UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Detecção e contagem de pessoas em multidões a partir de imagens aéreas usando a arquitetura YOLO

Ricardo Roberto Storck Pinto

Trabalho de Conclusão de Curso - MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

Ricardo Roberto Storck Pinto

Detecção e contagem de pessoas em multidões a partir de imagens aéreas usando a arquitetura YOLO

Trabalho de conclusão de curso apresentado ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Fernando Pereira dos Santos

Versão original

São Carlos

2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi e Seção Técnica de Informática, ICMC/USP, com os dados fornecidos pelo(a) autor(a)

R488d

Roberto Storck Pinto. Ricardo
Detecção e contagem de pessoas em multidões a partir de imagens aéreas usando a arquitetura YOLO / Ricardo Roberto Storck Pinto; orientador Fernando Pereira dos Santos. – São Carlos, 2024.
74 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

Ricardo Roberto Storck Pinto

Detection and counting of people in crowds based on aerial images using the YOLO architecture

Thesis presented to the Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: Artificial Intelligence

Advisor: Prof. Fernando Pereira dos Santos

Original version

São Carlos

2024

Este trabalho é dedicado aos alunos da USP, como uma contribuição das Bibliotecas do Campus USP de São Carlos para o desenvolvimento e disseminação da pesquisa científica da Universidade.

AGRADECIMENTOS

Agradeço a Deus por me proporcionar saúde, paz e segurança, essenciais para superar os desafios e alcançar os meus objetivos.

Agradeço à minha esposa Renata, minhas filhas Helena e Eloah. São a minha fonte inesgotável de amor e inspiração.

Agradeço aos meus familiares e amigos, cujas experiências de vida, tanto positivas quanto negativas, me inspiram a buscar o aprimoramento constante.

Agradeço à USP e aos excelentes Professores deste curso por compartilharem seus valiosos conhecimentos.

"... a dificuldade pode ser uma fonte de forças. Nas montanhas, só tensionamos os músculos nas passagens difíceis; as sendas planas deixam-nos relaxados e a frouxidão desatenta logo se mostra funesta. O que vale mais do que tudo é o querer, um querer profundo: querer ser alguém, chegar a algum lugar; ser já, pelo desejo, esse alguém qualificado por seu ideal."

Antonin-Dalmace Sertillanges - A Vida Intelectual

RESUMO

Roberto Storck Pinto. Ricardo; **Detecção e contagem de pessoas em multidões a partir de imagens aéreas usando a arquitetura YOLO**. 2024. 74 p. Trabalho de Conclusão de Curso (MBA em Inteligência Artificial e Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

Este trabalho de conclusão de curso teve como objetivo utilizar a arquitetura YOLOv8 para desenvolver e avaliar um sistema de detecção e contagem de pessoas em imagens aéreas, visando otimizar a precisão e a generalização do modelo para diferentes cenários. A automatização da contagem de multidões (Crowd Counting) se mostra cada vez mais relevante para a administração pública, especialmente no que tange ao planejamento urbano, segurança, gestão de eventos e planejamento estratégico.

O método utilizado consiste em implementar e treinar um modelo de detecção de objetos e contagem baseado na arquitetura YOLO, ajustando os hiperparâmetros para otimizar o desempenho. A capacidade de generalização do modelo foi avaliada em imagens com diferentes condições de iluminação, resolução e densidade de pessoas. A precisão do modelo foi quantificada utilizando métricas relevantes, como precisão, recall e mAP50.

Os resultados obtidos, principalmente no Experimento 5, demonstraram a importância de ajustar o tamanho das imagens de entrada e fornecer uma única classe para o modelo identificar. No entanto, a contagem final de pessoas no conjunto de teste evidencia a necessidade de aprimoramentos.

Conclui-se que a aplicação da arquitetura YOLO para contagem de pessoas em imagens aéreas apresenta grande potencial. Contudo, é necessário realizar novos experimentos para obter os hiperparâmetros ideais, bem como avaliar o custo computacional de treinamento ao utilizar imagens de entrada maiores e utilizar outras arquiteturas presentes no YOLOv8 mais robustas, como YOLOv8l ou YOLOv8x.

Palavras-chave: YOLO. Contagem de Multidões (*Crowd Counting*). Visão Computacional. Detecção de Objetos. Imagens Aéreas.

ABSTRACT

Roberto Storck Pinto. Ricardo; **Detection and counting of people in crowds based on aerial images using the YOLO architecture**. 2024. 74 p. Thesis (MBA in Artificial Intelligence and Big Data) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

This thesis aimed to utilize the YOLOv8 architecture to develop and evaluate a system for detection and counting people in aerial images, aiming to optimize the accuracy and generalization of the model for different scenarios. Automating crowd counting is increasingly relevant for public administration, especially in urban planning, security, event management, and strategic planning.

The method employed consisted of implementing and training an object detection and counting model based on the YOLO architecture, adjusting hyperparameters to optimize performance. The generalization ability of the model was evaluated on images with different lighting conditions, resolutions, and crowd densities. The model's accuracy was quantified using relevant metrics such as precision, recall, and mAP50.

The results obtained, mainly in Experiment 5, demonstrated the importance of adjusting the input image size and providing a single class for the model to identify. However, the final people count on the test set highlights the need for further improvements.

It is concluded that applying the YOLO architecture for people counting in aerial images presents great potential. However, further experiments are needed to obtain the ideal hyperparameters, as well as to evaluate the computational cost of training when using larger input images and utilizing other more robust architectures present in YOLOv8, such as YOLOv8l or YOLOv8x.

Keywords: YOLO. Crowd Counting. Computer Vision. Object Detection. Aerial Images.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

YOLO	You Only Look Once
MLP	Multilayer Perceptron
CNN	Convolutional Neural Networks
RGB	Red, Green, Blue
HSV	Hue-Saturation-Value
RNA	Redes Neurais Artificiais
PLN	Processamento de Linguagem Natural
ReLU	Rectified Linear Unit
UAV	Unmanned Aerial Vehicle
IOU	Intersection Over Union
OBB	Oriented Bounding Boxes
ROI	Region of Interest
CCTV	Closed Circuit Television

SUMÁRIO

1 INTRODUÇÃO

A tarefa de estimar o número de pessoas em uma imagem, conhecida pelo termo *Crowd Counting* (Contagem de Multidões), é muito importante para a administração pública de uma cidade pois permite que os administradores públicos tenham condições de estimar, com razoável precisão, o número de participantes em determinados eventos ou manifestações nos espaços públicos da cidade. Permite que eventos sazonais, como por exemplo eventos de Carnaval, possam ter estimativas de público razoáveis registradas a cada ano, em cada um dos espaços públicos utilizados e também registrar o número de participantes em eventos não programados, como manifestações.

Os benefícios de obter uma contagem razoavelmente precisa dos números de participantes traz impactos positivos sobre temas importantes para a administração pública de uma cidade, como: planejamento urbano, segurança e monitoramento do uso de espaços públicos, planejamento de obras ou controle do trânsito, gestão de eventos, planejamento estratégico da cidade.

Contar o número de pessoas em uma imagem é uma tarefa de detecção e contagem de objetos, sendo um dos principais desafios em visão computacional. É um desafio quando se tem uma alta presença de pessoas em uma imagem, o que seria um caso de alta densidade ou devido a oclusão de indivíduos na imagem (pessoas na imagem obstruídas por outras pessoas ou objetos presentes na área).

Abordar essa tarefa utilizando visão computacional é uma abordagem moderna, permitindo ter respostas mais rápidas e precisas e com menor impacto de possíveis vieses de contagem (vieses muito comuns quando se tem o confronto de números obtidos pelos meios de comunicação versus dados oficiais de segurança pública) e erros humanos de contagem.

Uma abordagem tradicional para estimar o número de pessoas em uma multidão é o chamado Método de Jacobs (Jacobs, 1967). Este método foi desenvolvido por Herbert Jacobs e consiste em dividir a área em seções menores e encontrar o número médio de pessoas em cada seção, depois calcular a densidade média e extrapolar esses resultados para toda a região ocupada.

A abordagem moderna, utilizando visão computacional, é para obter o resultado da contagem em menor tempo e com maior acurácia.

1.1 Objetivos

Este estudo teve como objetivo principal desenvolver e avaliar um sistema de contagem de pessoas em imagens aéreas utilizando a arquitetura YOLO, com foco em

otimizar a precisão e a generalização do modelo para diferentes cenários. Para isso, os seguintes objetivos específicos foram propostos:

- Implementar e treinar um modelo de detecção e contagem de objetos baseado em YOLO, ajustando os hiperparâmetros para otimizar o desempenho;

- Avaliar a capacidade de generalização do modelo em imagens com diferentes condições de iluminação, resolução e densidade de pessoas;

- Quantificar a precisão do modelo utilizando métricas relevantes, como precisão, recall e mAP50;

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Princípios de Processamento de Imagens

O processamento de imagens é uma técnica para realizar operações em imagens, com a finalidade de melhorá-las ou extrair informações úteis, analisar e tomar decisões a partir delas. Para isso, cálculos podem ser feitos, quantificando as informações existentes nas imagens.

O processamento de imagens é um subconjunto da visão computacional que utiliza diversas técnicas para manipular imagens digitais e obter imagens melhoradas ou extrair informações importantes de cada imagem (Ansari, 2023).

Processar imagens digitais é uma tarefa considerada a base da visão computacional (Sá, 2021). Existem diversas aplicações para esse uso, como: análise de imagens médicas, inteligência artificial, restauração de imagens, vigilância eletrônica, robótica, veículos autônomos, contagem de objetos, identificação de pessoas, etc.

O objetivo do processamento de imagem pode ser dividido em cinco grupos:

- Visualização: observar objetos que não estão visíveis;
- Nitidez e restauração de imagens: para melhorar a qualidade da imagem;
- Recuperação de imagens: facilitar a busca de uma imagem de interesse;
- Medição de padrão: obter medidas de vários objetos;
- Reconhecimento de imagem: distinguir objetos em uma imagem.

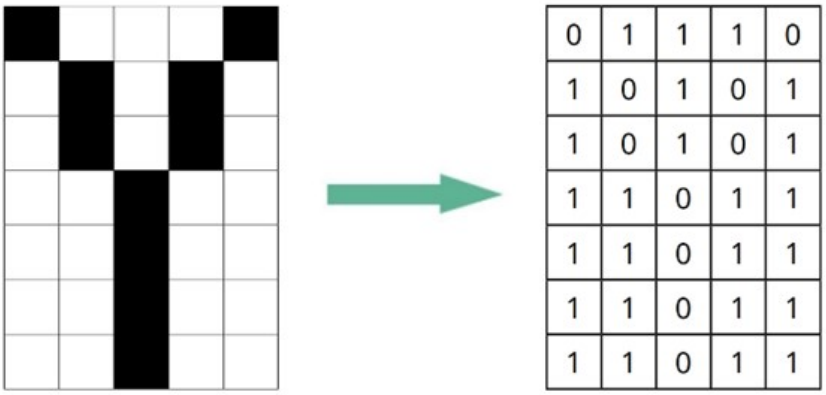
Um computador interpreta uma imagem como uma série de números. Uma imagem digital é uma matriz de pixels quadrados. Cada pixel é um elemento da imagem dispostos em colunas e linhas. Isso é o mesmo que uma matriz bidimensional. “As imagens são uma composição de cores e intensidade que representam uma cena, que pode ser interpretada pelo olho humano”(Sá, 2021), conforme a Figura 1 abaixo:

Figura 1 – Exemplo de uma imagem

Fonte: scikit-image

Imagens digitais podem ser representadas em uma matriz numérica conforme demonstrado na Figura 2:

Figura 2 – Imagem representada em uma matriz numérica

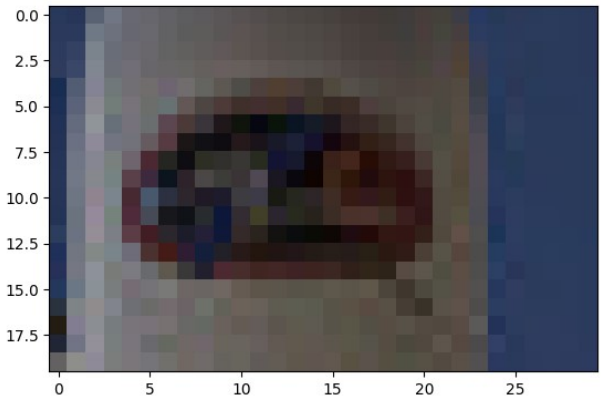


Fonte: (Sá, 2021)

A representação na Figura 2 é de uma imagem considerada como binária, pois a cor preta é representada numericamente como 0 e a cor branca é representada numericamente como 1.

Aplicando um zoom na Figura 1 é possível observar os pixels que formam a imagem.

Figura 3 – Pixels presentes em uma imagem



Fonte: elaborado pelo autor

Cada pixel contém informações sobre cor e intensidade. As imagens coloridas são armazenadas em matrizes tridimensionais. É possível verificar isso através do atributo 'shape' da biblioteca pandas. Para a Figura 1 o resultado será (427, 640, 3). O primeiro valor corresponde à altura, o segundo à largura e o terceiro é a dimensão (em que 3 significa três canais de cores).

Para obter o número total de pixels na imagem, basta utilizar o método 'size' da biblioteca pandas. Para a Figura 1 o resultado será: 819840.

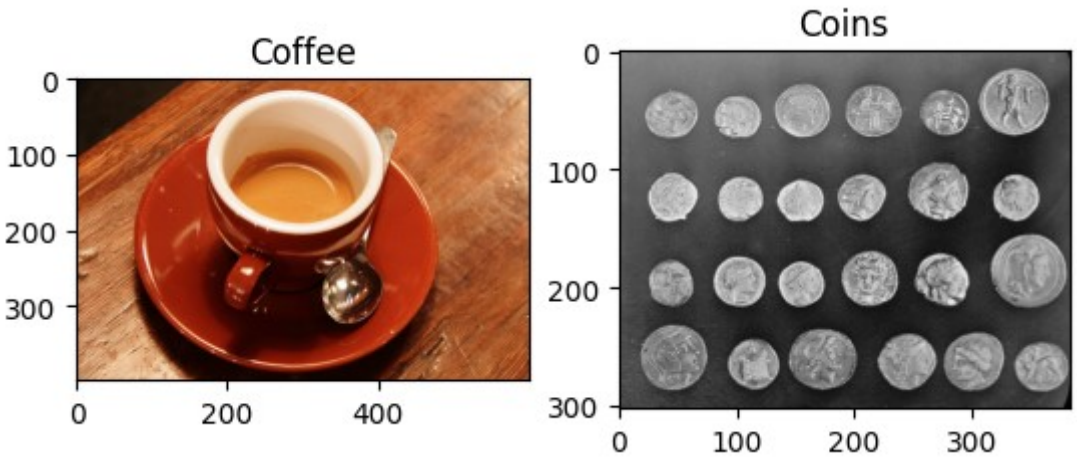
Outa forma de obter esse valor é multiplicar a altura (427) x largura (640) x canais de cores (3) = 819.840 pixels.

Imagens coloridas tridimensionais são frequentemente representadas em RGB (Red, Green, Blue), sendo 3 matrizes bidimensionais em que as três camadas representam os canais vermelho, verde e azul da imagem.

As imagens em cinza têm apenas tons de preto e branco. A intensidade da escala de cinza é armazenada como um número inteiro de 8 bits, fornecendo 256 tons diferentes de cinza possíveis. Não possuem nenhuma informação de cor.

Para mostrar a diferença entre imagens com canais RGB e imagens em tons de cinza, serão utilizadas as imagens da Figura 4:

Figura 4 – Imagem com canais RGB e imagem em tons de cinza



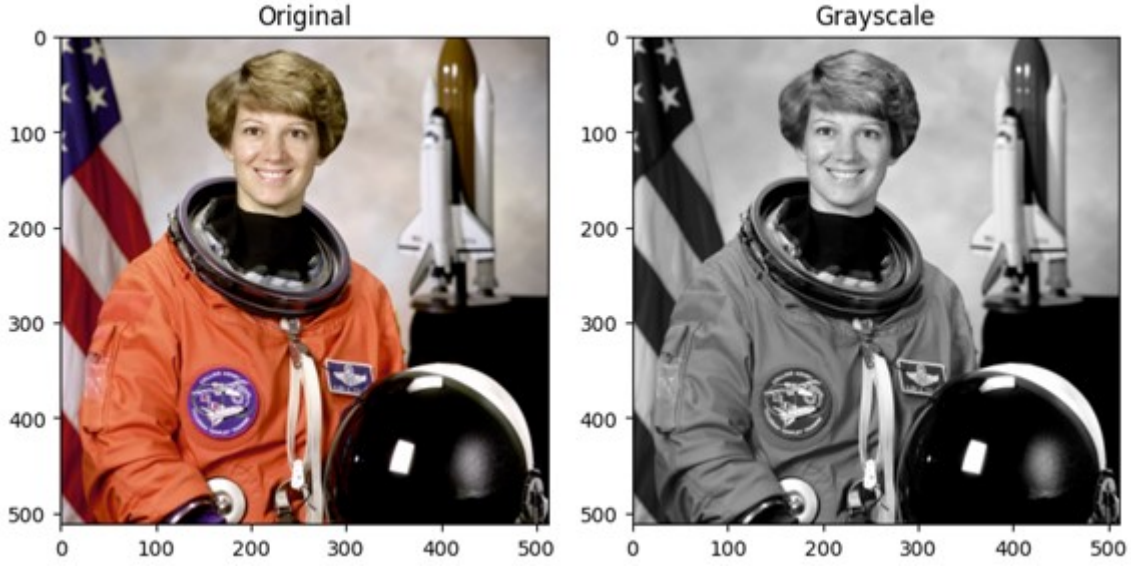
Fonte: scikit-image

A imagem da esquerda da Figura 4, em cores (RGB-3) tem o formato de (400, 600, 3), com 3 canais de cores. Já a imagem da direita da Figura 4 tem o formato de (303, 384), em tons de cinza com um único canal de cor.

Em alguns casos é necessário converter uma imagem com canais RGB para tons de cinza. Isso precisa ser feito porque em aplicações de processamento de imagens, as informações de cores podem não ajudar a identificar partes da imagem, como contornos e características.

A Figura 5 foi convertida do seu formato original, RGB, para uma escala de cinza:

Figura 5 – Conversão de imagem RGB para tons de cinza



Fonte: scikit-image

Uma outra forma de conversão de uma imagem RGB é para o modelo HSV (Hue-Saturation-Value). De acordo com (Sá, 2021), este modelo é adequado quando envolve

manipulações diretas, como a descrição de cores próximas, já que é um sistema que permite manipular os canais de Matiz e Saturação.

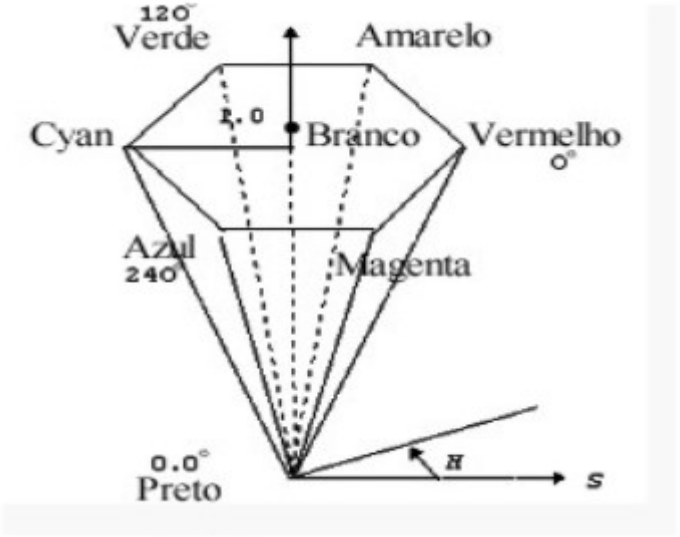
A Matiz (Hue) representa o tipo de cor e abrange todas as cores do espectro, desde a cor vermelha até a cor violeta, incluindo também o magenta (Conci,). O valor da Matiz varia de 0 a 360 graus.

A Saturação (Saturation) é conhecida como pureza da cor, determinando o quão "pura" a cor pode ser. Imagens em tons de cinza possuem baixa saturação. Esta pode variar de 0 a 100.

E o Valor (Value) é o brilho da cor, podendo variar de 0 a 100.

Na Figura 6 é possível compreender como o formato HSV é representado:

Figura 6 – Representação gráfica do Modelo HSV



Fonte: http://profs.ic.uff.br/~aconci/modeloHSV.html

E na Figura 7 é possível ver os canais de Matiz (Hue) e Valor (Value) para uma imagem.

Figura 7 – Conversão de imagem RGB para HSV

Fonte: scikit-image

2.2 Básico de processamento de imagens com Numpy

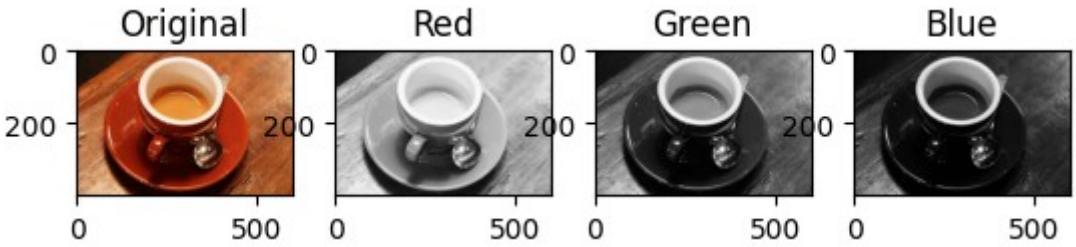
Alguns processamentos básicos de imagens consistem em converter a imagem colorida para tons de cinza, aplicar filtros para detecção de bordas, redimensionar o tamanho da imagem, melhorar a nitidez, brilho ou contraste, extrair características importantes e outras.

As imagens podem ser originadas de diversas fontes e por isso podem conter tamanhos diferentes, bem como caracaterísticas como o brilho, contraste, intensidade de cores, etc, que podem exigir um processamento para equalizar as imagens obtidas.

Essas transformações aplicadas nas imagens podem ser necessárias quando a resolução de um problema específico requer o uso da visão computacional (Ansari, 2023).

Uma imagem é reconhecida pelo Numpy como um objeto numpy.ndarray. Isso significa que a manipulação de matrizes do Numpy pode ser utilizada para as imagens. É possível por exemplo fatiar a matriz multidimensional e obter as intensidades de cores individuais ao longo de uma imagem. Isso pode ser feito obtendo os valores de cada RGB na imagem, conforme demonstrado na Figura 8.

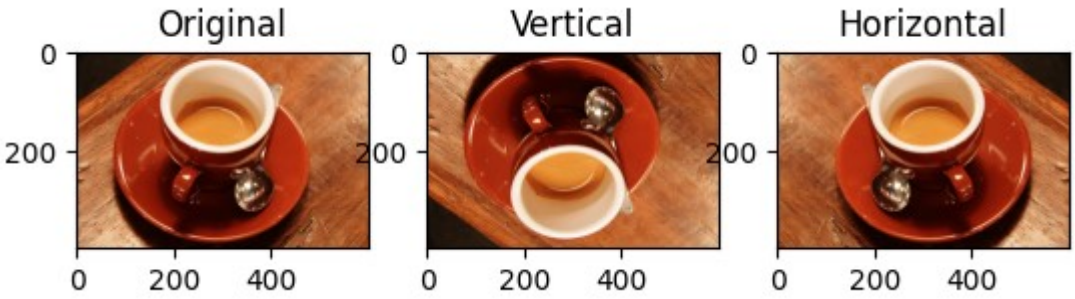
Figura 8 – Intensidade de cores individuais



Fonte: scikit-image com alterações do autor

Utilizando o Numpy, também é possível girar a imagem, conforme demonstrado na Figura 9:

Figura 9 – Imagem girada verticalmente e horizontalmente

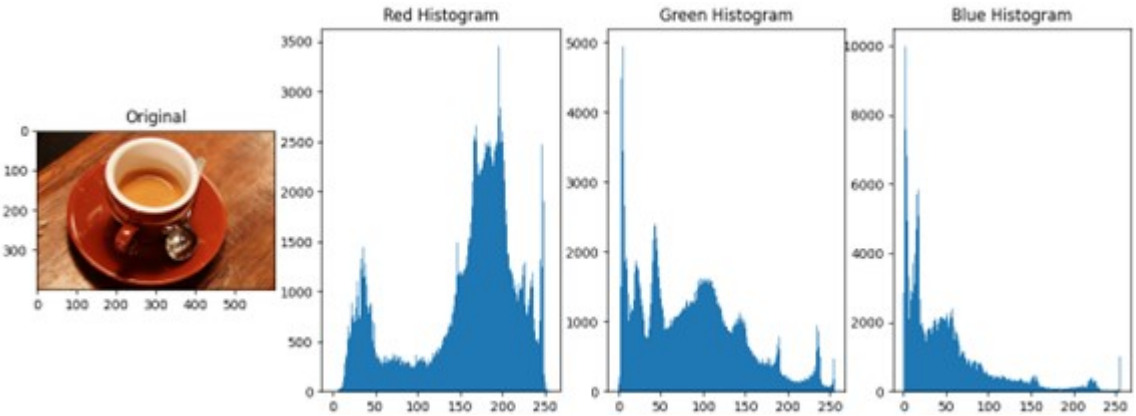


Fonte: scikit-image com alterações do autor

Para gerar um histograma da imagem, que é uma representação gráfica da quantidade de pixels de cada valor de intensidade, em que 0 é preto puro e 255 é branco puro, pode-se utilizar a biblioteca matplotlib. Este método precisa de uma matriz de entrada (que será a frequência) e a quantidade de bins como parâmetros.

O método '*ravel()*' da biblioteca Numpy pode ser usado para retornar uma matriz plana contínua dos valores de cor da imagem. Os bins são definidos como 256 para mostrar o número de pixels para cada valor, entre 0 e 255. Isso é demonstrado na Figura 10.

Figura 10 – Histograma para cada camada de cor



Fonte: scikit-image com alterações do autor

Os histogramas são usados para fazer análises de imagens, para definir quais imagens serão utilizadas, para equalizar imagens que estão sendo utilizadas igualando o nível de brilho e contraste entre as imagens.

2.3 Aplicando *Thresholding* em imagens

O *Thresholding* (limiar) é usado para separar o plano de fundo e o primeiro plano de uma imagem em tons de cinza, tornando ambos em preto e branco.

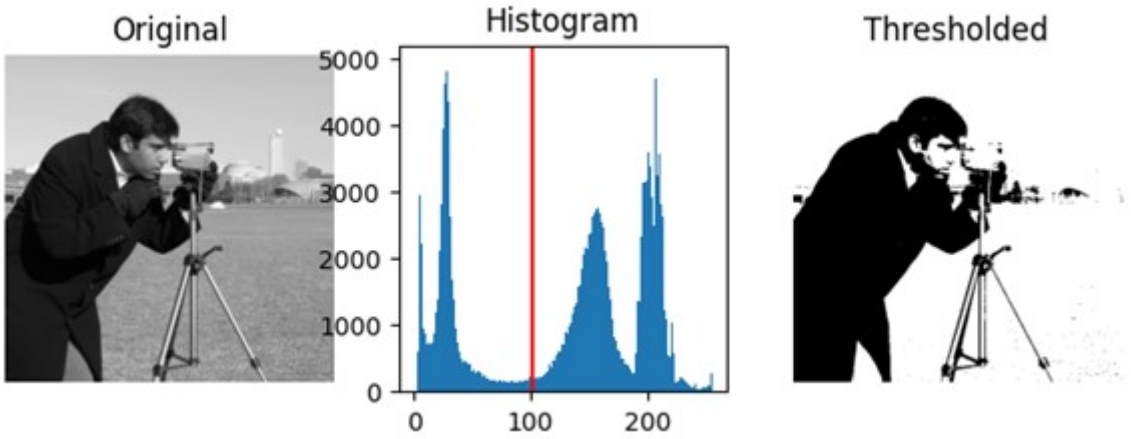
Cada pixel na imagem é comparado com um limite definido previamente. Se o pixel for menor que o valor definido, o pixel se tornará branco. Caso contrário, o pixel se tornará preto.

Esta técnica de *thresholding* é o método mais simples de segmentação de imagem. Ela permite isolar elementos na imagem e é usada para detecção de objetos, reconhecimento facial e outras aplicações.

A técnica será demonstrada através da utilização do Método de Otsu (Gazziro, 2013). Este método é aplicado para realizar a segmentação automática de imagens. Trata-se de um algoritmo que "retorna um limiar de intensidade único que separa os pixels em duas classes: primeiro plano (*foreground*) e fundo (*background*)"(Gazziro, 2013).

Este método de Otsu é muito importante para determinar de forma automática um limiar de intensidade nas imagens, sendo algo que facilita executar a separação de objetos do primeiro plano e do fundo, conforme a Figura 11.

Figura 11 – Uso do Método de Otsu para separar os planos de uma imagem



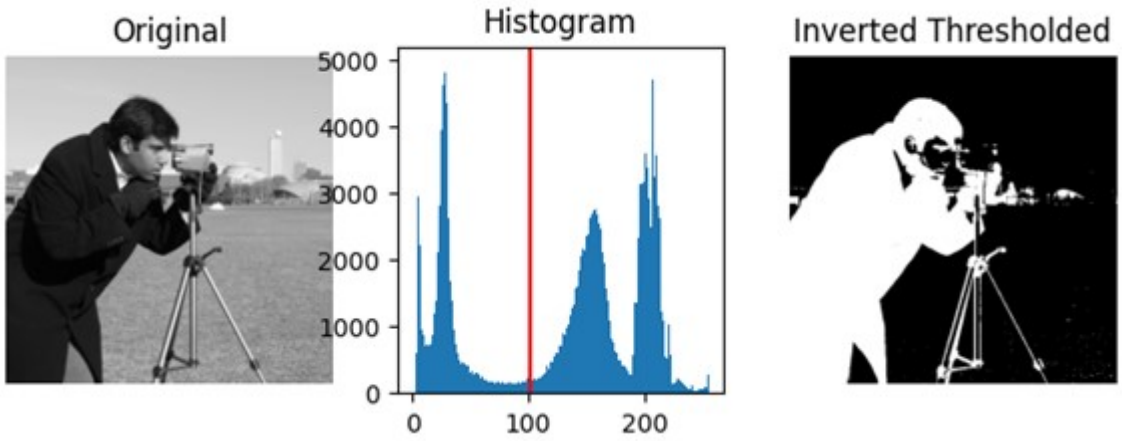
Fonte: scikit-image

A imagem utilizada é binarizada, ou seja, cada pixel é atribuído a uma das duas classes (primeiro plano ou fundo) com base no limiar calculado.

O método de Otsu calcula um limiar "ótimo", que é identificado pela linha vermelha no histograma da imagem acima. Essa linha vermelha é o limiar "ótimo" que maximiza a variância entre as duas classes de pixels, separadas pelo limiar. Ou, dito de outra forma, é o limiar "ótimo" que minimiza a variância intraclasse.

Uma outra abordagem é realizar um limiar invertido, cujo resultado será o primeiro plano da imagem na cor branca e o segundo plano na cor escura, conforme Figura 12.

Figura 12 – Uso do Threshold invertido

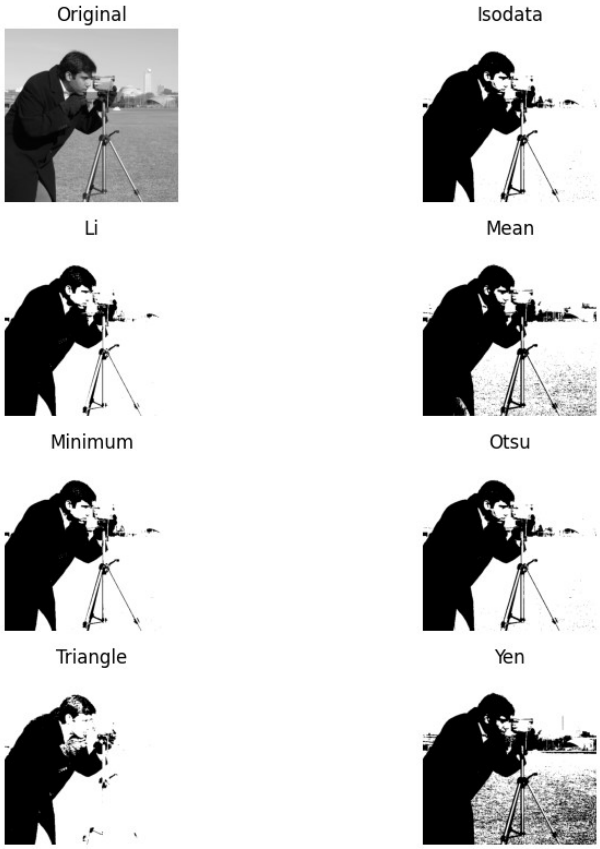


Fonte: scikit-image com alterações do autor

Existem outros métodos que efetuam essa separação do primeiro plano e do segundo plano de imagem, possibilitando a comparação e avaliação de qual método apresenta o melhor resultado.

Isso pode ser visto na Figura 13 que apresenta diversos métodos diferentes.

Figura 13 – Diferentes algoritmos de limiar



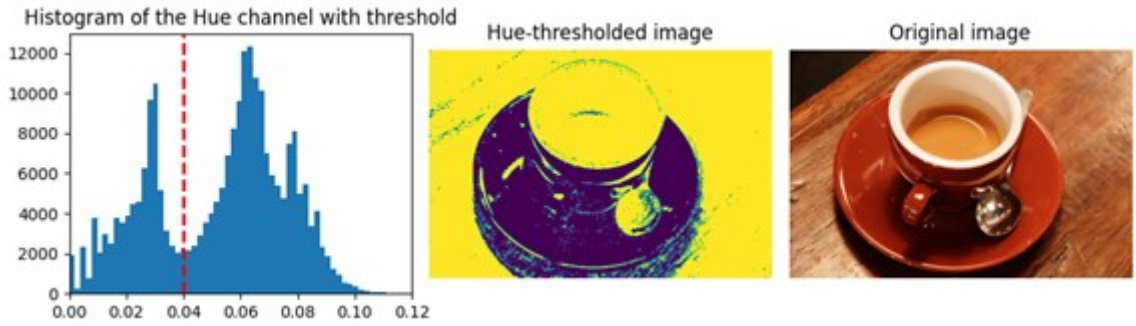
Fonte: scikit-image com alterações do autor

Como mostrado na Figura 11, o Método de Otsu se encarrega de calcular um limiar

ótimo para a separação. Mas este limiar pode ser definido manualmente. Por exemplo, para a Figura 7 foi definido um limiar no valor de 0.04 para o canal de Matiz (hue) e este limiar foi utilizado para binarizar a imagem, separando a imagem do copo do fundo.

Este exemplo pode ser visto na Figura 14:

Figura 14 – Definição manual do Threshold



Fonte: scikit-image com alterações do autor

Também pode ser definido um segundo limiar, desta vez para o canal de Valor (value), atuando juntamente com o canal de Matiz, para retirar parcialmente a sombra do copo, conforme a Figura 15:

Figura 15 – Ajuste manual do limiar para os canais Hue e Value

Fonte: scikit-image com alterações do autor

2.4 Redes Neurais Artificiais - RNA

As Redes Neurais Artificiais foram inspiradas no funcionamento do cérebro humano. Uma RNA é "um modelo de aprendizado de máquina inspirado em nossas redes neurais cerebrais"(Géron, 2021).

Podem ser utilizadas para diversas tarefas, como processamento de imagens, sistemas de recomendação, processamento de linguagem natural e tarefas comuns do aprendizado de máquina como a previsão de valores númericos e classificação.

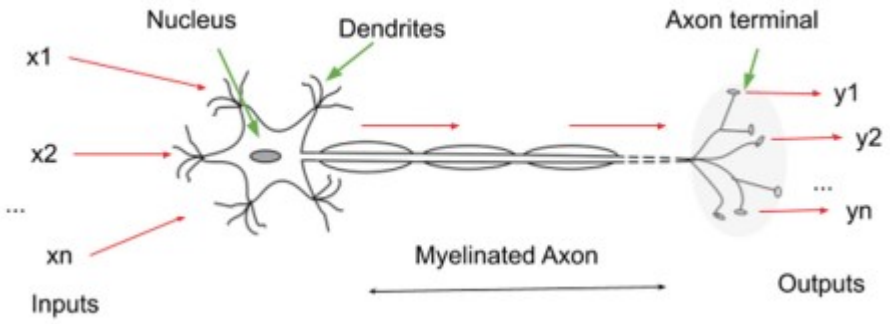
Segundo (Géron, 2021) citando o artigo de (McCulloch; Pitts, 1943), as Redes Neurais Artificiais foram apresentadas pela primeira vez em 1943. Neste artigo os autores apresentaram o que é considerado a primeira arquitetura de uma rede neural artificial.

Uma outra definição para o que é uma Rede Neural Artificial é a dada por (Ansari, 2023), em que uma RNA é um modelo computacional inspirado pela estrutura e o funcionamento das redes neurais biológicas do cérebro humano.

Uma rede neural biológica é aquela composta dos neurônios biológicos. São células presentes principalmente no cérebro dos animais, em quantidades estimadas de bilhões de células e são responsáveis pelas conexões cerebrais.

A Figura 16, presente em (Ansari, 2023), mostra a estrutura de um neurônio biológico:

Figura 16 – Neurônio Biológico



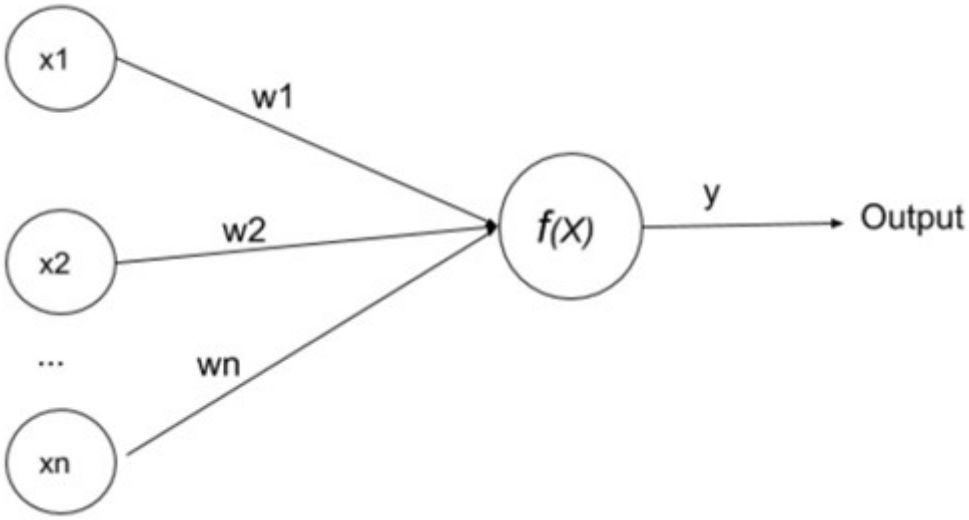
Fonte: (Ansari, 2023)

Analisando a Figura 16, é possível observar as diferentes partes que compõem um neurônio biológico, como:

- Corpo Celular: contém o núcleo do neurônio e é dele que partem os prolongamentos;
- Dendritos: são os prolongamentos ramificados. Existem vários deles no núcleo. São responsáveis por receber estímulos e sinais de outros neurônios;
- Axônio: é um prolongamento responsável por conduzir os impulsos nervosos para outras células, como glândulas e outros neurônios;
- Bainha de Mielina: é uma camada isolante responsável por envolver o axônio. É composta por cadamadas de proteínas e lipídios. A mielina é a responsável por aumentar a velocidade de condução do impulso nervoso;

Um neurônio artificial pode ser representado conforme a Figura 17, extraída de (Ansari, 2023).

Figura 17 – Representação de um Neurônio Artificial



Fonte: (Ansari, 2023)

Na Figura 17, x1, x2, e xn são as entradas. Os pesos são w1, w2, wn e são associados com cada entrada fornecida. As entradas são processadas utilizando funções matemáticas para gerar as saídas. O neurônio é definido como f(X). As funções matemáticas são chamadas de funções de ativação. E por último a variável y é a saída gerada pelo neurônio (Ansari, 2023).

2.5 Perceptron

É uma das arquiteturas mais simples de Rede Neural Artificial. A representação dele é a demonstrada na Figura 17.

A arquitetura do Perceptron foi criada em 1957 por Frank Rosenblatt (Rosenblatt, 1958). O neurônio recebe os valores de entrada (sinais) em forma de números e cada entrada é associada com um peso. A tarefa do Perceptron é determinar os pesos ideais para cada sinal de entrada.

A equação matemática básica que demonstra o funcionamento de um perceptron é a definida abaixo:

$$Z = W_1X_1 + W_2X_2 + ... + W_nX_n = X^TW$$

Fonte: (Géron, 2021)

Essa função realiza uma soma ponderada de suas entradas, aplica uma função degrau para a soma e gera o resultado conforme mostrado a equação abaixo extraída de (Géron, 2021):

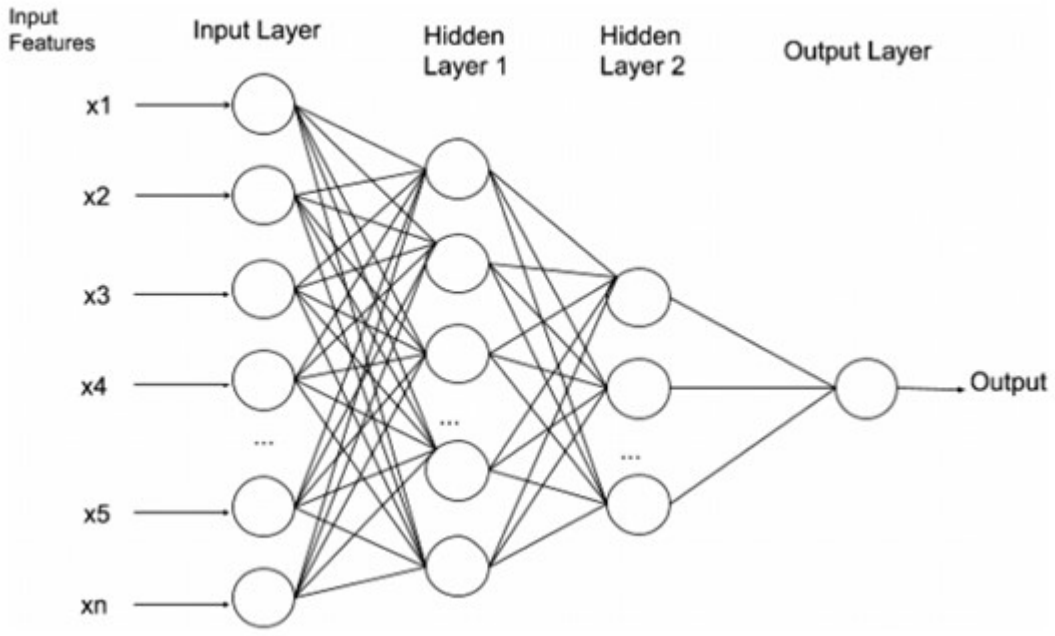
$$h_w(X) = degrau(Z), \text{ sendo que } z = X^t W$$

Fonte: (Géron, 2021)

Como demonstrado na Figura 17, o Perceptron é composto por um único neurônico conectado a todas as entradas.

Outro tipo de rede é o Perceptron Multicamadas (MLP), conforme Figura 18:

Figura 18 – Perceptron multicamadas



Fonte: (Ansari, 2023)

Nesse tipo de rede, existe a camada de entrada, que recebe os valores de entrada. Cada camada de entrada se conecta a uma ou mais camadas ocultas e existe uma camada final de saída.

No exemplo da Figura 18 é demonstrada uma arquitetura conhecida por Rede Neural *Feedforward*. O sinal de entrada percorre todas as camadas em direção à última camada, a de saída.

Quanto se tem uma arquitetura de Rede Neural Artificial composta de várias camadas ocultas, é dada a denominação de Rede Neural Profunda.

2.6 Redes Neurais Convolucionais

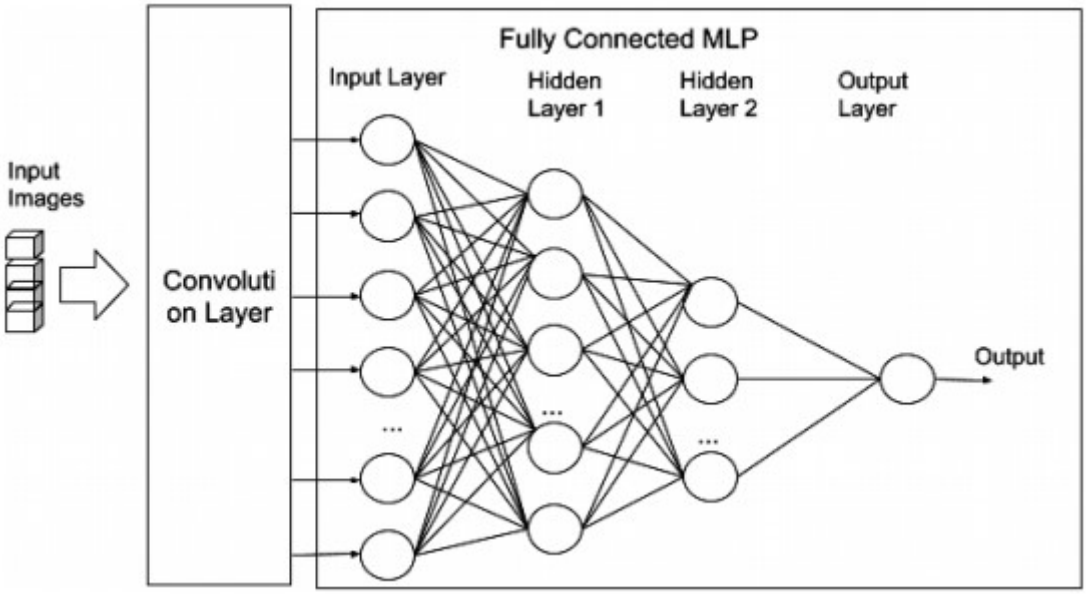
As Redes Neurais Convolucionais (CNNs) trouxeram muitos avanços em visão computacional e são utilizadas desde a década de 1980. Mas não são utilizadas apenas em visão computacional, também podem ser utilizadas, por exemplo, em tarefas de

reconhecimento de voz, processamento de linguagem natural (PLN) e processamento de sinais. Redes Convolucionais "são simplesmente redes neurais que usam convolução em vez de multiplicação de matriz geral em pelo menos uma de suas camadas" (Goodfellow; Bengio; Courville, 2016).

O principal elemento de uma CNN é a camada convolucional. O termo convolução é uma operação matemática simples que preserva as relações espaciais. Nas imagens, a convolução pode detectar áreas de interesse relevantes, como bordas, cantos, linhas verticais, etc.

Outra característica de destaque de uma CNN é a sua capacidade de extrair e selecionar automaticamente as principais características de uma imagem. A arquitetura de uma CNN, para processamento de imagens, é mostrada conforme Figura 19:

Figura 19 – Rede Neural Convolucional e MLP



Fonte: (Ansari, 2023)

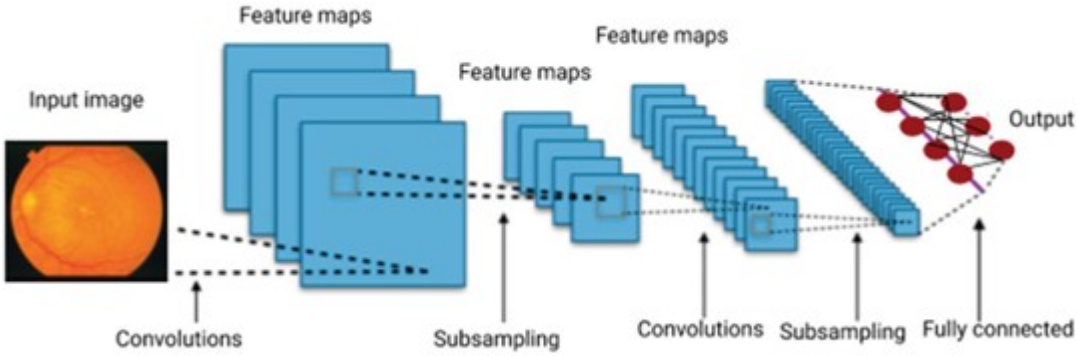
A Figura 19 é um Perceptron Multicamadas, representado anteriormente através da Figura 18, que contém uma camada anterior chamada de Camada de Convolução.

As camadas convolucionais de uma CNN são aplicadas em pequenas regiões dos dados de entrada (Ansari, 2023). As imagens de entrada são passadas para a primeira camada convolucional da rede. A camada convolucional aplica diversos filtros em pequenas regiões da imagem e produz o mapa de características. Cada filtro aplicado consegue capturar padrões específicos presentes na imagem, como bordas, linhas, contornos.

Além das camadas convolucionais, também há a camada de subsampling, que é o processo em que ocorre a redução da resolução ou do tamanho da imagem.

Um exemplo de uma CNN completa pode ser visto na Figura 20:

Figura 20 – Rede Neural Convolucional Completa



Fonte: (Ansari, 2023)

Na Figura 20 é possível ver uma CNN completa, com camadas convolucionais, camadas de subsampling e nós totalmente conectados.

CNNs têm sido utilizadas amplamente para tarefas de contagem de multidões, superando outros métodos tradicionais (Khan; Menouar; Hamila, 2023). Uma CNN frequentemente emprega a técnica de estimar a densidade da multidão, gerando um mapa de densidade. Esse mapa é utilizado para estimar o número de pessoas em uma multidão.

Redes Neurais Convolucionais são consideradas uma arquitetura de rede profunda, exigindo grandes quantidades de imagens para o treinamento e podendo ter camadas com milhares ou bilhões de neurônios (Carvalho; Menezes; Bonidia, 2024).

2.7 Arquitetura YOLO

YOLO (*You Only Look Once* - Você só olha uma vez) é um modelo de detecção de objetos e segmentação de imagens. Foi desenvolvido por Joseph Redmon, Santosh Divvala, Ross Girshick e Ali Farhadi e lançado em 2015 (Redmon *et al.*, 2016).

É uma arquitetura de rede neural para visão computacional considerada como o estado da arte neste domínio. Trata-se de um algoritmo para identificação de objetos que usa uma técnica de passada única (*single pass*), extraindo as características através do uso de uma rede neural convolucional. O acrônimo para o nome é devido a essa técnica de passada única.

Atualmente o modelo está em sua décima primeira versão (YOLO11) e para este estudo será utilizada a sua oitava versão (YOLOv8) (Jocher; Munawar; AyushExel, 2024), ainda considerada ideal para tarefas de detecção de objetos.

YOLO utiliza uma rede neural convolucional na imagem inteira, dividindo a imagem em regiões e deduzindo seus limites e probabilidades para cada região, ocorrendo a detecção e o cálculo das probabilidades (Sá, 2021).

A Figura 21 é um exemplo da detecção de cada objeto em uma imagem e o nível de probabilidade, confiança definido pelo modelo.

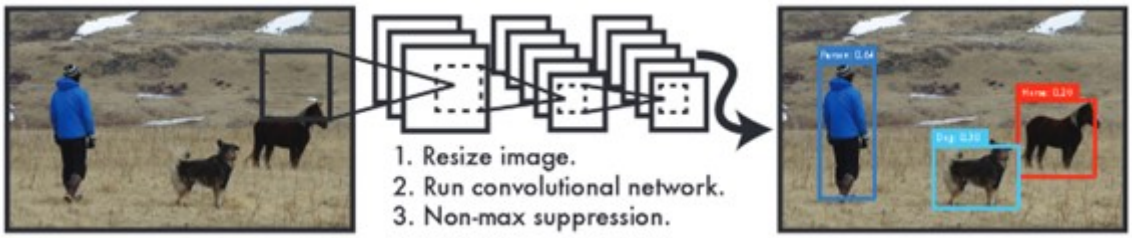
Figura 21 – Detecção de objetos com YOLO

Fonte: (Jocher; Munawar; AyushExel, 2024)

O modelo realiza um ajuste no tamanho da imagem de entrada para 448 x 448 e roda uma única rede neural convolucional na imagem e limita as detecções resultantes pela confiança do modelo (Redmon *et al.*, 2016).

Um exemplo simplicado das etapas pode ser visto na Figura 22:

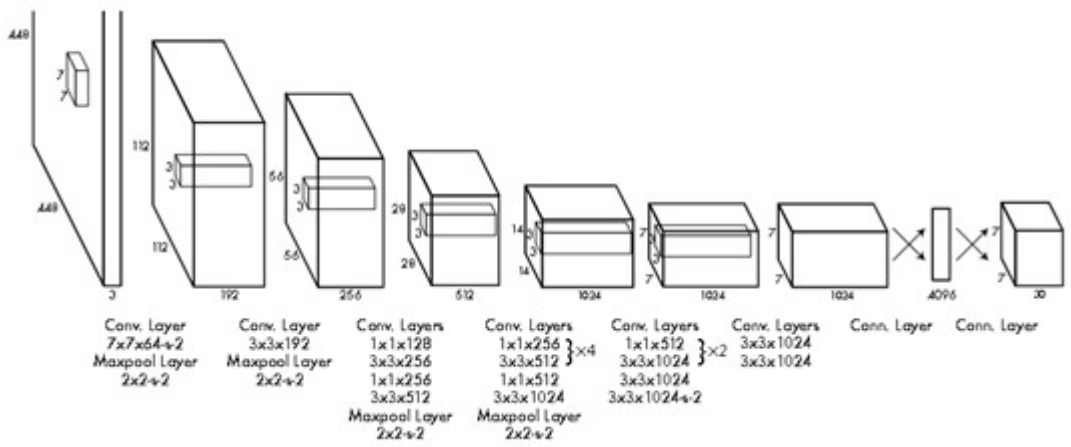
Figura 22 – Etapas do YOLO



Fonte: (Redmon *et al.*, 2016)

A arquitetura do YOLO é composta de 24 camadas convolucionais, 4 camadas de *max-pooling* e 2 camadas totalmente conectadas. Conforme é demonstrado na Figura 23:

Figura 23 – Arquitetura do YOLO



Fonte: (Redmon *et al.*, 2016)

Na arquitetura demonstrada na Figura 23 é utilizada a função de ativação ReLU, exceto para a última camada que utiliza uma função de ativação linear. ReLU significa *Rectified Linear Unit* (Unidade Linear Retificada). Trata-se de uma função de ativação muito utilizada em redes neurais artificiais, principalmente nas redes consideradas como de aprendizado profundo.

O objetivo de uso com uma função ReLU é introduzir não-linearidade ao modelo de rede neural, para que a rede possa aprender padrões complexos dos dados de entrada e saída.

Ao utilizar uma função ReLU, o resultado será o próprio valor de entrada se este for positivo ou o resultado será 0 se o valor de entrada for negativo.

É devido a esta simplicidade computacional que a função ReLU é bastante utilizada em redes neurais.

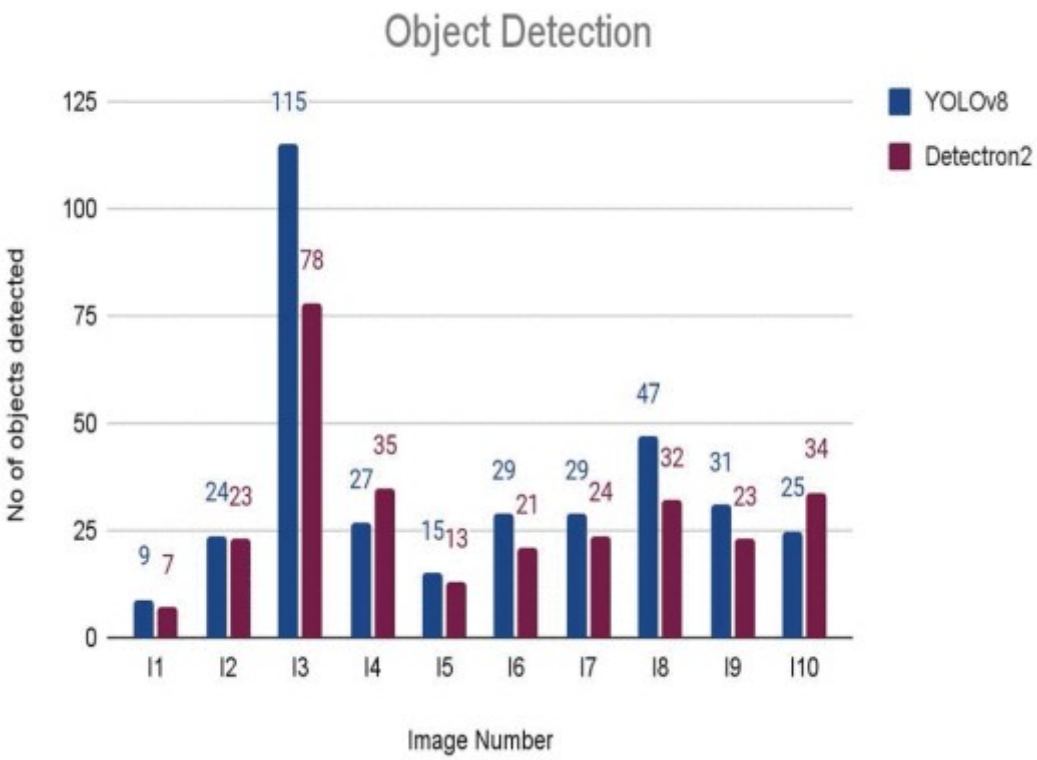
YOLO tem sido utilizado com sucesso devido à sua particularidade de passada única (*single-shot detection*) e velocidade, identificando múltiplos objetos com bons resultados de acurácia. Pode ser utilizado para contagem de pessoas, detectando os objetos em uma imagem (as pessoas) e realizando a contagem do número de objetos detectados.

Em um trabalho publicado em 2023, de autoria de M. Wadhwa e demais (Wadhwa *et al.*, 2023), os autores utilizaram o YOLOv8 e o Detectron2, este produzido pela equipe de Inteligência Artificial do Facebook (*Facebook AI Research*), que é um algoritmo para tarefas de detecção e segmentação de objetos em imagens. O objetivo do trabalho foi verificar se o YOLOv8 conseguiria ter melhores resultados do que o Detectron2.

Os conjuntos de dados utilizados foram o COCO (*Common Objects in Context*) para treinamento dos modelos e o conjunto de dados ShanghaiTech (Zhang *et al.*, 2016) para teste.

O YOLOv8 foi capaz de detectar mais objetos do que o Detectron2 na maioria dos casos (8 de 10 vezes). No entanto, o Detectron2 apresentou pontuações de confiança superiores em termos de detecção de objetos. Portanto, embora o YOLOv8 tenha detectado um maior número de objetos, o Detectron2 foi mais confiável nas detecções que realizou. O resultado obtido ao comparar os dois modelos pode ser visto na Figura 24.

Figura 24 – Comparação entre YOLOv8 e Detectron2



Fonte: (Wadhwa *et al.*, 2023)

Os autores concluíram que o YOLOv8 se mostrou mais adequado para a tarefa específica de contagem de multidões, devido à sua capacidade de detectar um maior número de objetos rapidamente, oferecendo uma abordagem de detecção única que permite processar imagens com maior velocidade.

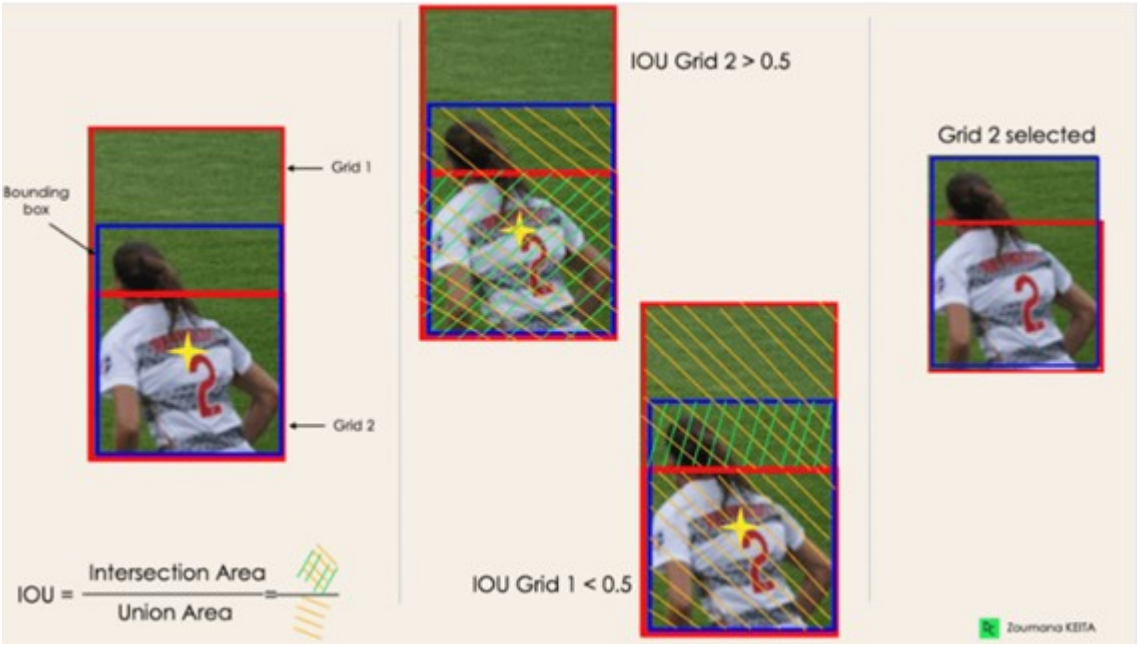
YOLOv8 também foi utilizado para a vigilância de multidões, a partir de imagens de Veículos Aéreos Não Tripulados (UAVs), em trabalho publicado em 2023 por M. Anzaruddin e demais (Anzaruddin *et al.*, 2023). Foi utilizado o conjunto de dados COCO (Lin *et al.*, 2015) e um outro conjunto elaborado pelos autores contendo vídeos de diversas partes da Índia.

Neste estudo foi explorada a eficácia do YOLO para identificar multidões, utilizando métricas como a Interseção sobre União (IOU). Esta é uma métrica comum usada em algoritmos de detecção de objetos para medir a precisão das caixas delimitadoras previstas.

O IOU funciona como um limiar (*threshold*) definido pelo usuário e tem o valor entre 0 e 1. O objetivo do IOU é descartar caixas delimitadoras irrelevantes, mantendo apenas as mais precisas para a detecção de objetos.

Um exemplo claro disto é o apresentado na Figura 25, elaborada por (Keita, 2022):

Figura 25 – Exemplo de uso do IOU



Fonte: (Keita, 2022)

As grades com um limiar menor ou igual ao limiar definido pelo usuário terão a sua previsão ignorada. São consideradas apenas as grades cujo IOU for maior que o limiar definido. No caso da imagem acima, haviam duas áreas candidatas à grade e apenas a grade 2 (*Grid* 2) foi selecionada.

Para a tarefa de detecção e contagem de pessoas, o YOLOv8 (Jocher; Chaurasia; Qiu, 2023) se destaca como a versão estável mais atual e é altamente recomendado devido ao seu desempenho avançado e eficiência aprimorada em relação às versões anteriores. Esse modelo incorpora melhorias expressivas tanto em sua estrutura quanto nas técnicas de treinamento, tornando-o uma excelente escolha para essas aplicações.

Entre os motivos que tornam o YOLOv8 uma opção vantajosa, estão sua maior precisão, resultante de refinamentos que aprimoraram a detecção de objetos; sua eficiência, com tempos de inferência reduzidos, favorecendo o uso em tempo real; e sua flexibilidade, permitindo ampla customização, como ajustes de hiperparâmetros e configurações de data augmentation, o que facilita a adaptação do modelo às demandas específicas de cada projeto.

O YOLOv8 possuí versões específicas para cada tipo de tarefa. Existem 5 tipos de tarefas disponíveis, descritos a seguir:

2.7.1 Detecção de Objetos

A Detecção de Objetos (*Object Detection*) é uma tarefa cujo objetivo é identificar a localização e a classe de determinado objeto em uma imagem ou em um vídeo. A partir disto é possível realizar a contagem de quantos objetos na imagem ou vídeo são pertencentes a cada classe. É a tarefa que será utilizada neste trabalho.

2.7.2 Segmentação de Instância

A Segmentação de Instância (*Instance Segmentation*) é uma técnica que além de detectar objetos em uma imagem também faz a segmentação desses objetos. A saída deste modelo é um conjunto de máscaras ou contornos que delineiam cada objeto em uma imagem, atribuindo a classe pertencente ao objeto e pontuações de confiança. Essa técnica é útil para identificar os objetos e a forma destes.

2.7.3 Estimativa de Pose

A tarefa de Estimativa de Pose (*Pose Estimation*) é uma técnica que realiza a identificação de pontos específicos em uma imagem, chamados de pontos-chave. Estes podem representar diversas partes de um objeto, como articulações ou pontos de referência. A saída do modelo é um conjunto de pontos que representam os pontos-chave do objeto na imagem, junto com pontuações de confiança para cada ponto.

2.7.4 Detecção de Objetos Orientada

A Detecção de Objetos Orientada (*Oriented Bounding Boxes - OBB*) inclue um ângulo adicional para melhorar a precisão da localização de objetos em imagens. As caixas delimitadores geradas por uma OBB podem girar para se ajustar melhor à orientação do objeto detectado. A tarefa de OBB é útil para aplicações que exigem maior precisão para a localização de objetos, a partir de imagens aéreas ou de satélite.

2.7.5 Classificação de Imagens

A tarefa de Classificação de Imagens (*Image Classification*) tem o objetivo de classificar imagens em classes prédefinidas. A saída do modelo é um classificador de imagens, contendo um rótulo de classe e uma pontuação de confiança. É utilizada quando precisa conhecer a qual classe uma imagem pertence e não precisa saber onde estão os objetos dessa classe ou qual a sua forma exata.

2.8 Modelos YOLOv8 para tarefas de detecção

Os modelos YOLOv8 disponíveis para a tarefa de detecção foram pré-treinados no conjunto de dados COCO (Lin *et al.*, 2015), que é um *dataset* usado para tarefas de detecção de objetos, segmentação, contendo 330 mil imagens com anotações detalhadas

para 80 classes de objetos. Os modelos pré-treinados e suas métricas de performance podem ser vistos na Tabela 1:

Tabela 1 – Modelos YOLOv8n pré-treinados

Model	*Size* (pixels)	mAPval 50-95	*Speed* CPU ONNX (ms)	*Speed* A100 TensorRT (ms)	*Params* (M)	*FLOPs* (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

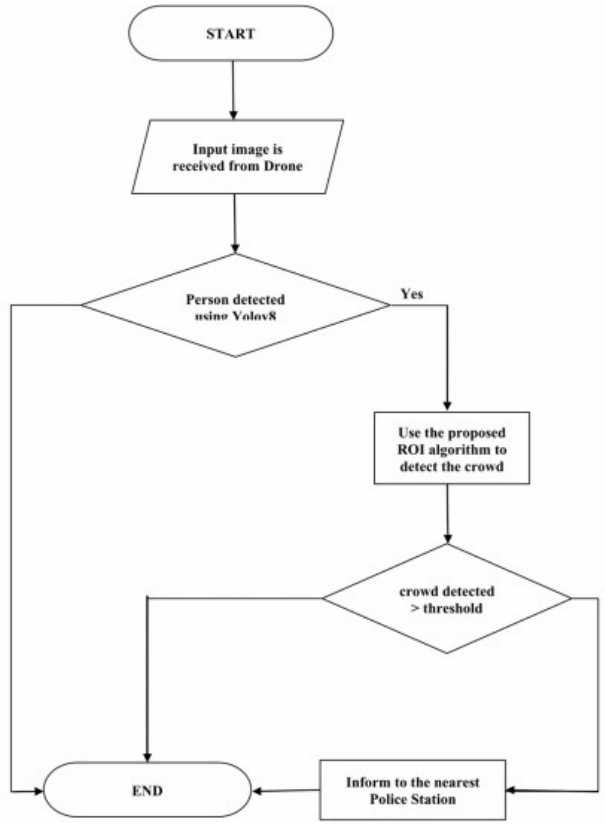
Fonte: Autor

2.8.1 Modelo YOLOv8 para detecção de multidões

Em artigo de (Anzaruddin *et al.*, 2023) é proposto um fluxo de detecção de multidões, usando imagens enviadas por um UAV (*Unmanned Aerial Vehicle*) ou Veículo Aéreo Não Tripulado. Se é detectada uma pessoa na imagem através do YOLOv8, na sequência é feita uma avaliação da área de interesse (ROI - *Region of Interest*), que é uma área delimitada para aquele local, com base em características físicas do mesmo e é feita uma contagem dos indivíduos presentes no ROI e se essa contagem está acima ou abaixo de um valor limite definido para o local. Esse valor limite é definido a partir de vários fatores, como o tamanho da área e/ou a multidão máxima suportada para o local. Se o valor de pessoas obtido pelo modelo naquela área for superior ao valor limite definido, é emitido um alerta pelo sistema.

O fluxo proposto pelos autores pode ser visto na Figura 26:

Figura 26 – Fluxograma proposto para análise de multidões



Fonte: (Anzaruddin *et al.*, 2023)

Foram testadas pelos autores diversas versões do YOLO e a versão com a melhor acurácia foi o YOLOv8s, com o resultado de 82,3%. A conclusão do trabalho é de que o modelo utilizado foi eficiente em termos de processamento, precisão e tempo de resposta. A solução proposta de definir um valor limite, que serve como um alerta de multidão que é enviado para equipes de segurança responsáveis, é interessante do ponto de vista de gerenciamento público de eventos e manifestações.

YOLO também foi utilizado para detecção e contagem de pessoas em imagens de CCTV (*Closed Circuit Television*) em artigo de G. Vasavi e demais (Vasavi *et al.*, 2023). Neste artigo foram utilizadas imagens privadas de um CCTV, com diferentes cenas e densidades de multidão. Foi implementado um algoritmo para rastreamento e contagem das pessoas ao entrarem ou saírem de áreas determinadas (regiões de interesse).

O modelo baseado em YOLO mostrou alta precisão e *recall* na identificação e localização de pessoas, com uma precisão média de 0.93. A contagem de pessoas apresentou um erro médio absoluto de apenas 1,2 pessoas por quadro, indicando alta precisão na estimativa de densidade populacional.

Os autores concluíram que o modelo desenvolvido apresentou um excelente desempenho mas que desafios enfrentados, como oclusões parciais, exigem a necessidade de mais pesquisa e melhoria do modelo.

Estes trabalhos citados corroboram com a proposta deste estudo, de que o YOLO é um modelo bem sucedido para tarefas de detecção e contagem de pessoas em multidões.

O modelo pré-treinado utilizado neste estudo é o YOLOv8n. O "n"no final do nome se refere ao tamanho do modelo. É a versão ideal para aplicações que exigem baixo consumo de recursos computacionais, como dispositivos móveis e/ou sistemas embarcados, a exemplo do uso de drones (UAV) para processamento em tempo real das imagens. A diferença entre as versões do YOLOv8, mostrada na Tabela 1, é decorrente do número de parâmetros de cada modelo e complexidade da arquitetura. Quanto maior o número de parâmetros do modelo, maior a precisão e maior o custo computacional.

Desta forma, a escolha pelo YOLOv8n para este trabalho se justifica por ser um modelo leve, rápido e que apresenta equilíbrio entre velocidade e precisão, podendo ser usado em aplicações em tempo real, como sistemas de vigilância, drones e dispositivos móveis.

2.9 Métodos comuns utilizados para contagem de multidões

Os principais métodos utilizados em tarefas de contagem de multidões podem ser divididos em duas categorias, conforme (Ilyas; Shahzad; Kim, 2020):

2.9.1 Métodos de Contagem Supervisionada

Na contagem supervisionada os dados são conhecidos e rotulados, podendo ser dividida em:

- Contagem por regressão:

Neste método é feita uma estimativa da densidade da multidão através de uma regressão. Segundo (Ilyas; Shahzad; Kim, 2020), este método foi explorado pela primeira vez por (Cheng *et al.*, 2014). Métodos baseados em regressão são mais eficazes para extrair características de partes da imagem.

- Contagem por estimativa de densidade:

Segundo (Ilyas; Shahzad; Kim, 2020) essa técnica utiliza informações espaciais da imagem para realizar uma estimativa de densidade. Dessa forma, o foco não é localizar e contar os objetos individualmente na cena mas sim realizar o cálculo da densidade ocupada na imagem por estes objetos. Consegue superar os problemas decorrentes de oclusão.

- Contagem por detecção:

Esta técnica procura realizar a contagem dos objetos identificando partes do objeto, como ombros, cabeças e a partir dessa detecção é realizada a contagem do número de objetos presentes na imagem. É afetada por problemas de oclusão.

- Contagem por Redes Neurais Convolucionais:

Continua sendo um dos métodos mais utilizados com melhores resultados apresentados (Khan; Menouar; Hamila, 2023). São empregadas camadas de convolução, pooling, uso de função de ativação (ReLU sendo a mais comum) e redes neurais totalmente conectadas. Considerado pela literatura o método mais eficiente mas com custo elevado devido à complexidade computacional.

2.9.2 Métodos de Contagem Não-supervisionada

Nestes métodos os dados não são conhecidos e não existem rótulos. Um exemplo é o método de contagem por agrupamento. Neste o objetivo é encontrar características semelhantes e agrupá-las. São extraídas as características de baixo nível e essas características são agrupados em grupos, sendo realizado na sequência a contagem dos objetos em cada grupo.

2.9.3 Desafios comuns na contagem de multidões

A tarefa de contagem de multidões apresenta alguns desafios comuns que impactam na obtenção do melhor resultado. Estes desafios foram listados em (Khan; Menouar; Hamila, 2023) e podem ser vistos conforme abaixo:

- Variação de cena: imagens ou vídeos obtidos de fontes diferentes, como CCTV ou imagens áereas de drones;
- Variação de escala: um objeto na imagem, por exemplo um humano, pode ter diferentes tamanhos em imagens variadas, causadas pela distância, perspectiva ou resolução;
- Densidade da multidão: o número de pessoas presentes em uma multidão pode variar de uma imagem para outra;
- Distribuição não-uniforme da multidão: pessoas em locais diferentes podem gerar uma distribuição diferente da multidão;
- Oclusão: é quando um objeto é obstruído parcialmente ou completamente por outro. Por exemplo: uma ou várias pessoas à frente de outras pessoas em uma imagem;

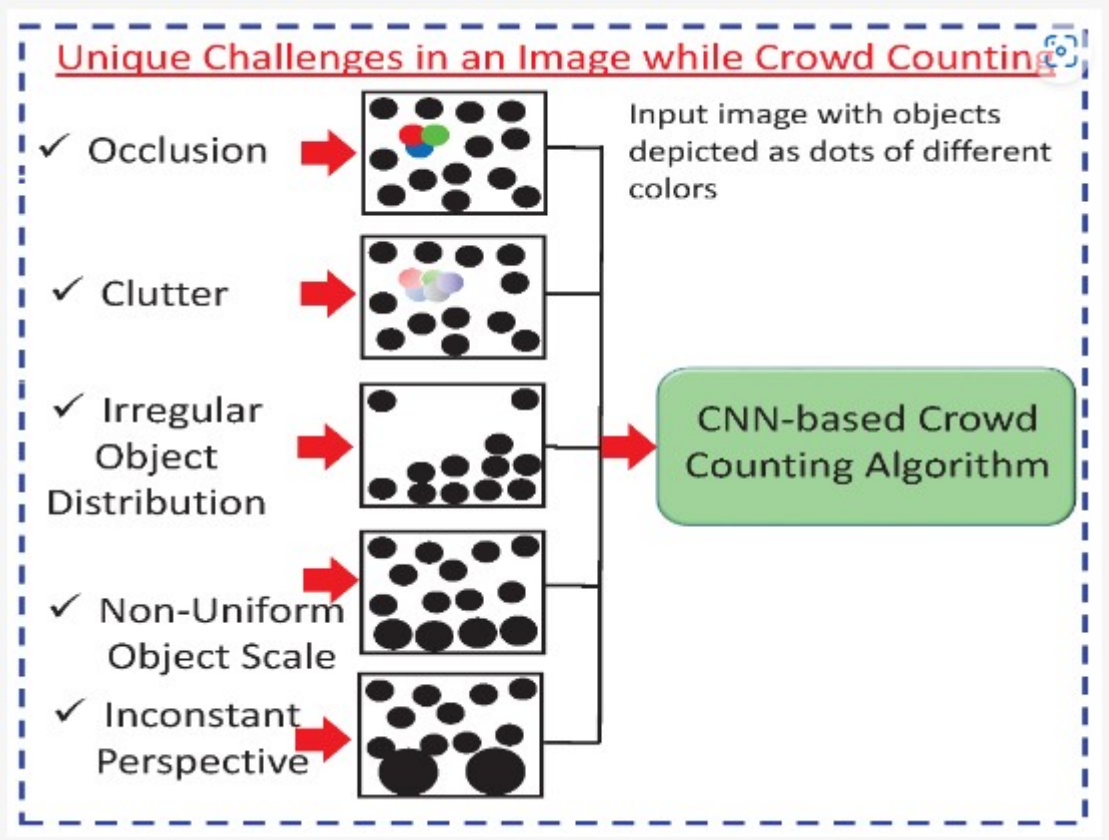
- Fundo da imagem: o fundo da imagem contém vários objetos que não são de interesse para a contagem, podendo dificultar a contagem das pessoas na imagem;

- Variações de iluminação: imagens podem conter diferentes condições de luz, por exemplo imagens obtidas durante o dia, com luz natural versus imagens obtidas durante a noite com ausência de luz ou com luz artificial;

- Outros desafios: condições do tempo como chuva, sol, neve. Imagens com muito ruído ou de baixa qualidade, etc.

Em (Ilyas; Shahzad; Kim, 2020) foi apresentado um quadro que resume os desafios mais comuns, listados anteriormente, encontrados em tarefas de contagem de multidões, conforme Figura 27:

Figura 27 – Desafios comuns na contagem de multidões



Fonte: (Ilyas; Shahzad; Kim, 2020)

3 AVALIAÇÃO EXPERIMENTAL

3.1 Dados

O conjunto de dados a ser utilizado neste trabalho será o VisDrone 2019 (Zhu *et al.*, 2021a), composto de 10.209 imagens capturadas a partir de câmeras instaladas em drones.

É um conjunto de dados disponibilizado pela equipe AISKYEYE do Laboratório de Aprendizagem Automática e Extração de Dados da Universidade de Tianjin, na China. Este *dataset* contém dados anotados para várias tarefas de visão computacional relacionadas com a análise de imagem e vídeo obtidas a partir do uso de drones.

É composto de 288 clipes de vídeo e 10.209 imagens estáticas, capturadas por várias câmeras montadas em drones que "...apresentam uma ampla variedade de aspectos, incluindo localização (obtida de 14 cidades diferentes separadas por milhares de quilômetros na China), ambiente (urbano e rural), objetos (pedestres, veículos, bicicletas, etc.) e densidade (cenas esparsas e lotadas). Observe que o conjunto de dados foi coletado usando várias plataformas de drones (ou seja, drones com diferentes modelos), em cenários diferentes e sob várias condições climáticas e de iluminação"(Zhu *et al.*, 2021b).

A divisão do conjunto de dados para os experimentos deste estudo foi realizada da seguinte forma: 6.471 imagens para treino, 548 para validação e 1.610 para teste.

3.2 Métricas

3.2.1 Métricas para Avaliação do Modelo

As métricas são utilizadas para avaliar a performance de um modelo de detecção de objetos. Demonstram se o modelo está sendo eficiente ao identificar e localizar as classes presentes em cada imagem. Para o YOLOv8 as métricas a seguir são utilizadas para avaliar a qualidade geral do modelo.

3.2.1.1 União sobre Interseção (IoU)

O IoU (*Intersection over Union)* mede a sobreposição entre as caixas delimitadoras preditas e as verdadeiras, indicando a qualidade da localização dos objetos.

3.2.1.2 Precisão Média (AP)

A AP combina precisão e revocação em um único valor, representando o desempenho geral do modelo.

3.2.1.3 Média da Precisão Média (mAP)

A mAP calcula a AP média para todas as classes, fornecendo uma avaliação completa em casos de múltiplas classes.

3.2.1.4 Precisão e Revocação

A precisão avalia a capacidade do modelo de evitar erros falsos positivos, enquanto a revocação mede sua habilidade de detectar todos os objetos.

3.2.1.5 Pontuação F1

A Pontuação F1 equilibra precisão e revocação, fornecendo uma medida abrangente do desempenho do modelo.

3.2.1.6 mAP50 (*mean Average Precision at IoU threshold* 0.5)

A mAP50 avalia a precisão do modelo em detectar objetos com uma sobreposição mínima de 50% com os objetos reais. É uma medida de desempenho focada em detecções relativamente fáceis.

3.2.1.7 mAP75 (*mean Average Precision at IoU threshold* 0.75)

Mesma definição da mAP50 porém considerando uma sobreposição mínima de 75%.

3.2.1.8 mAP50-95 (*mean Average Precision across multiple IoU thresholds from 0.5 to 0.95*)

A mAP50-95 é uma média ponderada da mAP calculada em diferentes níveis de sobreposição, que variam de 50% a 95%. Isso fornece uma visão mais completa do desempenho do modelo, considerando tanto detecções fáceis quanto difíceis.

3.2.2 Funções de perda para avaliação do modelo

Em modelos de detecção de objetos como o YOLOv8, a função de perda (*loss function*) é muito importante para o processo de treinamento. Ela quantifica o erro entre as previsões do modelo e as anotações verdadeiras, fazendo com que o modelo ajuste os seus parâmetros para minimizar esse erro. As funções de perda principais no YOLOv8 são:

3.2.2.1 *cls_loss* (Perda de Classificação)

O objetivo da *cls_loss* é avaliar a classificação correta dos objetos. Ela mede o quão bem o modelo está classificando os objetos dentro das bounding boxes. A *cls_loss* compara as probabilidades previstas pelo modelo para cada classe com as classes

verdadeiras anotadas nas imagens. Quanto menor a *cls_loss*, mais confiante e correto está o modelo ao identificar os objetos em uma imagem.

3.2.2.2 *box_loss* (Perda de *Bounding Box*)

O objetivo da *box_loss* é a precisão da localização dos objetos. Ela mede a precisão das bounding boxes previstas pelo modelo em relação às bounding boxes verdadeiras, comparando as coordenadas das bounding boxes previstas com as coordenadas verdadeiras. Uma *box_loss* baixa indica que o modelo está localizando os objetos com precisão nas imagens.

3.2.2.3 *dfl_loss* (Perda Focal de Distribuição)

Combina os dois aspectos anteriores e se concentra nos exemplos difíceis. A *dfl_loss* é uma função de perda introduzida no YOLOv8, que combina *cls_loss* e *box_loss* para um treinamento mais eficiente e robusto. Ela considera tanto a classificação das classes quanto a precisão das bounding boxes, mas com um foco maior em exemplos difíceis ou ambíguos. A *dfl_loss* contribui para o modelo se concentrar em exemplos difíceis e melhorar a precisão geral da detecção.

3.3 Pré-processamento

O pré-processamento em uma tarefa de visão computacional é composto de etapas como redimensionamento do tamanho das imagens, normalização dos valores de pixels para um intervalo definido, divisão dos dados em conjuntos de treinamento, validação e teste e aumento do conjunto de dados através de técnicas de data augmentation.

A etapa de pré-processamento é composta de objetivos sobrepostos (Lakshmanan; Görner; Gillard, 2021) como: redimensionamento do tamanho da imagem, aumentar a qualidade das imagens e aumentar a qualidade do modelo.

3.3.1 Redimensionamento das imagens

É uma das etapas mais comuns e importante no pré-processamento de imagens. Pode até ser uma exigência por parte de alguns tipos de modelos, que exigem um tamanho de imagem padronizado. O redimensionamento torna o tamanho das imagens padronizado e contribui para reduzir a complexidade computacional.

Uma vantagem do uso do YOLOv8 é que essa etapa de pré-processamento pode ser feita diretamente na etapa de treinamento do modelo, ajustando o parâmetro '*imgsz*' para o tamanho de imagem desejado. Dessa forma, o modelo efetuará o redimensionamento de cada imagem do conjunto de treinamento, mantendo a proporção original.

3.3.2 Aumentar a qualidade das imagens

Algumas imagens do conjunto de dados podem ter problemas como condições de iluminação diferentes umas das outras, resultando em imagens mais claras ou mais escuras que as outras. Isso pode afetar a capacidade do modelo de entender os padrões visuais de forma consistente, levando a um desempenho inferior na identificação ou classificação de objetos. Para mitigar esse problema, técnicas de normalização de imagem, como equalização de histograma ou ajuste de brilho e contraste, podem ser aplicadas para uniformizar as condições de iluminação e garantir que o modelo receba dados mais homogêneos e representativos.

Outra técnica comumente usada é a normalização dos valores dos pixels. Essa normalização faz com que os valores dos pixels se situem em uma faixa padronizada. Isso melhora a velocidade de treinamento e o desempenho do modelo. Algumas técnicas comumente usadas são: Escalonamento *min-max*, que transforma os pixels para uma faixa entre 0 e 1; e a normalização Z-score, que escala os valores dos pixels com base na sua média e no seu desvio padrão.

Com o uso do YOLOv8, essa etapa também é realizada durante a fase de treinamento do modelo. O escalonamento *min-max* é realizado automaticamente.

3.3.3 Aumentar a qualidade do modelo

Conforme (Lakshmanan; Görner; Gillard, 2021), um terceiro objetivo do pré-processamento é o de realizar transformações nas imagens de entrada que resultem em um melhor desempenho do modelo.

A técnica mais comumente utilizada é a conhecida como *Data Augmentation*. Essa técnica consiste em aumentar o tamanho do conjunto de dados criando versões artificiais das imagens existentes no conjunto. Também é a primeira técnica utilizada quando existe o problema de poucos dados disponíveis para treinamento.

O objetivo da *Data Augmentation* é aumentar o tamanho e a qualidade dos dados de treinamento, para que o modelo possa ter melhor desempenho e uma melhor generalização.

As técnicas mais comuns de data augmentation são: rotacionar as imagens, alterar a escala, ajustar a cor, ajustar o brilho e espelhamento (*flipping*).

3.4 Metodologia

Esta seção descreve a metodologia empregada neste estudo para desenvolver um modelo de contagem de pessoas em multidões a partir de imagens aéreas utilizando a arquitetura YOLO. São abordados a seguir o ambiente de desenvolvimento, o conjunto de dados utilizado, o processo de modelagem e os experimentos realizados.

3.4.1 Ambiente de Desenvolvimento

Todo o desenvolvimento e experimentação do modelo foram realizados em um ambiente local, utilizando o *Jupyter Notebook* como plataforma de programação. Para acelerar o processamento e treinamento do modelo, todos os treinamentos foram realizados utilizando o poder computacional da GPU (*Graphics Processing Unit*).

3.4.2 Customização do Conjunto de Dados VisDrone

O conjunto de dados original foi customizado para atender especificamente aos objetivos deste trabalho. Foram realizadas as seguintes alterações:

- Redução de Classes: A partir do Experimento 2, os arquivos de anotações e rótulos de cada imagem presente nos conjuntos de treinamento, validação e teste, foram alterados para incluir apenas duas classes de objetos: '*pedestrian*' e '*people*'. Essa decisão visou simplificar o problema e direcionar o aprendizado do modelo para a detecção específica de pessoas.

- Unificação de Classes: No Experimento 5 optou-se por unificar as classes '*pedestrian*' e '*people*' em uma única classe, denominada '*people*'. Essa estratégia teve como objetivo verificar se a simplificação das classes proporciona melhoria nos resultados do modelo.

- Armazenamento Local: Tanto o conjunto de dados original quanto as versões modificadas foram armazenadas localmente para facilitar o acesso e agilizar o processo de treinamento.

3.4.3 Processo de Modelagem

O modelo disponível YOLOv8n pré-treinado (conjunto COCO) foi utilizado em todos os experimentos. É o menor modelo da arquitetura YOLOv8 disponível. É utilizado em aplicações que exigem baixo consumo de recursos computacionais, podendo ser utilizado em sistemas embarcados, como drones.

3.4.4 Etapas da modelagem

- Modelo de base com os parâmetros padronizados: O modelo de base utilizou o YOLOv8n, utilizando os parâmetros de treinamento padronizados, com apenas a alteração no número de épocas de 100 para 300. Isso permitiu avaliar o desempenho do modelo em um ponto de partida já otimizado.

- Ajuste de Hiperparâmetros com *model.tune()*: Buscando otimizar o modelo para o problema específico deste estudo, utilizou-se a função *model.tune()* do YOLO. Essa

funcionalidade permite a busca automatizada por hiperparâmetros mais adequados ao conjunto de dados e à tarefa de detecção e consequente contagem de objetos. Em um dos experimentos a função *model.tune()* foi executada com 20 iterações para determinar os melhores parâmetros.

- Pré-processamento Automático de Imagens: Um diferencial da arquitetura YOLO é a capacidade de realizar o pré-processamento das imagens de entrada automaticamente durante a etapa de treinamento. Essa característica simplifica o processo de desenvolvimento, eliminando a necessidade de etapas manuais de tratamento das imagens.

3.4.5 Considerações

Este estudo utilizou a arquitetura YOLO, usando o seu modelo YOLOv8n pré-treinado, para realizar a contagem de pessoas em multidões a partir de imagens aéreas. Os resultados obtidos serão apresentados e analisados no próximo capítulo.

4 RESULTADOS

4.1 Experimento 1 - Modelo base

Foi feito o experimento utilizando o modelo pré-treinado YOLOv8n, utilizando os parâmetros default do modelo, tendo sido alterado apenas o número de épocas (*epochs*) de 100 para 300.

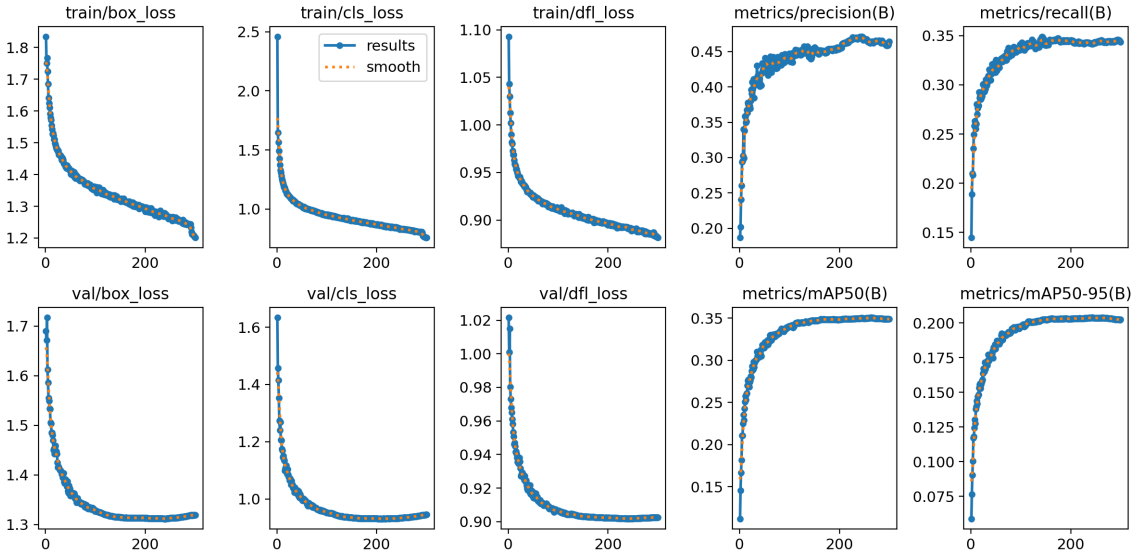
Uma época é a passagem completa do modelo por todo o conjunto de dados de treinamento. Em cada época o modelo processa cada exemplo do conjunto de treinamento uma vez e os seus parâmetros são atualizados com base no algoritmo de aprendizado. Se o modelo apresentar *overfitting* (sobreajuste) antes do valor definido de épocas, é um indicador de que o número de épocas pode ser diminuído. Se não apresentar *overfitting* até o número total de épocas, este número pode ser aumentado.

Sobre o tamanho das imagens de entrada, foi utilizado o valor *default* para o parâmetro '*imgsz*' definido em 640. Trata-se do tamanho das imagens usadas no treinamento. Todas as imagens são ajustadas para esse tamanho antes de serem usadas pelo modelo. O tamanho das imagens pode influenciar a capacidade do modelo de aprender padrões e fazer previsões corretas. Também pode afetar a quantidade de recursos computacionais necessários (como memória e processador) para treinar o modelo. Imagens contendo muitos objetos pequenos podem se beneficiar de um treinamento com maior resolução da imagem, como por exemplo '*imgsz*=1280'.

O tamanho das imagens é importante para o treinamento de um modelo de visão computacional. Se as imagens forem muito grandes, o treinamento pode ser lento e exigir mais recursos. Se forem muito pequenas, o modelo pode ter dificuldade em aprender detalhes importantes.

As curvas de treinamento, validação e métricas do modelo YOLOv8n, para 300 épocas, no conjunto de validação, podem ser vistas na Figura 28:

Figura 28 – Curvas de treinamento, validação e métricas



Fonte: Autor

Os resultados sugerem que o modelo treinado pode ter atingido seu desempenho máximo no conjunto de dados com os hiperparâmetros atuais. Isso é sugerido através do platô nas três métricas de perda no conjunto de validação e das métricas mAP50 e mAP50-95.

Os resultados para o Conjunto de Teste podem ser vistos na Tabela 2:

Tabela 2 – Resultados Detalhados Experimento 1

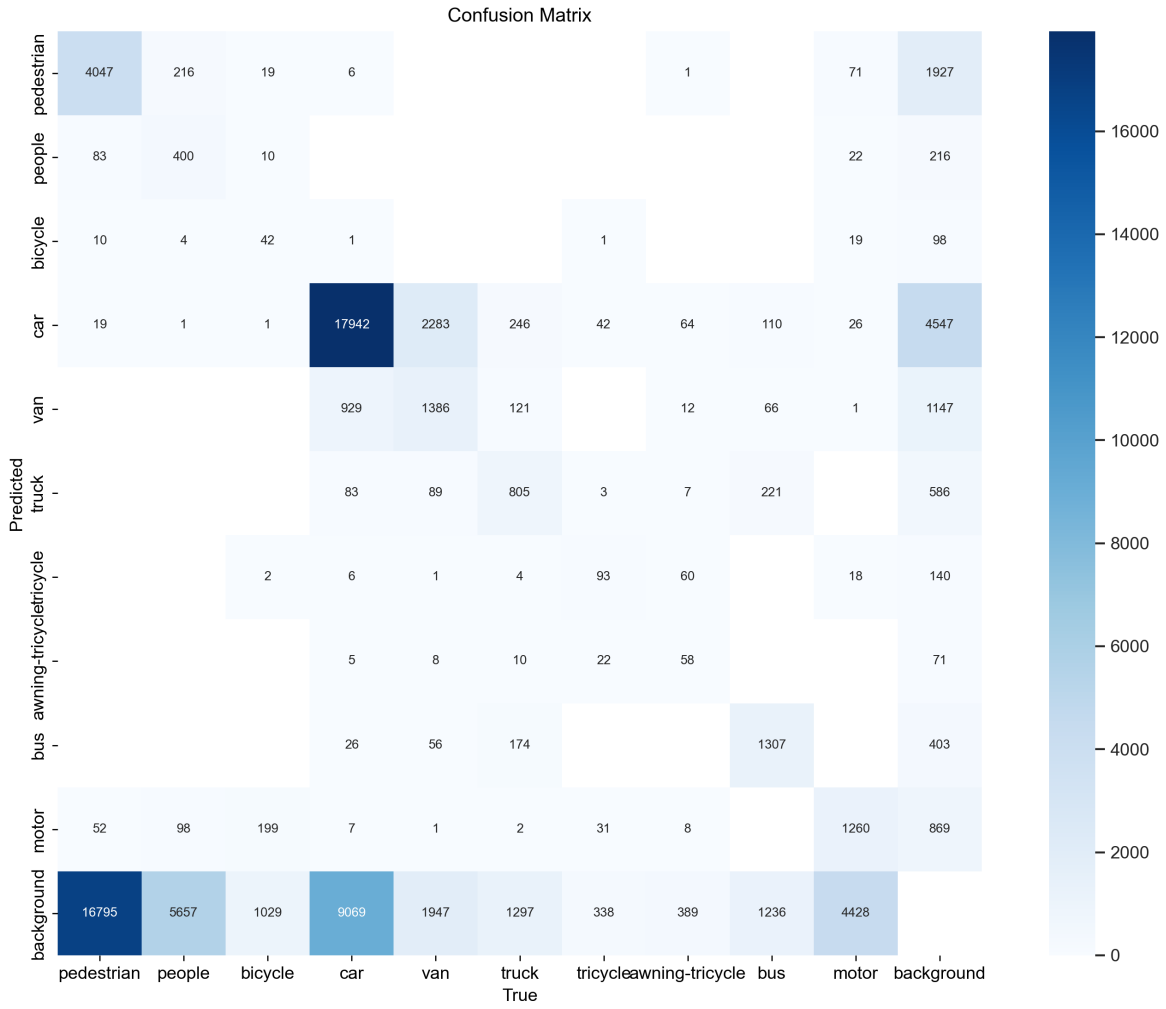
Class	Images	Instances	Precision	Recall	mAP50	mAP50-95
all	1610	75102	0.405	0.307	0.285	0.161
pedestrian	1197	21006	0.471	0.225	0.238	0.0932
people	797	6376	0.455	0.097	0.128	0.042
bicycle	377	1302	0.217	0.069	0.062	0.023
car	1530	28074	0.613	0.695	0.679	0.417
van	1168	5771	0.353	0.362	0.315	0.2
truck	750	2659	0.385	0.382	0.338	0.212
tricycle	245	530	0.219	0.249	0.15	0.082
awning-tricycle	233	599	0.351	0.19	0.166	0.088
bus	838	2940	0.617	0.513	0.528	0.358
motor	794	5845	0.37	0.291	0.243	0.093

Fonte: Autor

As duas classes de interesse são: '*pedestrian*' e '*people*'. No conjunto de dados VisDrone é considerado como sendo da classe '*pedestrian*' se o ser humano estiver em uma pose em pé ou estiver andando. Caso contrário, será classificado como uma pessoa, será da classe '*people*' (Zhu *et al.*, 2021a).

Através da matriz de confusão é possível verificar o desempenho do modelo para a classificação de cada classe, conforme demonstrado na Figura 29.

Figura 29 – Experimento 1 - Conjunto de Teste



Fonte: Autor

O modelo classificou 83 instâncias da classe '*pedestrian*' como sendo da classe '*people*'. Isso é esperado dada a semelhança entre essas classes. E de um total de 21.006 instâncias da classe '*pedestrian*', o modelo classificou corretamente apenas 4.047. É uma taxa baixa de classificação correta.

Para a classe '*people*', o modelo classificou 216 instâncias de '*people*' como sendo da classe '*pedestrian*'. De um total de 6.376 instâncias da classe '*people*', o modelo classificou corretamente apenas 400. Isso indica que o modelo é melhor em identificar '*pedestrian*' corretamente do que '*people*'.

A semelhança visual entre as classes '*pedestrian*' e '*people*' pode estar levando a erros de classificação. Outros fatores, como o tamanho e a posição dos objetos na imagem, também podem contribuir para essa confusão.

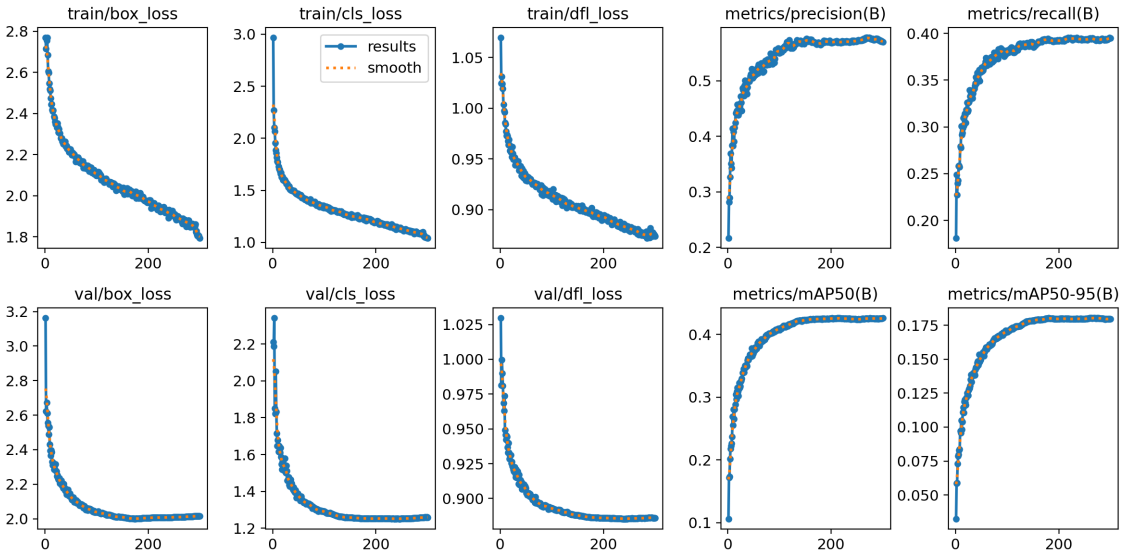
Mas os resultados para ambas as classes indicam que o desempenho do Experimento 1 está abaixo do ideal. A métrica de precisão, que significa a proporção de detecções positivas que são verdadeiramente positivas, está abaixo de 50% para ambas as classes. E o resultado da métrica *recall* é ainda mais baixo, sendo próximo de 25% para a classe '*pedestrian*', ou seja, o modelo identificou apenas cerca de 25% dos pedestres presentes nas imagens. E o *recall* é próximo de 10% para a classe '*people*', indicando que o modelo teve uma dificuldade maior para identificar pessoas nas imagens. Essa dificuldade não é vista para a classe '*car*', em que o Experimento 1 teve um *recall* de 70%, que demonstra que para essa classe específica o modelo do Experimento 1 consegue ter um bom resultado.

Considerando a dificuldade encontrada pelo Experimento 1 em obter melhores resultados na identificação das classes '*pedestrian*' e '*people*', evidenciada pelos resultados da Tabela 2, e pelo platô atingido nas funções de perda após 300 épocas de treinamento, optou-se por realizar um novo experimento. A hipótese é que, ao treinar o modelo exclusivamente com essas duas classes, o modelo poderá se especializar na tarefa de contagem de pessoas, melhorando sua capacidade de identificá-las e reduzindo o número de falsos positivos e falsos negativos.

4.2 Experimento 2

As curvas de treinamento, validação e métricas do modelo YOLOv8n, para 300 épocas, no Conjunto de Validação, apenas com as classes 'pedestrian' e 'people' podem ser vistas na Figura 30:

Figura 30 – Curvas de treinamento, validação e métricas



Fonte: Autor

Assim como ocorreu no Experimento 1, o modelo treinado pode ter atingido seu desempenho máximo no conjunto de dados com os hiperparâmetros atuais. No entanto,

houveram melhorias dos resultados no Conjunto de Teste, comprovando que reduzir o número de classes contribuiu para melhorar a performance do modelo, conforme Tabela 3:

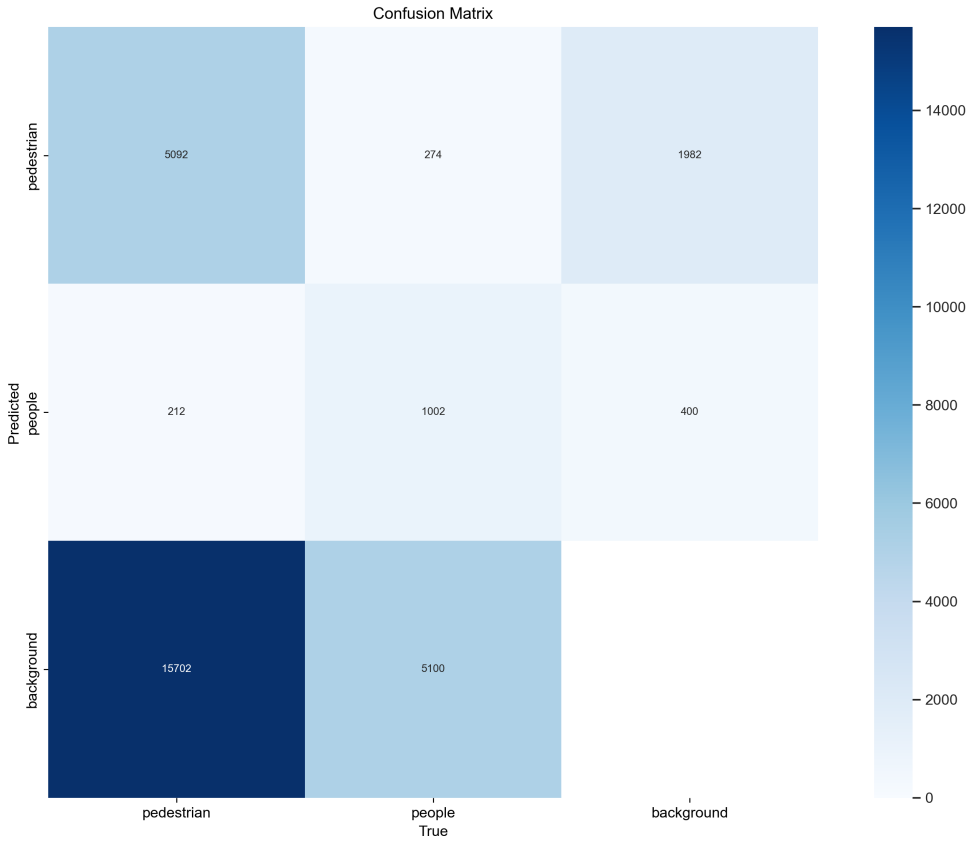
Tabela 3 – Experimento 1 x Experimento 2 - Conjunto de Teste

Resultados Experimento 1						
Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
pedestrian	1197	21006	0.471	0.225	0.238	0.093
people	797	6376	0.455	0.097	0.128	0.042
Resultados Experimento 2						
Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
pedestrian	1197	21006	0.494	0.283	0.288	0.116
people	797	6376	0.434	0.207	0.196	0.067

Fonte: Autor

A melhoria dos resultados também pode ser vista através da matriz de confusão da Figura 31:

Figura 31 – Experimento 2 - Conjunto de Teste



Fonte: Autor

O modelo classificou 212 instâncias da classe '*pedestrian*' como sendo da classe '*people*' (no Experimento 1 foram 83). E de um total de 21.006 instâncias da classe '*pedestrian*', o modelo classificou corretamente apenas 5.092 (no Experimento 1 foram

4.047). Para a classe '*people*', o modelo classificou 274 instâncias de '*people*' como sendo da classe '*pedestrian*' (no Experimento 1 foram 216). De um total de 6.376 instâncias da classe '*people*', o modelo classificou corretamente apenas 1.002 (no Experimento 1 foram 400). Apesar dos resultados melhores, isso ainda sugere que o modelo está tendo dificuldade para identificar e classificar corretamente ambas as classes.

O comparativo dos resultados das métricas entre os dois experimentos e sua variação percentual podem ser vistos na Tabela 4:

Tabela 4 – Comparativo de Resultados no Conjunto de Teste

	Class: pedestrian			
	Precision	*Recall*	mAP50	mAP50-95
Experimento 1	0,471	0,225	0,238	0,093
Experimento 2	0,494	0,283	0,288	0,116
Variação percentual	4,9%	25,8%	21,0%	24,7%
	Class: people			
	Precision	*Recall*	mAP50	mAP50-95
Experimento 1	0,455	0,097	0,128	0,042
Experimento 2	0,434	0,207	0,196	0,067
Variação percentual	-4,6%	113,4%	53,1%	59,5%

Fonte: Autor

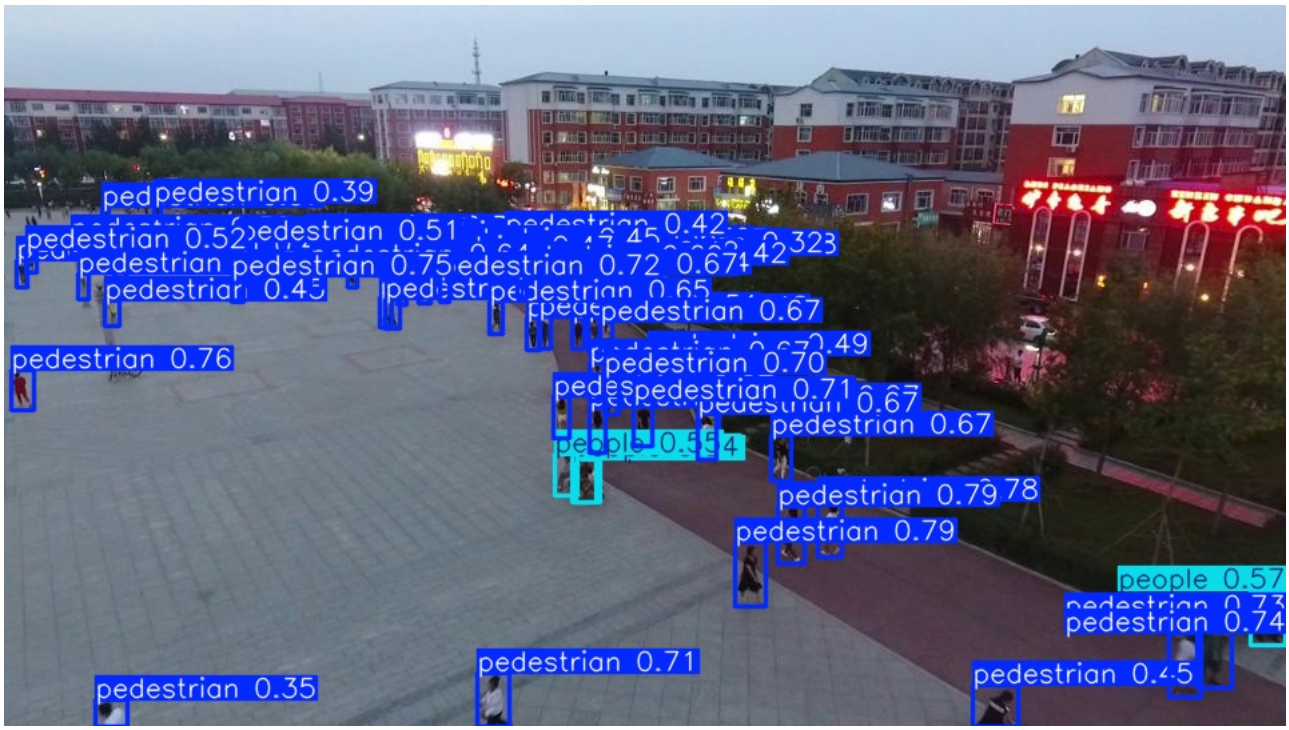
Os resultados no Conjunto de Teste comprovam a hipótese de que reduzir o número de classes contribuiu para melhorar a performance do modelo.

O resultado gerado pelo modelo, com as *bounding boxes* e o nível de confiança do modelo na classificação de cada objeto, pode ser visto na Figura 33 que é uma das imagens presentes no conjunto de teste.

Figura 32 – Imagem do conjunto de teste

Fonte: VisDrone dataset

Figura 33 – Contagem de pessoas - Imagem do conjunto de teste



Fonte: Autor

O número total de pessoas (classes '*pedestrian*' e '*people*') identificados nessa imagem foram de 65, sendo: 61 objetos da classe '*pedestrian*' e 4 objetos da classe '*people*'.

A saída gerada pelo modelo resultante do Experimento 2, para uma única imagem, fornece informações sobre os objetos detectados e o tempo de execução. A saída inclui o índice da imagem, a resolução após o pré-processamento (384x640), a contagem de objetos classificados como '*pedestrian*' (61) e '*people*' (4), e o tempo total de processamento (72.0ms). Detalhes sobre o tempo de execução, como tempo de pré-processamento (3.0ms), inferência (72.0ms) e pós-processamento (15.0ms), também são fornecidos. Adicionalmente, a saída informa que o processamento foi realizado em um *batch* de tamanho 1, utilizando imagens com 3 canais de cor (RGB) e resolução 384x640 após o redimensionamento. Os tempos de processamento podem variar de acordo com o sistema operacional e hardware utilizados. Os resultados apresentados foram obtidos em um sistema operacional Windows 11 Pro, com 64 GB de memória RAM, processador Intel i5-9600K 3.70GHz e GPU Nvidia RTX 4070Ti.

Ao comparar os resultados da detecção do modelo com o número real de instâncias na imagem de teste, foram observadas divergências. Enquanto a imagem continha 84 instâncias da classe '*pedestrian*', o modelo identificou apenas 61. Da mesma forma, para a classe '*people*', o modelo contabilizou 4 instâncias, sendo que haviam 3. Essa discrepância entre os valores reais e os valores previstos (contagem) evidencia a necessidade de aprimorar a precisão do modelo.

O próximo experimento se concentrou em realizar ajustes nos hiperparâmetros do modelo.

4.3 Experimento 3

Com o objetivo de melhorar os resultados, foi conduzido um terceiro experimento, no qual a função *model.tune()* foi empregada para ajustar os hiperparâmetros do YOLOv8n. A ferramenta realizou múltiplas iterações, ajustando parâmetros como a taxa de aprendizado inicial (lr0=0.00919), a taxa final de aprendizado (lrf=0.01068), o *momentum* (0.92024) e outros relacionados à otimização e à augmentação de dados. Apesar das diversas combinações testadas, os hiperparâmetros resultantes, embora otimizados pelo algoritmo, não proporcionaram resultados melhores em relação ao Experimento 2, que utilizou os hiperparâmetros padronizados. Esses resultados indicam que a otimização de hiperparâmetros pode ser um processo complexo e que, em alguns casos, a combinação padrão, frequentemente utilizada em modelos pré-treinados como o YOLOv8, pode já ser suficientemente eficaz para o problema em questão.

Os resultados para o Conjunto de Teste obtidos com o Experimento 3 e comparados com o Experimento 2 podem ser verificados na Tabela 5.

Tabela 5 – Comparativo de resultados - Ajustes em Hiperparâmetros

Experimento 2 (hiperparâmetros default)						
Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
pedestrian	1197	21006	0.494	0.283	0.288	0.116
people	797	6376	0.434	0.207	0.196	0.067
Experimento 3 (hiperparâmetros customizados)						
Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
pedestrian	1197	21006	0.472	0.281	0.279	0.113
people	797	6376	0.423	0.201	0.190	0.065

Fonte: Autor

4.4 Experimento 4

Com o objetivo de investigar o impacto do tamanho da imagem de entrada na performance do modelo, neste experimento o parâmetro '*imgsz*' foi ajustado, passando de 640 para 736 pixels (múltiplos de 32).

Com o aumento de tamanho da imagem de entrada, o modelo processa uma quantidade maior de dados, o que resulta em um aumento da complexidade computacional e do tempo de treinamento. Para mitigar esse impacto, o número de épocas foi reduzido de 300 para 100. Essa decisão foi tomada considerando o *trade-off* entre a precisão e o tempo de treinamento. Os resultados entre os experimentos podem ser analisados na Tabela 6.

Tabela 6 – Comparativo de Resultados - Conjunto de Teste

Experimento 1 (todas as classes)						
Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
pedestrian	1197	21006	0.471	0.225	0.238	0.093
people	797	6376	0.455	0.097	0.128	0.042
Experimento 2 (hiperparâmetros default)						
Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
pedestrian	1197	21006	0.494	0.283	0.288	0.116
people	797	6376	0.434	0.207	0.196	0.067
Experimento 3 (hiperparâmetros customizados)						
Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
pedestrian	1197	21006	0.472	0.281	0.279	0.113
people	797	6376	0.423	0.201	0.190	0.065
Experimento 4 ('imgsz' e 'epoch' ajustados)						
Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
pedestrian	1197	21006	0.473	0.318	0.310	0.126
people	797	6376	0.441	0.212	0.207	0.075

Fonte: Autor

Ao comparar os resultados do segundo e quarto experimentos, é possível observar uma melhora na performance do modelo após o ajuste do parâmetro '*imgsz*'. A classe

'*pedestrian*' apresentou um aumento de 12% na métrica de *recall*, enquanto a classe '*people*' obteve uma pequena melhoria nas métricas de precisão e *recall*. Nas métricas mAP50 mAP50-95, ambas as classes tiveram resultados melhores. Esses resultados sugerem que o aumento do tamanho da imagem contribuiu para uma melhor generalização do modelo e uma maior capacidade de detectar objetos menores e mais detalhados.

A redução do número de épocas, por sua vez, foi uma medida necessária para mitigar o aumento do tempo de treinamento ocasionado pelo aumento do tamanho da imagem.

Essa redução, para apenas um terço do experimento anterior, demonstrou ser uma estratégia eficaz para otimizar o tempo de treinamento, sem comprometer significativamente a performance do modelo. Os resultados obtidos indicam que, mesmo com um tempo de treinamento reduzido, foi possível alcançar uma leve melhoria nos indicadores de desempenho.

4.5 Experimento 5

Neste experimento foram utilizadas duas abordagens para melhorar o desempenho do modelo YOLOv8n: aumentar o tamanho da imagem de entrada (parâmetro '*imgsz*') e unificar as classes '*pedestrian*' e '*people*' em uma única classe chamada '*people*'. O objetivo foi extrair características visuais mais detalhadas, através do aumento do tamanho da imagem de entrada e simplificar o treinamento e aumentar a precisão, com a unificação das classes.

Os resultados do Experimento 5 para o Conjunto de Teste podem ser verificados na Tabela 7:

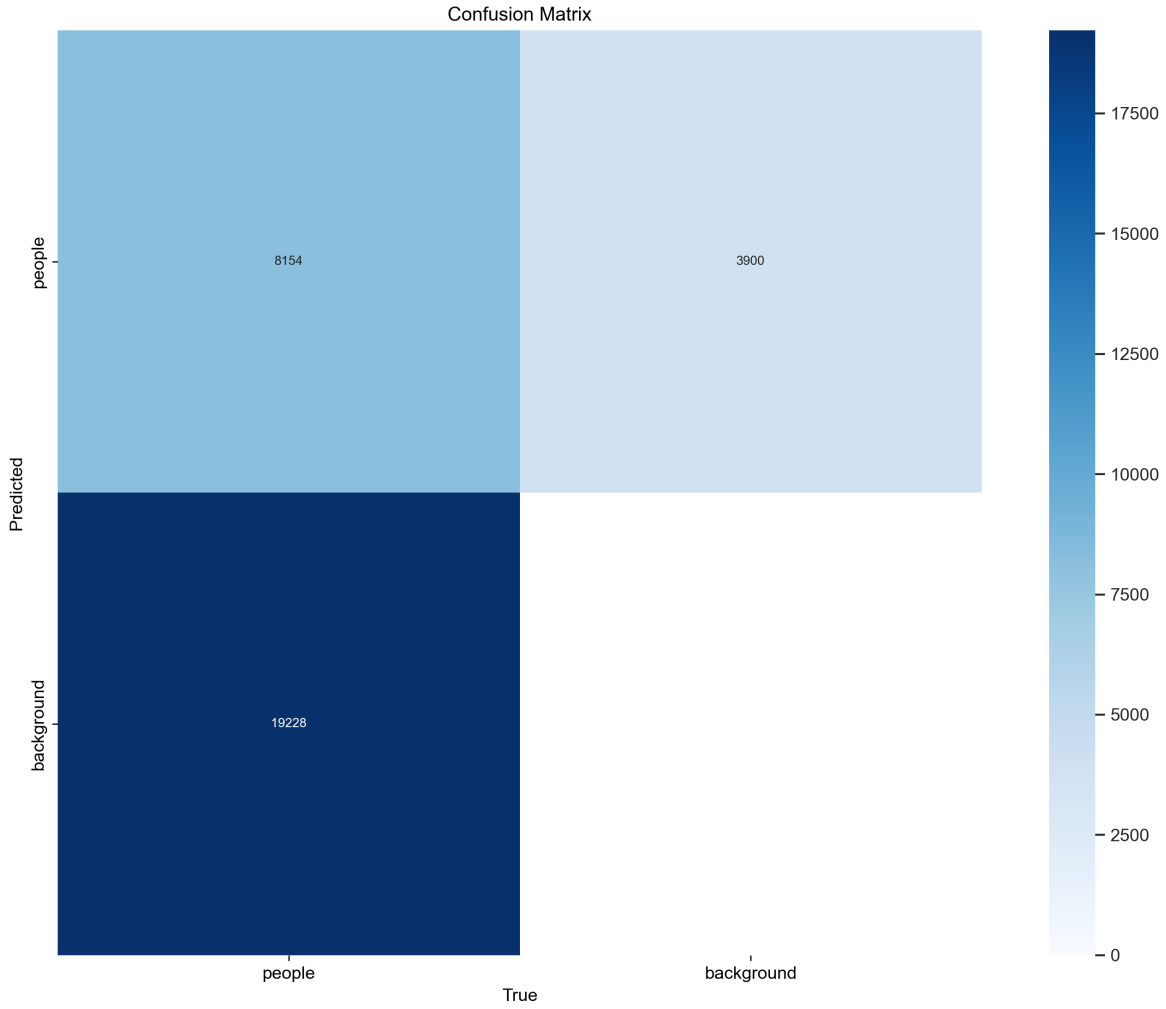
Tabela 7 – Experimento 5 - Resultados no Conjunto de Teste

Class	*Images*	*Instances*	*Precision*	*Recall*	mAP50	mAP50-95
people	1610	27382	0.55	0.306	0.323	0.126

Fonte: Autor

O resultado da Matriz de Confusão pode ser visto na Figura 34:

Figura 34 – Experimento 5 - Conjunto de Teste



Fonte: Autor

A análise da matriz de confusão apresentada na Figura 34, demonstra que o modelo do Experimento 5 classificou corretamente 8.154 instâncias como '*people*'. No entanto, 3.900 instâncias foram classificadas como sendo da classe '*people*' erroneamente, quando na verdade são o fundo ('*background*') da imagem, ou seja, é uma *bounding box* que não contém nenhum objeto, classe. Isso indica uma tendência do modelo a gerar falsos negativos. Essa dificuldade pode estar relacionada à escala reduzida das pessoas nas imagens aéreas e às variações de iluminação e fundo presentes nas imagens, tornando necessário aprimorar a capacidade do modelo em discernir entre pessoas e o ambiente complexo que as cerca. Apesar de apresentar um mAP50 de 0.323, o baixo valor de *recall* (0.306) para a classe "*people*", considerando um total de 27.382 instâncias em 1.610 imagens, reforça a necessidade de investigar estratégias para minimizar os falsos negativos e aumentar a sensibilidade do modelo na detecção de pessoas.

O resultado do Experimento 5 em uma única imagem do Conjunto de Teste pode ser verificado na Figura 35:

Figura 35 – Contagem de pessoas - Experimento 5

Fonte: Autor

O número total de pessoas (classe '*people*') identificados nessa imagem foram de 78, de um total de 87 pessoas. Uma diferença de -10%.

O tempo de processamento foi 25.0ms (contra 72.0ms do Experimento 2). O tempo de pós-processamento foi de 4.0ms (contra 15.0ms do Experimento 2) e o tempo de inferência foi de 25.0ms (contra 72.0ms do Experimento 2). Isso demonstra que manter uma única classe também contribuiu para tempos mais reduzidos de processamento do modelo.

O Conjunto de Teste é composto de 1.610 imagens, totalizando 27.382 instâncias rotuladas como classe '*people*'. O modelo do Experimento 5 ao ser utilizado em todas as imagens do Conjunto de Teste, obteve a contagem de um total de 12.424 pessoas. Isso representa apenas 45% do total de pessoas presentes em todas as imagens.

Esse resultado sugere que o modelo ainda enfrenta dificuldades na detecção precisa de pessoas em imagens aéreas.

5 CONCLUSÕES

O Experimento 5, com a estratégia de aumentar o tamanho das imagens de entrada e a unificação das classes "*pedestrian*"e "*people*", demonstrou avanços, mesmo tendo sido treinado em uma quantidade menor de épocas, mas ainda necessita de aprimoramentos. A hipótese de que essas alterações aumentariam a capacidade do modelo em detectar pessoas em diferentes escalas e simplificariam o aprendizado se mostrou parcialmente válida, com uma leve melhora na precisão e *recall* para a classe "*people*". Entretanto, a contagem final de pessoas no conjunto de teste (45% do total real) evidencia a necessidade de explorar novas estratégias para melhorar a acurácia do modelo.

Uma das estratégias é treinar novos modelos com maior tamanho das imagens de entrada, avaliando o custo computacional para esse treinamento. Outra abordagem é aumentar a quantidade de imagens, especialmente aquelas que retratam cenários complexos com alta densidade de pessoas, como os encontrados em aglomerações urbanas. Isso pode contribuir para uma detecção mais robusta. Adicionalmente, a aplicação de técnicas de aumento de dados, como rotações, espelhamentos e variações de brilho, pode auxiliar na generalização do modelo, tornando-o menos suscetível a variações nas condições de imagem. Permitir maior número de iterações entre os hiperparâmetros, através da função '*model.tune()*' pode proporcionar a obtenção de hiperparâmetros customizados que obtenham melhores resultados.

Outra abordagem é explorar as demais arquiteturas e hiperparâmetros do YOLOv8, testando modelos maiores como o YOLOv8l ou YOLOv8x, que possuem mais parâmetros e níveis de complexidade e poder de processamento. Isso pode levar a resultados mais precisos.

A combinação dessas estratégias, juntamente com a análise aprofundada das características do conjunto de dados e das saídas do modelo, fornecerá insights valiosos para o desenvolvimento de um sistema de detecção e contagem de pessoas mais preciso e confiável.

REFERÊNCIAS

ANSARI, S. **Building Computer Vision Applications Using Artificial Neural Networks**: With examples in opencv and tensorflow with python. [*S.l.: s.n.*], 2023. 2. Disponível em: https://link.springer.com/book/10.1007/978-1-4842-9866-4#about-this-book. Acesso em: 2024.04.12.

ANZARUDDIN, M. *et al.* Unmanned aerial vehicles for crowd surveillance. *In*: **2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)**. [*S.l.: s.n.*], 2023. p. 200–205.

CARVALHO, A. C. P. L. F. d.; MENEZES, A. G.; BONIDIA, R. P. **Ciência de Dados: Fundamentos e Aplicações**. 1ª ed.. ed. [*S.l.: s.n.*]: LTC, 2024.

CHENG, Z. *et al.* Recognizing human group action by layered model with multiple cues. **Neurocomputing**, v. 136, p. 124–135, 2014. ISSN 0925-2312. Disponível em: https://www.sciencedirect.com/science/article/pii/S0925231214001738.

CONCI, A. **MODELO HSV**.

GAZZIRO, P. D. M. Método de otsu. 2013. Disponível em: http://wiki.icmc.usp.br/images/b/bb/Otsu_e_derivadas.pdf.

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

GéRON, A. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn e Tensorflow**: Conceitos, ferramentas e técnicas para a construção de sistemas inteligentes. [*S.l.: s.n.*], 2021. Version 2. Disponível em: https://www.oreilly.com/library/view/maos-a-obra/9788550803814/. Acesso em: 12 abr. 2024.

ILYAS, N.; SHAHZAD, A.; KIM, K. Convolutional-neural network-based image crowd counting: Review, categorization, analysis, and performance evaluation. **Sensors**, v. 20, n. 1, 2020. ISSN 1424-8220. Disponível em: https://www.mdpi.com/1424-8220/20/1/43.

JACOBS, H. To count a crowd. **Columbia Journalism Review**, Columbia University, Graduate School of Journalism, v. 6, n. 1, p. 37, 1967.

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

JOCHER, G.; MUNAWAR, R.; AYUSHEXEL. **Ultralytics YOLOv8 Documentação**. 2024. https://docs.ultralytics.com/pt. Disponível em: https://docs.ultralytics.com/pt.

KEITA, Z. **YOLO Object Detection Explained**. 2022. https://www.datacamp.com/blog/yolo-object-detection-explained. Acesso em: 27/04/2024.

KHAN, M. A.; MENOUAR, H.; HAMILA, R. Revisiting crowd counting: State-of-the-art, trends, and future perspectives. **Image and Vision Computing**, v. 129, p. 104597, 2023. ISSN 0262-8856. Disponível em: https://www.sciencedirect.com/science/article/pii/S0262885622002268.

LAKSHMANAN, V.; GöRNER, M.; GILLARD, R. **Practical Machine Learning for Computer Vision**. [*S.l.: s.n.*]: O'Reilly Media, Inc., 2021. ISBN 9781098102364.

LIN, T.-Y. *et al.* **Microsoft COCO: Common Objects in Context**. 2015.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, p. 115–133, 1943.

REDMON, J. *et al.* You only look once: Unified, real-time object detection. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2016. p. 779–788.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, v. 65 6, p. 386–408, 1958. Disponível em: https://api.semanticscholar.org/CorpusID:12781225.

Sá, Y. V. d. A. **Desenvolvimento de aplicações IA: robótica, imagem e visão computacional**. [*S.l.: s.n.*], 2021. Disponível em: https://app.minhabiblioteca.com.br/#/books/9786589881681/. Acesso em: 11 abr. 2024.

VASAVI, G. *et al.* People counting based on yolo. *In*: **2023 4th IEEE Global Conference for Advancement in Technology (GCAT)**. [*S.l.: s.n.*], 2023. p. 1–6.

WADHWA, M. *et al.* Comparison of yolov8 and detectron2 on crowd counting techniques. *In*: **2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS)**. Istanbul, Turkiye: [*S.l.: s.n.*], 2023. p. 1–6.

ZHANG, Y. *et al.* Single-image crowd counting via multi-column convolutional neural network. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.: s.n.*], 2016. p. 589–597.

ZHU, P. *et al.* Detection and tracking meet drones challenge. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 44, n. 11, p. 7380–7399, 2021.

ZHU, P. *et al.* **VisDrone-Dataset: Drone-Based Detection and Tracking Dataset**. [*S.l.: s.n.*]: IEEE, 2021. 7380–7399 p. https://github.com/VisDrone/VisDrone-Dataset.