia o desempenho percebido do sistema, uma vez que um teste complementar em navegador móvel indicou melhor reconhecimento de uma expressão de reprodução mais difícil quando utilizada uma câmera de maior qualidade.

Dessa forma, conclui-se que o trabalho atingiu seu objetivo principal ao unir uma etapa experimental comparativa e uma etapa aplicada de desenvolvimento. A comparação entre modelos permitiu selecionar uma arquitetura adequada para tempo real, enquanto o protótipo demonstrou a viabilidade de integrar o modelo escolhido a uma aplicação web funcional. Assim, a pesquisa contribui tanto para a análise do desempenho de diferentes versões YOLO no reconhecimento de expressões faciais quanto para a demonstração prática de seu uso em um sistema interativo com feedback e visualização de resultados.

## 6.1 OBJETIVOS ALCANÇADOS

Os objetivos definidos para este trabalho foram alcançados ao longo das duas etapas desenvolvidas. Em relação à etapa comparativa, foram implementadas e treinadas quatro versões da arquitetura YOLO, especificamente *YOLOv8n*, *YOLOv10n*, *YOLOv11n* e *YOLOv12n*, utilizando o mesmo protocolo experimental. Os modelos foram avaliados por métricas de desempenho preditivo, como precisão, revocação, *mAP50* e *mAP50-95*, além de métricas de eficiência, como FPS, tempo de inferência, número de parâmetros e GFLOPs.

Também foi realizada a análise qualitativa dos resultados por meio das matrizes de confusão e das métricas por classe, permitindo identificar padrões de acerto e erro entre as expressões faciais avaliadas. Essa análise possibilitou compreender que expressões mais

marcadas tendem a ser reconhecidas com maior estabilidade, enquanto classes mais sutis ou ambíguas apresentam maior propensão a confusões.

A comparação sistemática entre os modelos permitiu selecionar o *YOLOv8n* como a arquitetura mais adequada para continuidade do trabalho, considerando o equilíbrio entre acurácia, velocidade de inferência e viabilidade prática. Embora o *YOLOv12n* tenha apresentado leve vantagem preditiva nos experimentos realizados, o ganho em relação ao *YOLOv8n* foi pequeno diante do maior tempo de inferência. Assim, a escolha do *YOLOv8n* mostrou-se coerente com o objetivo de desenvolver uma aplicação em tempo real.

Na etapa aplicada, o modelo selecionado foi integrado a um protótipo web funcional, composto por *frontend*, *backend*, módulo de inferência e persistência local dos dados. O sistema desenvolvido permitiu capturar imagens por *webcam*, processar os quadros no servidor, executar a inferência com o modelo treinado e exibir as predições ao usuário em tempo real.

Além disso, foi implementado um mecanismo de feedback manual, permitindo registrar avaliações sobre as predições realizadas pelo modelo. Esses registros foram consolidados em um dashboard, possibilitando a visualização de indicadores, gráficos, matriz de confusão baseada nos feedbacks e registros recentes. Com isso, o trabalho também atingiu o objetivo de construir uma ferramenta de apoio à análise do comportamento do modelo em uso interativo.

Portanto, os objetivos específicos relacionados à revisão teórica, exploração do conjunto de dados, treinamento e comparação dos modelos, seleção da arquitetura mais adequada, desenvolvimento do protótipo, implementação da inferência em tempo real, registro de feedbacks e construção do dashboard foram contemplados.

## 6.2 TRABALHOS FUTUROS

A partir das limitações observadas, uma continuidade natural deste trabalho seria ampliar os testes do protótipo com mais usuários e em diferentes condições de uso. Nesta etapa, os testes foram realizados principalmente em ambiente local e com quantidade reduzida de feedbacks, o que limita a interpretação dos resultados obtidos no dashboard. Assim, uma avaliação futura poderia incluir diferentes participantes, variações de iluminação, distâncias em relação à câmera, mudanças de pose e uso de diferentes dispositivos de captura.

Outro ponto importante seria aumentar a quantidade de feedbacks registrados durante o uso do sistema. Com um volume maior de dados, seria possível analisar com mais segurança quais classes apresentam maior tendência de erro no protótipo, especialmente expressões mais sutis ou ambíguas, como *contempt*, *natural*, *sad*, *disgust* e *fear*. Esses re-

gistros também poderiam formar um conjunto de exemplos difíceis, útil para reavaliações posteriores ou para um possível ajuste fino do modelo.

Também podem ser investigadas estratégias para melhorar a robustez do reconhecimento em situações que apresentaram maior dificuldade, como rosto parcialmente de lado, baixa iluminação e expressões de baixa intensidade. Para isso, poderiam ser avaliadas novas técnicas de aumento de dados, melhor balanceamento entre classes, ajustes nos limiares de confiança e treinamento incremental com amostras coletadas durante o uso do protótipo.

Por fim, o sistema pode ser evoluído do ponto de vista da aplicação web. Entre as possibilidades estão a inclusão de autenticação de usuários, armazenamento em banco de dados remoto, implantação em nuvem e suporte a múltiplos usuários. Essas melhorias permitiriam transformar o protótipo desenvolvido em uma aplicação mais próxima de um sistema real de reconhecimento de expressões faciais em tempo real.

## REFERÊNCIAS

- ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: IEEE. **2017 International Conference on Engineering and Technology (ICET)**. 2017. p. 1–6. Disponível em: [⟨https://doi.org/10.1109/ICEngTechnol.2017.8308186⟩](https://doi.org/10.1109/ICEngTechnol.2017.8308186).
- ALI, J.; NAEEM, M. et al. The yolo framework: A comprehensive review of its advances and applications. **Computers**, v. 13, n. 12, p. 281, 2024. Disponível em: [⟨https://www.mdpi.com/2073-431X/13/12/281⟩](https://www.mdpi.com/2073-431X/13/12/281).
- ALSHAMMARI, A.; ALSHAMMARI, M. E. Emotional facial expression detection using YOLOv8. **Engineering, Technology & Applied Science Research**, v. 14, n. 5, p. 16619–16623, 2024.
- EKMAN, P. **An Argument for Basic Emotions**. [S.l.]: Cognition and Emotion, 1992. v. 6. 169–200 p.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.
- JEGHAM, N. et al. Yolo evolution: A comprehensive benchmark and architectural review of yolov12, yolo11, and their previous versions. ResearchGate, February 2025. Preprint. Disponível em: [⟨https://doi.org/10.13140/RG.2.2.15952.83201⟩](https://doi.org/10.13140/RG.2.2.15952.83201).
- JOCHER, G.; CHAURASIA, A.; QIU, J. **Ultralytics YOLOv8**. 2023. Disponível em: [⟨https://github.com/ultralytics/ultralytics⟩](https://github.com/ultralytics/ultralytics).
- JOCHER, G.; QIU, J. **Ultralytics YOLO11**. 2024. Disponível em: [⟨https://github.com/ultralytics/ultralytics⟩](https://github.com/ultralytics/ultralytics).
- KANNA, K. et al. Yolo deep learning algorithm for object detection in agriculture: a review. **Journal of Agricultural Engineering**, v. 55, n. 4, 2024. Disponível em: [⟨https://www.agroengineering.org/jae/article/view/1641⟩](https://www.agroengineering.org/jae/article/view/1641).
- KUCUKAYAN, O.; YAVUZ, M. et al. Yolo-ihd: Improved real-time human detection system for indoor drones. **Sensors**, v. 24, n. 3, p. 922, 2024. Disponível em: [⟨https://www.mdpi.com/1424-8220/24/3/922⟩](https://www.mdpi.com/1424-8220/24/3/922).
- KUMARI, J.; RAJESH, R.; POOJA, K. M. Facial expression recognition: A survey. **Procedia Computer Science**, Elsevier, v. 58, p. 486–491, 2015.
- LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015. Disponível em: [⟨https://doi.org/10.1038/nature14539⟩](https://doi.org/10.1038/nature14539).
- LI, S.; DENG, W. Deep facial expression recognition: A survey. **arXiv preprint arXiv:1804.08348**, 2018.
- LI, W.; ZHANG, H.; CHEN, Y. Research on facial expression recognition based on improved yolov8. In: **Proc. SPIE 13230, Third International Conference on Computer Vision, Image and Deep Learning**. [s.n.], 2024. Disponível em: [⟨https://www.spiedigitallibrary.org/conference-proceedings-of-spie/13230/1323018/⟩](https://www.spiedigitallibrary.org/conference-proceedings-of-spie/13230/1323018/).
- LIAO, J. et al. Facial expression recognition methods in the wild based on fusion feature of attention mechanism and lbp. **Journal of Healthcare Engineering**, v. 2023, p. 1–21, 2023. Disponível em: [⟨https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10180539/⟩](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10180539/).

- LIU, P.; QI, B.; BANERJEE, S. Edgeeye: An edge service framework for real-time intelligent video analytics. In: *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*. New York, NY, USA: Association for Computing Machinery, 2018. (EdgeSys '18), p. 1–6.
- LU, H.; XU, J.; ZHENG, H. Explore the facial expression recognition in different complex environment. In: *Proceedings of CONF-CDS 2025 Symposium: Data Visualization Methods for Evaluation*. [s.n.], 2025. Open access under CC BY 4.0. Disponível em: <https://doi.org/10.54254/2755-2721/2025.PO24884>.
- MAHMOUD, H.; ABOZARIBA, R. A systematic review on webrtc for potential applications and challenges beyond audio video streaming. *Multimedia Tools and Applications*, Springer, v. 84, p. 2909–2946, 2025.
- MATSUMOTO, D. More evidence for the universality of a contempt expression. *Motivation and Emotion*, v. 16, n. 3, p. 363–368, 1992.
- MOLLAHOSSEINI, A.; HASANI, B.; MAHOOR, M. H. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, IEEE, v. 10, n. 1, p. 18–31, 2017.
- NAIM, I. et al. Automated analysis and prediction of job interview performance. *arXiv preprint arXiv:1504.03425*, 2015. Disponível em: <https://arxiv.org/abs/1504.03425>.
- PANTIC, M.; ROTHKRANTZ, L. J. M. Automatic analysis of facial expressions: The state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, n. 12, p. 1424–1445, 2000.
- PINTO-COELHO, L. How artificial intelligence is shaping medical imaging technology: A survey of innovations and applications. *Bioengineering*, v. 10, n. 12, p. 1435, 2023. Disponível em: <https://pmc.ncbi.nlm.nih.gov/articles/PMC10740686/>.
- PLESTED, J.; GEDEON, T. Deep transfer learning for image classification: a survey. *arXiv preprint arXiv:2205.09904*, 2022. Disponível em: <https://arxiv.org/abs/2205.09904>.
- RAWAT, W.; WANG, Z. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, MIT Press, v. 29, n. 9, p. 2352–2449, 2017.
- REDMON, J. et al. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2016.
- SAREEN, V.; SEEJA, K. R. Video-based facial emotion recognition using YOLO and Vision Transformer. In: *EPJ Web of Conferences*. Les Ulis, France: EDP Sciences, 2025. v. 328, p. 01040. Published online: 18 June 2025. Disponível em: <https://doi.org/10.1051/epjconf/202532801040>.
- SASSU, A.; SAENZ-COGOLLO, J. F.; AGELLI, M. Deep-framework: A distributed, scalable, and edge-oriented framework for real-time analysis of video streams. *Sensors*, MDPI, v. 21, n. 12, p. 4045, 2021.
- SHAO, J.; QIAN, Y. Three convolutional neural network models for facial expression recognition in the wild. *Neurocomputing*, Elsevier, v. 355, p. 82–92, 2019.
- TAWARE, S.; THAKARE, A. 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*, v. 11, n. 9s, p. 523–531, 2023. ISSN 2321-8169.

TIAN, Y.; YE, Q.; DOERMANN, D. Yolo12: Attention-centric real-time object detectors. *arXiv preprint arXiv:2502.12524*, 2025.

WANG, A. et al. Yolo10: Real-time end-to-end object detection. *arXiv preprint arXiv:2405.14458*, 2024.

ZHONG, H. et al. Research on real-time teachers' facial expression recognition based on yolov5 and attention mechanisms. *EURASIP Journal on Advances in Signal Processing*, v. 2023, n. 55, p. 1–17, 2023. Disponível em: <https://asp-urasipjournals.springeropen.com/articles/10.1186/s13634-023-01019-w>.