UNIVERSIDADE FEDERAL DO MARANHÃO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DETECÇÃO E CLASSIFICAÇÃO DE DERRAMAMENTO DE ÓLEO NA SUPERFÍCIE OCEÂNICA BASEADA EM APRENDIZAGEM PROFUNDA VIA ALGORITMO YOLO

TAYNÁ CRISTINA SOUSA SILVA

ORIENTADOR: PROF. DR. JOÃO VIANA DA FONSECA NETO

São Luís - MA
Março/2024

UNIVERSIDADE FEDERAL DO MARANHÃO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DETECÇÃO E CLASSIFICAÇÃO DE DERRAMAMENTO DE ÓLEO NA SUPERFÍCIE OCEÂNICA BASEADA EM APRENDIZAGEM PROFUNDA VIA ALGORITMO YOLO

TAYNÁ CRISTINA SOUSA SILVA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Maranhão, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica, área de concentração: Automação e Controle.

Orientador: Pr. Dr. João Viana da Fonseca Neto

São Luís – MA
Março/2024

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Silva, Tayná Cristina Sousa.
Detecção e Classificação de Derramamento de Óleo na Superfície Oceânica Baseada em Aprendizagem Profunda via Algoritmo Yolo / Tayná Cristina Sousa Silva. - 2024.
75 f.

Orientador(a): João Viana da Fonseca Neto.
Dissertação (Mestrado) - Programa de Pós-graduação em Engenharia Elétrica/ccet, Universidade Federal do Maranhão, São Luís, Maranhão, 2024.

1. Algoritmo YOLO. 2. Aprendizagem Profunda. 3. Classificação. 4. Derramamento de óleo. 5. Detecção. I. Fonseca Neto, João Viana da. II. Título.

UNIVERSIDADE FEDERAL DO MARANHÃO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DETECÇÃO E CLASSIFICAÇÃO DE DERRAMAMENTO DE ÓLEO NA SUPERFÍCIE OCEÂNICA BASEADA EM APRENDIZAGEM PROFUNDA VIA ALGORITMO YOLO

TAYNÁ CRISTINA SOUSA SILVA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Maranhão, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica, área de concentração: Automação e Controle.

Aprovado em ___ de _____ de _____.

Membros da Banca:

Prof. Dr. João Viana da Fonseca Neto

(Orientador – UFMA)

Prof. PhD. Allan Kardec Duailibe Barros Filho

(Examinador Interno - UFMA)

Prof. Dr. Roberto Célio Limão de Oliveira

(Examinador Externo - UFPA)

São Luís - MA
Março/2024

Dedico este trabalho aos meus pais,

Wágner Rodrigues da Silva e Kelma Karine Sousa Silva.

AGRADECIMENTO

Agradeço primeiramente a Deus, pois sem Ele, nada na minha vida faz sentido. Agradeço ao meu orientador João Viana da Fonseca Neto pela orientação, pela paciência e por não ter desistido de mim.

Agradeço ao Programa de Recursos Humanos da ANP, pelo apoio financeiro, sou grata pelas oportunidades do programa.

Agradeço aos meus colegas de laboratório pelas horas descontraídas – a hora do cafezinho; com vocês, o caminho até aqui, foi mais leve. Agradeço em especial à Monik Silva, Evandro Martins, Victor Guimarães e a Bianca Fontenele, por todos os conhecimentos repassados, por nunca terem deixado eu desistir, pelas conversas amigas e engraçadas. Vocês se tornaram verdadeiros amigos!

Agradeço aos meus pais, Kelma Karine e Wágner Rodrigues, vocês são minha base, sei que sempre posso contar com vocês para tudo em minha vida! Agradeço a minha tia Lorena Karine, você é um espelho de mulher, de pesquisadora, amo a sua dedicação pela ciência. Agradeço pelo apoio e conversas descontraídas ao longo do mestrado.

Agradeço ao meu noivo, Renato Guimarães, você é um exemplo para mim, agradeço por nunca ter desistido de mim, pelo apoio e força durante esses anos, você é meu grande incentivador.

Agradeço a todos que contribuíram, de alguma maneira, para a concretização deste trabalho.

"Olho nenhum viu, ouvido nenhum ouviu, mente nenhuma imaginou o que Deus preparou para aqueles que o amam"

1 Coríntios2:9

RESUMO

Devido ao crescimento global, a importância da economia mundial relacionado ao uso de matérias primas advindas do petróleo vem aumentando cada vez mais. Em decorrência a esse crescimento, a importância de temas relacionados com a preocupação socioambiental e econômico vem se destacando no âmbito científico. Consequentemente, a ocorrência de incidentes de derrames de óleos em superfície oceânica demanda o desenvolvimento de metodologias para mitigar os impactos causados pela problemática nas áreas atingidas. Com a disponibilidade de satélites equipados com Radar de Abertura Sintética, é viável monitorar, detectar e classificar os derramamentos de petróleo e seus derivados no mar. Neste trabalho é apresentada uma proposta de metodologia baseada em aprendizagem profunda, especificamente por meio da família do algoritmo YOLO. Portanto, de acordo com os experimentos realizados através da base de dados obtida via radar e disponibilizada pela missão SENTINEL-1, durante os testes na fase de validação para a YOLOv8 nano, *small* e *medium*, observou-se um melhor desempenho para o *medium,* com métricas de precisão, mAP-50 e mAP50-90 são equivalentes à 0.891%, 0.85% e 0.716%, respectivamente. Já o resultado na fase de teste atingiu um nível de confiança, de acordo com a métrica IoU (*Intersection over Union*) acima de 70% dos objetos classificados como manchas de óleos.

Palavras-chave: Derramamento de óleo, Aprendizagem Profunda, Algoritmo YOLO, Detecção e Classificação.

ABSTRACT

Due to global growth, the importance of the world economy in relation to the use of petroleum-based raw materials has increased. As a result of this growth, the importance of issues related to socio-environmental and economic concerns has been highlighted in the scientific field. Consequently, the occurrence of oil spill incidents on the ocean surface requires the development of methodologies to mitigate the impact caused by the problem in the affected areas. With the availability of satellites equipped with Synthetic Aperture Radar, it is possible to monitor, detect and classify spills of oil and its derivatives at sea. This dissertation presents a proposal for a methodology based on deep learning, specifically using the YOLO family of algorithms. Therefore, according to the experiments carried out using the dataset obtained via radar and provided by the SENTINEL-1 mission, during the tests in the validation phase for YOLOv8 nano, small and medium, better performance was observed for medium, with accuracy metrics, mAP-50 and mAP50-90 equivalent to 0.891%, 0.85% and 0.716%, respectively. The result in the test phase reached a confidence level, according to the IoU (Intersection over Union) metric, of more than 70% of the objects classified as oil slicks.

Keywords: *Oil Spill, Deep Learning; YOLO algorithm; Detection and Classification.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANP – Agência Nacional de Petróleo, Gás Natural e Biocombustíveis,

AUV - Veículo Submarino Autônomo,

CNN – Rede Neural Convolucional,

DL – *Deep Learning,*

FLOPs - Floating-point Operations per Second,

FN – Falso Negativo,

FP - Falso Positivo,

IOU - *Intersection over Union,*

ITOPF - *International Tanker Owners Pollution Federation Limited,*

LDA - Análise Discriminante Linear,

MAP - *mean Average Precision*,

MLP - *Multilayer Perceptron,*

RELU - Rectified Linear Unit,

RGB – Red, Green e Blue,

RNA – Rede Neural Artificial,

RSSF - Redes de Sensores Sem Fio,

SAR - *Synthetic Aperture Radar*,

SILU - *Sigmoid Linear Unit,*

SPPF - *Spatial Pyramid Pooling – Fast*,

VANT - Veículo Aéreo Não Tripulado,

VN – Verdadeiro Negativo,

VP – Verdadeiro Positivo,

YOLO – You Only Look Once.

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Neste capítulo é apresentado uma contextualização sobre a dissertação desenvolvida. Os objetivos, estado da arte, justificativa e a motivação que levaram a construção dessa dissertação são expostos. Após sintetizar a problemática, a organização do trabalho é apresentada em capítulos, onde são exibidos o referencial teórico, problema-solução, metodologia, resultados, discussões, conclusões e anexo.

1.1 Contextualização

Com o advento do crescimento global, a demanda de produção e transporte do petróleo e seus derivados tornou-se em uma grande competitividade mundial, estimulando empresas a buscarem estratégias comerciais, visando assim, um resultado satisfatório de crescimento econômico.

A extração global de hidrocarbonetos, com foco no petróleo, começou por volta do século XIX e tem continuado a aumentar desde então. De acordo com as informações fornecidas pela Agência Nacional do Petróleo, Gás Natural e Biocombustíveis (ANP), o Brasil produziu um total de 161,703 bilhões de litros (equivalente a 1,017 bilhão de barris) de petróleo em 2019 (ANP, 2020).

Mediante este cenário, Silva (2004), relata em seu trabalho, que a crescente industrialização vem ocasionando grandes poluições ao meio ambiente,

principalmente no meio aquático, pois há um fluxo maior de navios petroleiros em rotas. No cenário brasileiro, não é diferente, também, já existem ocorrências de derramamentos de petróleos, óleos brutos e derivados, causando impactos ambientais, sociais e econômicos.

Conforme mencionado por Pena et al., (2020), um incidente ocorrido em 30 de agosto de 2019 na costa brasileira resultou em um derramamento de petróleo que afetou uma extensa área de 4.334 km nas regiões Nordeste e Sudeste do país. Este acidente foi considerado o maior já registrado no Brasil e também o derramamento de petróleo mais extenso documentado no mundo. As primeiras observações ocorreram nas áreas costeiras dos estados de Pernambuco e Paraíba. Até o dia 22 de novembro de 2019, haviam registros do derramamento em um total de 11 estados nas regiões do Nordeste e Sudeste do Brasil.

Na mesma vertente, o Relatório Anual de Segurança Operacional da ANP em 2022, mostrou que o volume de óleo derramado no alto mar em decorrência de incidentes em exploração e produção de petróleo e gás natural (E&P) apresentou cerca de $217{,}16\ \mathrm{m}^3$ nesse mesmo ano, sendo o segundo maior registrado durante o período de 2013 a 2022, pois o primeiro registrado ocorreu em 2019, conforme descrito no parágrafo anterior (ANP, 2022).

A continuidade e a abrangência desses fenômenos não podem ser atribuídas ao acaso, uma vez que indicam a probabilidade de ocorrer devido a modelos insustentáveis tanto em termos sociais quanto técnicos.

Conforme observado pela *International Tanker Owners Pollution Federation Limited* - ITOPF (2020), embarcações de grande porte têm o potencial de causar desastres ambientais de grande escala, como acidentes marítimos que resultam em derramamentos de óleo, vazamentos de dutos em operações *offshore* e a liberação de resíduos oleosos durante a limpeza de tanques de navios petroleiros, bem como durante operações de carga e descarga. Esses eventos têm o potencial de causar consequências significativas e muitas vezes irreversíveis para o meio ambiente, representando uma ameaça considerável para a ecologia marinha e os ecossistemas costeiros (ITOPF, 2020).

Diante do que foi discutido nos parágrafos anteriores, para minimizar os efeitos causados por esses desastres, a pesquisadores, engenheiros e ambientalistas vem

desenvolvendo técnicas de monitoramento, detecção e classificação desses poluentes.

Dentro deste contexto, o propósito desta dissertação é conduzir uma comparação do desempenho de algoritmos baseados de aprendizagem profunda, no que se refere à detecção de manchas de óleo em ambientes aquáticos por meio de imagens SAR.

1.2 Justificativa

Esse trabalho tem como preocupação os impactos ambientais gerados por derrames de óleos na costa marítima, pois como visto na Seção 1.1, suas consequências geram uma ação negativa nos aspectos ambientais, sociais e econômicos da área atingida.

Dessa forma, as consequências decorrentes dessas atividades têm motivado as empresas a utilizarem técnicas de monitoramento, detecção e classificação capazes de minimizar tais efeitos ao meio ambiente de forma mais rápida de solucionar tal problemática.

O que fomenta a necessidade desse trabalho, um estudo comparativo entre algoritmos de detecção de derrames de óleos em superfície oceânica, facilitando que as empresas, organizações de preservação ao meio ambiente e de interesse políticos elaborem planos de contingências capazes de evitar maiores problemas ambientais, como o fato ocorrido no Brasil nos anos de 2019 e 2022.

1.3 Motivação

Visto a importância da produção e do consumo de petróleo pelo mundo, diversos incidentes são registrados pelos órgãos competentes, de acordo com o relatório anual da ANP, tanto em decorrência da exploração e como também em rotas de navios petroleiros. Este trabalho tem como principal motivação a preocupação socioambiental, sendo desenvolvido um estudo comparativo de algoritmos baseado

em aprendizagem profunda, com a finalidade de detectar e classificar manchas de óleos e em decorrências desses incidentes.

Portanto, este trabalho propõe o desenvolvimento de uma metodologia viável de mitigar as problemáticas ambientais causados por danos humanos, como forma de alertar órgãos competentes responsáveis por implementações de planos de ações rápida e contínua nesses ambientes que são monitorados por satélites, rede de sensores entre outros.

Já o que fomenta a motivação científica é a oportunidade de estudar, desenvolver, analisar algoritmos, técnicas e metodologias utilizadas para o monitoramento, detecção e classificação nesses ambientes onde a problemática é exposta, assim como a possibilidade da implementação da metodologia para sistemas embarcados.

1.4 Objetivos

Os objetivos desta dissertação estão divididos em objetivo geral e objetivos específicos. Dessa maneira, distinguem-se os pontos a serem cumpridos ao longo da dissertação.

1.4.1 Objetivo Geral

Aplicar e analisar o desempenho de algoritmos de *deep learning* para detecção de manchas de óleo em superfícies oceânicas obtidos através de monitoramento por satélites equipados com radar de abertura sintética.

1.4.2 Objetivos Específicos

Para atingir o objetivo geral, foram estabelecidos os seguintes objetivos específicos:

- Estudar os assuntos abordados em relação aos derramamentos de óleos em ambientes aquáticos e da preservação ao meio ambiente;

- Aplicar o algoritmo *You Only Look Once version* V8 (YOLOv8) para a detecção classificação de manchas de óleo na superfície oceânica;
- Analisar o desempenho do algoritmo proposto;
- Analisar o custo computacional do algoritmo proposto;
- Realizar uma análise comparativa dos algoritmos propostos com os já existentes na literatura.

1.5 Estado da Arte

Com o avanço da globalização, a demanda por petróleo e seus derivados como matéria-prima na fabricação de diversos produtos e serviços tem aumentado cada vez mais, resultando em uma crescente necessidade logística entre rotas de navios petroleiro, exploração e produção para atender a essa demanda.

Mediante tal cenário, o Anuário Estatístico Brasileiro do Petróleo, Gás Natural e Biocombustíveis de 2023, apresenta a evolução desses setores no Brasil, com dados de 2013 a 2022. De acordo com os dados apresentados, em 2022, a produção nacional de petróleo cresceu em 4%, atingindo cerca de 3 milhões de barris/dia. Já a produção nacional de petróleo originado do pré-sal, alcançou em média 2,3 milhões de barris/dia no ano, o que equivale a 76% da produção do país. Ao que tange a exportação do petróleo brasileiro, em 2022, alcançaram o volume de 1,3 milhões de barris/dia, atingindo uma taxa de crescimento de 68,3% (ANP, 2023).

Da mesma maneira, em consequência de fatores das atividades essenciais do país, devido a sua extensão territorial, a demanda do consumo de petróleo e derivados aumentou ao longo da última década, atingindo cerca de 2,1 milhões de barris/dia em 2022 (ANP, 2022).

Nessa vertente, a *International Tanker Owners Pollution Federation Limited* - ITOPF (2020), contabilizou 10.000 incidentes em mais de 100 países no período entre 1970 a 2020, em sua maioria das vezes, foram causados por colisões, danos estruturais de navios, explosões, acidentes de navios petroleiros, encalhes e vazamentos de oleodutos em operações *offshore* (ITOPF, 2020).

Da mesma maneira, o Relatório Anual de Segurança Operacional da ANP em 2022, recebeu 2410 registros de incidentes em instalações marítimas, sendo 1244

incidentes em plataformas de produção e poços marítimos, o que corresponde 82% do total de incidentes *offshore* comunicados. Dentre esses registros, os mais comunicados às autoridades estão as descargas de óleo, de material com alto potencial de dano e descarga de fluidos de perfuração (ANP, 2022).

De acordo com Cardoso (2007), incidentes conhecidos mundialmente ao longo dos anos são destacados a seguir:

- O incidente de *Nestucca,* em Washington (EUA), no dia 23 de dezembro de 1988, derramando aproximadamente 875 m^3 de óleo pesado no mar;
- O incidente de *Exxon Valdez* nos EUA, em 24 de março de 1989, derramando cerca de 41.000 m^3 de óleo no mar;
- O incidente *Sea Empress* em Gales, no dia 15 de fevereiro de 1996, derramando aproximadamente 81.771 m^3 de óleo e de 545 m^3 combustível no mar;

Incidentes ocasionados no Brasil são mencionados a seguir:

- Em 09 de janeiro de 1978, o petroleiro *Brazilian Marina* derramou em torno de 6.000 m^3 de óleo cru nas águas do estado de São Paulo;
- Em 16 de março de 2000, o navio *Mafra IV* causou um derramamento de 7.240 m^3 de petróleo no estado de São Paulo;
- Em 18 de outubro de 2001, um navio que carregava nafta derramou mais de 350 m^3, na bacia Paranaguá – litoral paranaense.
- Em 15 de novembro de 2004, ocorreram diversas explosões no navio *Vicuña,* causando aproximadamente um derrame de 4.000 m^3 de três diferentes tipos de combustíveis nas águas do Paraná;
- Em 09 de novembro de 2006, no Porto de Santos, vazou cerca de 1 m^3 de óleo do navio *Smart.*

De acordo com um levantamento bibliográfico, existem várias abordagens que remetem aos métodos disponíveis para solucionar o exposto problema.

Conforme o trabalho de Ahmed et al. (2023), propôs em seu trabalho uma *deep learning*, conhecida como Seg-Net para produzir imagens de derramamento de óleo de alta qualidade e um discriminador Patch-GAN, utilizando apenas 50 imagens de Radar de Abertura Sintética (*Synthetic Aperture Radar* - SAR). Seu resultado foi significativo mesmo com um conjunto de dados pequeno, quando comparado com outros modelos de aprendizagem profunda.

No trabalho desenvolvido por Shaban et al. (2021), apresenta uma estrutura também baseada em aprendizagem profunda, tal método é proposto em duas etapas para detecção de derramamento de óleos. A primeira etapa classifica as manchas de óleos com uma rede neural convolucional (CNN) de 23 camadas. A segunda etapa, utiliza a rede U-Net de 5 estágios. Os resultados foram satisfatórios quando comparados com outras *deep learning.*

Em Yang et al. (2022), foi proposto um detector de objetos baseado em aprendizagem profunda, utilizando imagens SAR. O detector de imagens treinados tiveram uma precisão média de 69,10% e nos conjuntos de teste, obtiveram 68,69%. O que mostrou ser eficaz ao sistema de vigilância de contaminação responsável em sua cidade.

De acordo com o trabalho desenvolvido por Zeng e Wang (2020), apresentaram uma rede convolucional chamada de OSCNet para detecção de derrames de óleo para imagens SAR. Essa rede mostrou ser muito superior aos métodos tradicionais de aprendizado de máquina através das métricas de desempenho como *recall* e a precisão, quando comparadas.

De acordo com Sousa (2023), em sua dissertação foram desenvolvidos dois algoritmos, o primeiro foi baseado na integração da análise discriminante linear (LDA) com uma rede neural *perceptron* e o segundo, foi baseado numa *deep learning,* chamada U-net. Seus resultados também foram satisfatórios para a detecção e classificação de manchas de óleos em superfície marinha.

Além dessas técnicas utilizadas, também são necessários estudos voltados para a possibilidade de embarcar algoritmos de aprendizado de máquina, como forma de acelerar a tomada de decisões em eventuais derramamentos de óleos.

Conforme Yoo et al. (2019), os autores propuseram uma estrutura de aprendizagem profunda para embarcar em veículos automotivos. Ele mostrou uma problemática em relação à memória quando se trata de um banco de dados extenso.

Em Diana e Fanucci (2021), foi apresentada uma CNN para imagens SAR para utilização em sistemas embarcados. Esta CNN foi projetada em critérios como: tempo de inferência, consumo de energia, tamanho do algoritmo e métricas de desempenho. Embora não tenha tido um desempenho excepcional em termos de precisão com outros algoritmos, ela é compatível com aceleradores de *hardware* disponíveis no

mercado devido ao baixo uso de memória e do tamanho reduzido, como também, da execução do algoritmo diretamente a bordo.

A pesquisa desenvolvida por Maawali et al. (2019), os autores propuseram uma solução para a detecção de petróleo no mar através de uma embarcação de superfície não tripulada, onde além de detectar o petróleo, também coleta amostras da área poluída. Os experimentos mostraram resultados satisfatórios em relação a manobrabilidade e no mecanismo confiável de coleta de amostras.

1.6 Organização do Trabalho

Neste capítulo foi realizada uma contextualização do tema abordado na dissertação, assim como os objetivos, justificativa e o estado da arte.

No Capítulo 2 é apresentada a fundamentação teórica para o entendimento e compreensão do tema proposto. Ele aborda assuntos sobre redes neurais artificiais, discorrendo sobre *deep learning* e dando ênfase em algoritmos de detecção e classificação de manchas de óleos em ambientes aquáticos. Ainda, neste capítulo são expostos métodos de avaliação de dados, relacionado às métricas de desempenho do algoritmo.

No Capítulo 3 são apresentadas a descrição do problema, sua formulação e o classificador-YOLO desse problema, visando mitigar os efeitos causados por derrames de óleos em superfície oceânica.

No Capítulo 4 é abordado a descrição da família do algoritmo YOLOv8 na função de detecção de manchas de óleo, apresentando a metodologia proposta para o conjunto de dados das imagens, originadas da missão SENTINEL-1, como o pré-processamentos realizados. Também, neste capítulo são apresentados diagramas que facilitam a compreensão da metodologia utilizada.

No Capítulo 5 são apresentados os resultados obtidos através dos experimentos computacionais, de acordo com as metodologias abordadas no capítulo anterior, fomentando em uma análise comparativa dos algoritmos desenvolvidos.

Por fim, no Capítulo 6 são apresentadas as conclusões da dissertação, de acordo com a análise comparativa de algoritmos de detecção e classificação de derramamentos de óleos em ambientes aquáticos. É apresentado as contribuições da

metodologia desenvolvida, e agregando valor ao tema proposto são citadas sugestões para trabalhos futuros.

Em seguida, mostram-se as referências utilizadas durante o desenvolvimento da dissertação e o anexo sobre a arquitetura da YOLO.

Capítulo 2

PRELIMINARES

Neste capítulo é apresentado o referencial teórico dos assuntos abordados para o desenvolvimento dessa dissertação. O capítulo é organizado em cinco seções, onde em cada uma delas são expostos pontos relevantes para o entendimento do tema proposto, sendo assim, é discorrido sobre redes neuronais artificiais, rede perceptron de única camada e múltiplas camadas, deep learning, redes neurais convolucionais, algoritmo de detecção de manchas de óleos no mar, denominado You Only Look Once (YOLO) e métricas de avaliação.

2.1 Redes Neurais Artificiais (RNA)

De acordo com Goodfellow, Bengio, & Courville (2016), o ser humano sempre sentiu a necessidade de criar uma máquina com a capacidade de pensar e agir de maneira semelhante a si mesmo. Desde então criou-se uma máquina denominada como computador e em decorrência da evolução dos computadores, surgia temas como a inteligência artificial (IA) e aprendizado de máquina. Fomentando a necessidade de melhorar tais evoluções, a IA é vista como um campo próspero, sendo aplicada em resolução de atividades complexas, como em áreas de engenharias e até

mesmo em atividades simples, como em suporte em escritas, desenvolvimentos de lógicas, textos e soluções de problemas aplicados ao dia-a-dia.

Sendo assim, o ser humano e sua busca por evolução tecnológica, impulsionou temas relacionados às redes neuronais artificiais (RNA), assemelhando ao cérebro do ser humano. Dessa maneira, surgiu a necessidade de concretizar tal invenção.

Portanto, o funcionamento de uma RNA é análogo ao cérebro humano. O livro "Neural Networks for Pattern Recognition" escrito por Christopher M. Bishop (1995), define redes neurais como sistemas computacionais inspirados pelo funcionamento do cérebro humano, que são capazes de aprender a partir de dados e realizar tarefas de reconhecimento de padrões.

Por conseguinte, Hayken (2008), em seu livro relata que o cérebro humano é composto por milhares de células, conhecidas como neurônios, que por sua vez, é composta por três partes importantes: corpo celular, os dendritos e axônios. Da mesma maneira de uma RNA, sendo sua unidade básica composta por receptores, rede neural e atuadores. A Figura 2.1 é a representação de um esquema de uma RNA, através de diagrama de blocos, na qual pode-se fazer uma relação também com um sistema neural do ser humano.

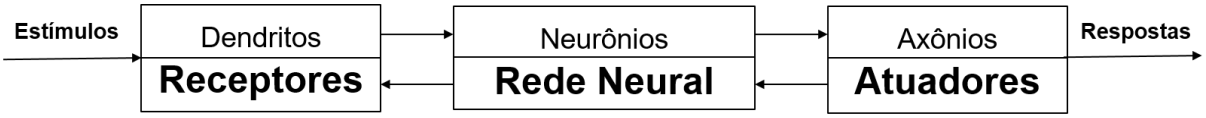

Figura 2.1: Diagrama de blocos de um sistema neural.

De acordo com a Figura 2.1, assim como nos neurônios naturais, os receptores de um neurônio artificial encaminham os estímulos para o corpo dos neurônios (rede neural). No corpo do neurônio artificial, esses estímulos são processados, e quando um certo limiar de informações é atingido, elas são enviadas para os atuadores e a partir delas, tomam as decisões mais adequadas (respostas).

De acordo com Haykin (2008), um neurônio *k* pode ser representado

$$u_k = \sum_{j=1}^{m} w_{kj} x_j, \tag{2.1}$$

sendo as entradas representadas por X_m, onde tem-se um arranjo $X = (x_1, x_2, x_3, \dots, x_m)$ e cada uma das entradas são multiplicadas pelos pesos $P =$

$(p_1, p_2, p_3, \dots, p_m)$. Há também uma entrada externa, denominada de *bias* (b_k), permite que o neurônio tenha uma saída distinta de zero.

A Equação (2.1) representa a atividade pré-sináptica e a atividade pós-sináptica é dada por

$$y_k = \varphi(u_k + b_k), \tag{2.2}$$

A saída y_k é uma mapeamento da função de ativação $(\varphi(.))$, sendo que, se a entrada combinada extrapolar um certo limite, ela restringe o intervalo da amplitude do sinal de saída (y). Portanto, a função de ativação determina a saída de um neurônio artificial. Logo, ela pode assumir diversas formas, como funções lineares e não lineares, que são valores entre 0 e 1 ou -1 e 1 (HAYKIN, 2008).

2.2 Arquiteturas de Redes Neurais Artificiais

Uma RNA é constituída por um neurônio, conhecido também por *perceptron.* A sua estrutura é formada por diversas maneiras, sendo essas conectadas por um ou mais neurônios, formando assim, diversas camadas neuronais e essas camadas são organizadas sequencialmente para processar informações de entrada e gerar resultados. Essa organização é determinante para especificar o tipo de arquitetura da rede.

2.2.1 Rede Perceptron de Única Camada

As RNA's são dispostas em camadas que podem conter diferentes quantidades de neurônios. Porém, a rede de única camada é composta apenas por um neurônio, sendo assim, torna-se a mais simples perante outras arquiteturas. Segundo Haykin (2008), ela é representada por uma camada de entrada que é projetada para a camada de saída. Esse tipo de rede é utilizada para classificação de uma ou duas classes, caso elas sejam linearmente separáveis. Portanto, os pesos sinápticos devem ser adaptados de iteração para iteração de acordo com a regra de correção de erro, denominada como algoritmo de convergência do *perceptron.*

2.2.1.1 Algoritmo de Convergência do *Perceptron*

O algoritmo de convergência do *perceptron* é um método de treinamento utilizado para ajustar os pesos sinápticos das conexões de um *perceptron*. Sua função é encontrar um arranjo de pesos que permitem que o *perceptron* diferencie duas classes de dados distintas (Haykin, 2008).

De acordo com Fausett (1994), o processo do treinamento do *perceptron* ocorre em cinco estágios:

1. **Inicialização dos pesos:** Os pesos das conexões são definidos com valor igual a zero.
2. **Ativação das entradas do perceptron:** Cada amostra é composta por um conjunto de dados, entradas. E uma saída desejada, relacionado à classe associada.
3. **Cálculo da saída:** Para cada amostra, é feito o cálculo ponderado entre as entradas e os pesos atuais, somando os produtos das entradas e pesos. Em seguida aplica a função de ativação que determina a saída do perceptron.
4. **Atualização dos pesos, se ocorrer erros:** Se a saída calculada corresponder à classe desejada, os pesos não são atualizados. Caso contrário, é utilizado este passo, para ajustar e corrigir o erro. Portanto, conforme Haykin (2008), a representação da atualização, é dada por

$$w(n+1) = w(n) + \eta[d(n) - y(n)]x(n), \qquad (2.3)$$

   sendo $w(n)$ o peso sináptico atual e $w(n+1)$ o peso sináptico da próxima iteração. A taxa de aprendizagem representado por η, varia em intervalo de $0 < \eta \leq 1$; a correção dos pesos é realizada pela diferença entre o erro, onde é representado pela parcela $[d(n) - y(n)]$, em que $d(n)$ é o sinal desejado e $y(n)$ é a saída do perceptron calculada, por fim, $x(n)$ é a entrada do algoritmo.
5. **Convergência:** Esse processo garante em convergir, encontrar uma solução que classifique corretamente as amostras de treinamento, já que as duas classes de dados são linearmente separáveis.

2.2.2 Rede Perceptron de Múltiplas Camada

Uma rede de múltiplas camadas, denominada de *Multilayer Perceptron* (MLP), é composta por mais de um *perceptron* e se caracterizam por possuírem uma camada de entrada, camadas intermediárias e uma camada de saída. De acordo com Haykin (2008), a camada intermediária, também denominada de camada oculta, passam a intermediar as informações de entrada e saída da rede. A Figura 2.2 é uma representação da estrutura de uma MLP.

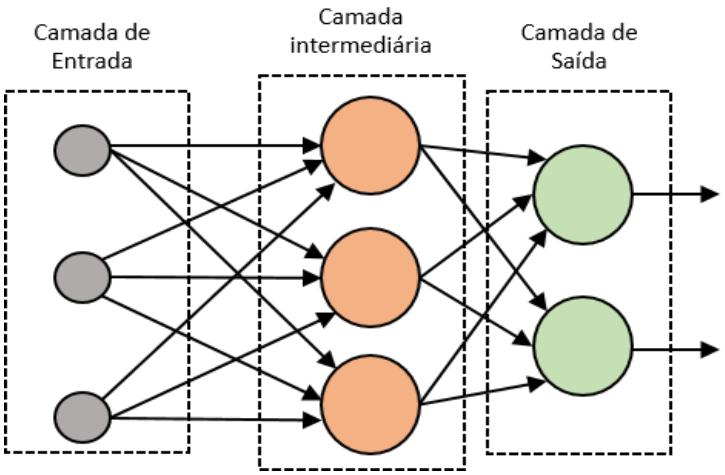

Figura 2.2: Estrutura de uma rede MLP. (Adaptado Haykin, 2008).

Como visto na Subseção 2.2.1, para a rede perceptron de uma única camada, a análise do erro é obtida através do algoritmo de convergência do *perceptron*. Segundo Braga, Ludermir e Bernarda (2011), nessa nova estrutura tem como problemática encontrar o erro das camadas intermediárias. Portanto, uma MLP consiste em um treinamento em duas etapas: propagação do sinal e a retropropagação do erro.

2.2.2.1 Algoritmo de Backpropagation

De acordo com Haykin (2008), para a propagação do sinal ou *forward pass*, um conjunto de dados padrão é aplicado na rede, passando de camada por camada, gerando uma resposta real em sua camada de saída. Cada neurônio em uma camada recebe os valores de saída dos neurônios da camada anterior, realiza cálculos ponderados com os pesos associados às suas conexões e, em seguida, aplica uma função de ativação para produzir sua saída ativada. Esse processo é repetido em todas as camadas até que a saída final seja obtida. Já na retropropagação do erro ou

backward pass, há uma comparação com a resposta real e a resposta desejada. Os pesos são atualizados conforme os erros encontrados, de modo a minimizar o erro na próxima iteração do treinamento.

Como a MLP é treinada de forma supervisionada, para o algoritmo de *backpropagation* ser realizado de forma correta, é apresentado um padrão de entrada, assim como, uma resposta de saída ideal para esse padrão.

Baseado nesses conceitos, para a formulação do algoritmo de *backpropagation,* o sinal do erro na saída do neurônio *j* em uma iteração *n,* sendo $d_j(n)$ a resposta desejada na camada de saída que é definida por

$$e_j(n) = d_j(n) - y_j(n). \tag{2.4}$$

O valor instantâneo do erro quadrático para o neurônio *j* é dado por

$$e_j = \frac{1}{2}e^2, \tag{2.5}$$

A soma dos erros quadráticos é definida por

$$\varepsilon(n) = \frac{1}{2}\sum_{j\epsilon S} e_j(n)^2, \tag{2.6}$$

sendo S o conjunto de neurônios da camada de saída.

Considerando o número total de padrões de treinamentos como N, o objetivo é do algoritmo de treinamento é minimizar o erro $\varepsilon_{méd}$, sendo assim, o erro quadrático médio é definido por

$$\varepsilon_{méd} = \frac{1}{N}\sum_{n=1}^{N} \varepsilon(n). \tag{2.7}$$

2.3 Deep Learning (DL)

Com o algoritmo de *backpropagation* tornou viável o treinamento de camadas intermediárias de uma rede neural artificial, permitindo adicionar inúmeras camadas entre as camadas de entrada e de saída. A partir dessa consideração, surgiu o novo

conceito de *deep learning*, conhecidas como redes neurais de camadas profundas. Esse novo conceito é utilizado em realizar tarefas que antes eram impossíveis de realizar, como por exemplo, reconhecer objetos em uma imagem.

2.3.1 Redes Neurais Convolucionais (CNNs)

As CNNs, também conhecidas como ConvNets, fazem parte de uma área da aprendizagem profunda, sua origem é baseada no cérebro humano, voltada para o desenvolvimento do campo de visão computacional, para detectar e classificar imagens. Sua característica importante está relacionada ao número de arquiteturas que podem ser configuradas.

De acordo com Guo et al. (2018), as CNNs são redes neurais de várias camadas que, em comparação com redes neurais convencionais, utilizam neurônios de aprendizado individuais, usam matrizes responsáveis pela filtragem, extraindo características utilizadas para aprender padrões e detalhes de imagens.

Uma CNN procura, ao longo do seu processo de treinamento, transformar gradualmente um sinal de entrada. Ela começa detectando elementos simples no início e, com o tempo, aprende a identificar detalhes específicos relacionados à tarefa que precisa realizar. Esse processo de aprendizado se baseia na ideia de decompor conceitos complexos em elementos mais básicos. A CNN realiza essa transformação gradual através da busca e ajuste de um conjunto de filtros de convolução. A escolha desses filtros é influenciada tanto pelos dados de treinamento disponíveis quanto pela estrutura geral da rede neural.

De acordo Almeida (2019), o funcionamento de uma CNN segue as seguintes etapas: treinamento, entrada, primeira camada, camada mais altas, última camada e saída. Na etapa de treinamento, a CNN é alimentada por inúmeras imagens de diversas classes e sua função é aprender cada uma das imagens. Após essa etapa, a rede pré-treinada recebe uma entrada, nesse contexto, uma imagem é utilizada como o alvo, essa etapa é caracterizada como a entrada da rede.

Na primeira camada, os neurônios respondem as diferentes características simples, como bordas e texturas da imagem. E nas camadas mais altas, os neurônios respondem a estruturas mais complexas da imagem e isso pode se estender a mais números de camadas – convoluções. Em seguida, após a extrações de tais características da imagem, na etapa da última camada, a rede treinada é capaz de

observar os objetos da imagem. E por fim, na etapa de saída da rede, baseado em seu treinamento, prevê qual é a classe da imagem alvo.

2.3.1.1 Camada Convolucional

De acordo com Buduma e Locasio (2017), o funcionamento de uma camada convolucional envolve um filtro ou *kernel* que é posicionado no topo da imagem e movido a cada passo (conforme a dimensão do *stride* do *kernel*). Durante esse deslocamento, o filtro verifica se há alguma característica entre ele e a imagem, se houver, a saída ficará realçada. Esse processo se repete por toda a dimensão da imagem, resultando em um mapa de características identificadas.

As camadas convolucionais são compostas por *"n"* filtros, com pesos inicializados aleatoriamente e depois são ajustados através do método de *backpropagation* (Almeida, 2018).

No entanto, o uso da convolução é aplicado as redes neurais convolucionais, como uma operação fundamental desse processo para extrair informações e características da imagem. Desta forma, considera-se dois vetores x e w, a convolução entre eles é definida por

$$y = x * w, \tag{2.8}$$

sendo x a entrada da camada convolucional, w é denominado como *kernel* ou filtro de características, conforme a Figura 2.3.

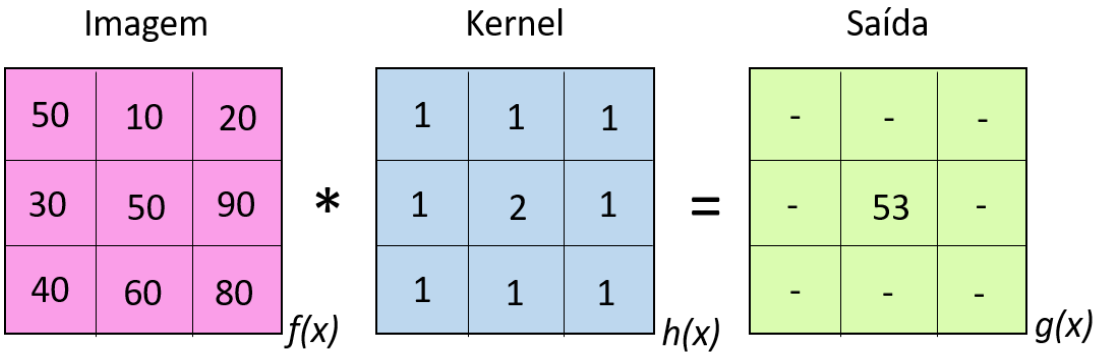

Figura 2.3: Exemplo de processo de convolução. (Adaptado de Khan et al., 2018).

A Figura 2.3 é a representação do processo de convolução que ocorre na etapa das camadas, observa-se que a saída é igual ao *pixel* 53, pois a convolução é a média

das multiplicações ponto a ponto com soma de um filtro, conforme as equações que são dadas por

$$g(x) = f(x) * h(x) = \sum_{\forall k} h(x-k) * f(x), \tag{2.9}$$

Portanto, quando um computador processa uma imagem, ele está “vendo” uma matriz de *pixels* com valores variando de 0 a 255, sua entrada é tridimensional, sendo a altura e largura (de acordo com a dimensão da imagem) e profundidade, determinada pela quantidade de canais de cores - (RGB) ou duas imagens em tons de cinza (Wu, 2017), conforme apresentado na Figura 2.4.

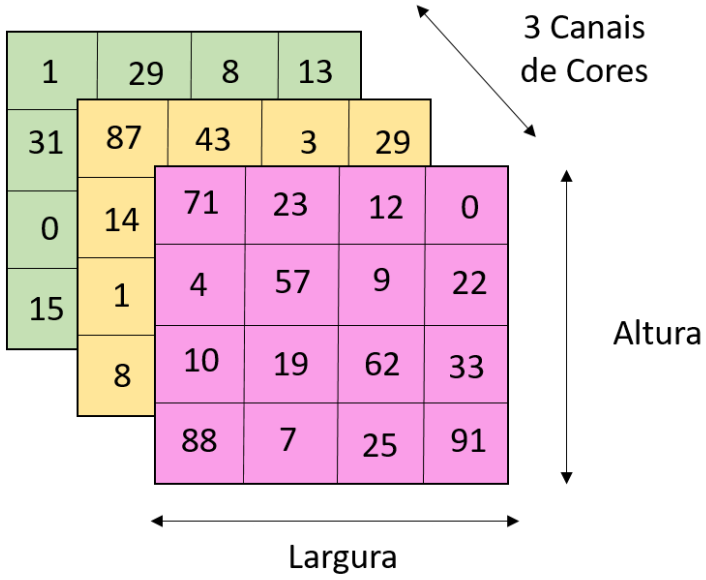

Figura 2.4: Camadas convolucionais. (Adaptado de Khan et al., 2018).

Por conseguinte, durante essa etapa, são estabelecidos hiperparâmetros, sendo eles a quantidade de filtros na camada, tamanho dos filtros, número de pixels em descolamento *(stride)* e *padding* (técnica usada para adicionar zeros nas bordas das imagens após a convolução).

2.3.1.2 Camada de *Pooling*

A camada de *pooling*, também conhecida como camada de agrupamento, é uma forma de *down-sampling*, essa técnica é utilizada para reduzir os tamanhos das dimensões das matrizes resultantes das camadas de convoluções, contribuindo também para o efeito de *overfitting* na rede.

De acordo com Melo (2018), a camada de *pooling* dispõe de dois parâmetros que ajudam nesse processo, uma máscara – um filtro, de dimensão quadrada (2x2, 4x4, 11x11) e o deslocamento desse filtro *(stride)*.

Há dois métodos frequentemente usados para operações de agrupamento: o max-pooling e o mean-pooling. No max-pooling, o sistema identifica o valor mais alto dos pixels em uma área delimitada pelo filtro. No mean-pooling, a média dos valores dos pixels dentro da mesma área delimitada pelo filtro é calculada. Essas operações são usadas para resumir informações em uma imagem ou em uma camada convolucional, destacando os valores máximos ou médios em regiões específicas (Massucatto, 2018). Para maior entendimento, a Figura 2.5 é um exemplo dessas duas funções.

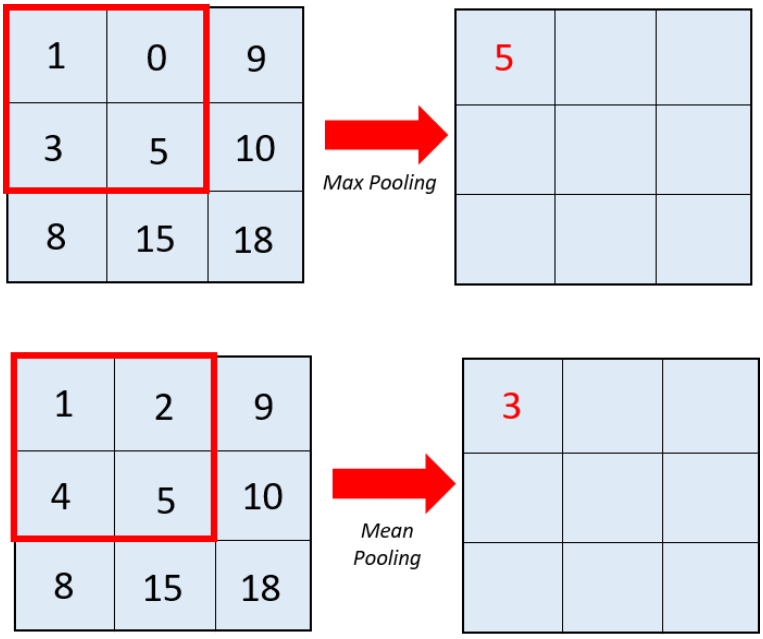

Figura 2.5: Exemplo das funções de pooling. (Adaptado de Khan et al., 2018).

2.3.1.3 Camada Densa

As camadas densas, também denominada de camadas totalmente conectadas, são uma representação de uma rede neural artificial clássica incorporada em um modelo convolucional (Melo, 2018). De acordo com Khan et al. (2018), sua função é classificar e atuar como saída da rede.

Porém, para o funcionamento da camada densa, é necessário converter as matrizes de características resultantes da camada de *pooling* em um vetor coluna, sendo utilizada como entrada para a RNA clássica (Massucatto, 2018). Este processo é conhecido como *flattening,* conforme apresentado na Figura 2.6:

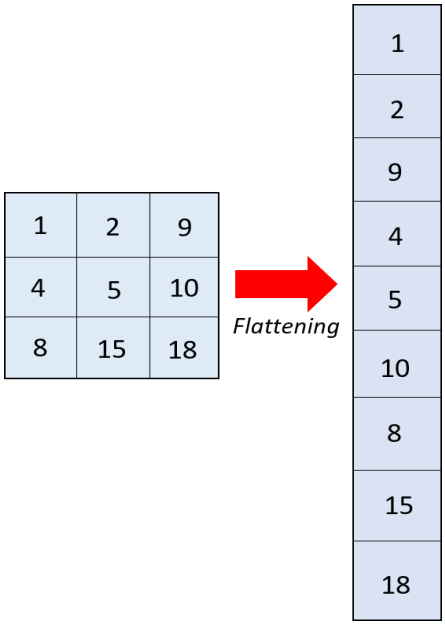

Figura 2.6: Exemplo Flattening. (Adaptado de Massucatto, 2018).

Após a realização de *flattening*, o objetivo da RNA clássica é combinar as características em mais atributos, de forma a aprimorar a capacidade de previsão das classes das imagens. Sendo assim, o erro é calculado semelhante a uma rede neural recorrente (RNR).

Conforme apresentado na Seção 2.1, as redes neurais precisam de uma função de ativação. Atualmente, a função de ativação que é mais utilizada no método das redes convolucionais é a função de unidade linear retificada (ReLU), que mapeia a saída para zero, caso o valor da entrada seja negativo ou se for positivo, envia-o diretamente para a saída, sem nenhuma modificação. Essa função evita que ocorra o *overfitting* em camadas extensas de convoluções, sua expressão é dada por

$$f_{ReLU}(x) = (0, x). \tag{2.11}$$

2.4 Algoritmos Baseado em CNN

De acordo com a literatura, a CNN tem se mostrado bastante eficaz em relação à detecção de imagens. Portanto, devido a tal eficácia e popularidade, surgiram

algumas arquiteturas de CNN, como a AlexNet, LeNet, Inception, Unet, Xception, VGG, DenseNet.

Por ser utilizada na proposta dessa dissertação, a seguir será exposto a rede You Only Look Once (YOLO) com mais detalhes.

2.4.1 You Only Look Once (YOLO)

A família YOLO se destacou dos outros algoritmos citados anteriormente, pois, ela requer apenas uma propagação de entrada da imagem pela rede para predizer seu resultado, oferecendo maior desempenho em relação a aplicações em tempo real.

A criação da YOLOv1 por Redmon et al., (2016), foi um divisor na área da visão computacional, pois a partir dela, evoluíram outras versões através de inúmeras iterações baseadas nas versões anteriores, resolvendo as limitações e melhorando o desempenho do modelo. A Figura 2.7 é uma representação de uma linha do tempo do desenvolvimento da família YOLO.

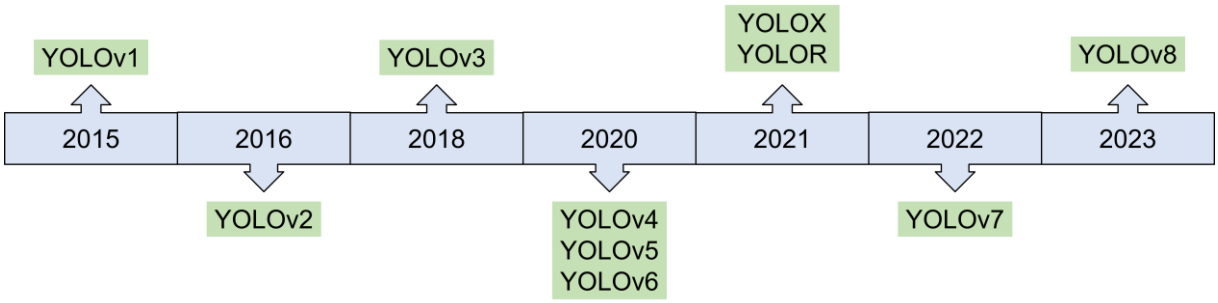

Figura 2.7: Desenvolvimento da família YOLO ao longo do tempo.

Este trabalho utiliza do algoritmo YOLOv8, lançada em janeiro de 2023 pela Ultralytics, sendo a mesma empresa que desenvolveu a YOLOv5, contudo, há algumas semelhanças na estrutura das duas versões.

Nesse contexto, a família YOLO revoluciona a abordagem da detecção de objetos ao redefini-la como um problema de regressão, partindo exclusivamente dos pixels de uma imagem. Esse processo estima as predições dos objetos por probabilidade de classes, coordenadas e dimensões que delimitam os objetos a serem identificados (Redmon et al., 2017).

Portanto, a utilização da YOLOv8 como proposta da dissertação para detecção e classificação de manchas em superfície oceânica torna-se interessante. Dessa maneira, prevendo a possibilidade de mitigar a problemática exposta para autoridades e órgãos competentes, para que eles possam agir com rapidez mediante à situação.

De acordo com Huang, Pedoeem e Chen (2018), a rede funciona em dividir as imagens em grade de células $S \times S$, cada uma dessas células gera predições de acordo com as caixas delimitadoras, também denominada de *bounding boxes*, baseadas pelos seguintes componentes: (b_x, b_y) são coordenadas que representam o centro das *bounding boxes*, (b_h, b_w) representam a altura e a largura das *bounding boxes* respectivamente, (P_c) significa a probabilidade de existir algum objetos dentro da caixa delimitadora e (C) representa a classe determinada para denominar o objeto. A Figura 2.8 é a representação funcional de um objeto detectado baseado nas caixas delimitadoras.

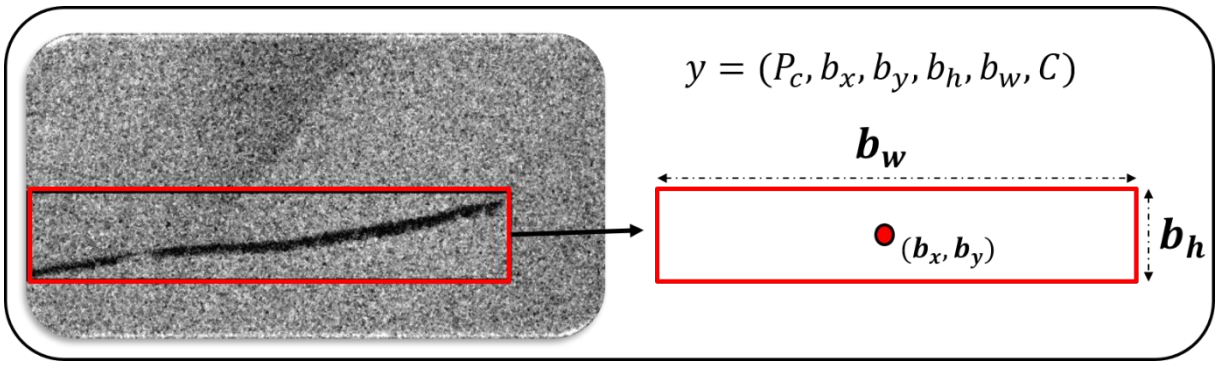

Figura 2.8: Esquema funcional de detecção de um objeto baseado nas *bounding boxes.*

Portanto, cada célula gera predições em conjunto com o grau de confianças dos objetos identificados, ou seja, caso algum objeto esteja presente na célula, o grau de confiança será 1, caso contrário, será 0. A Figura 2.8 é um exemplo de como funciona o grau de confiança em relação aos objetos presentes nas células.

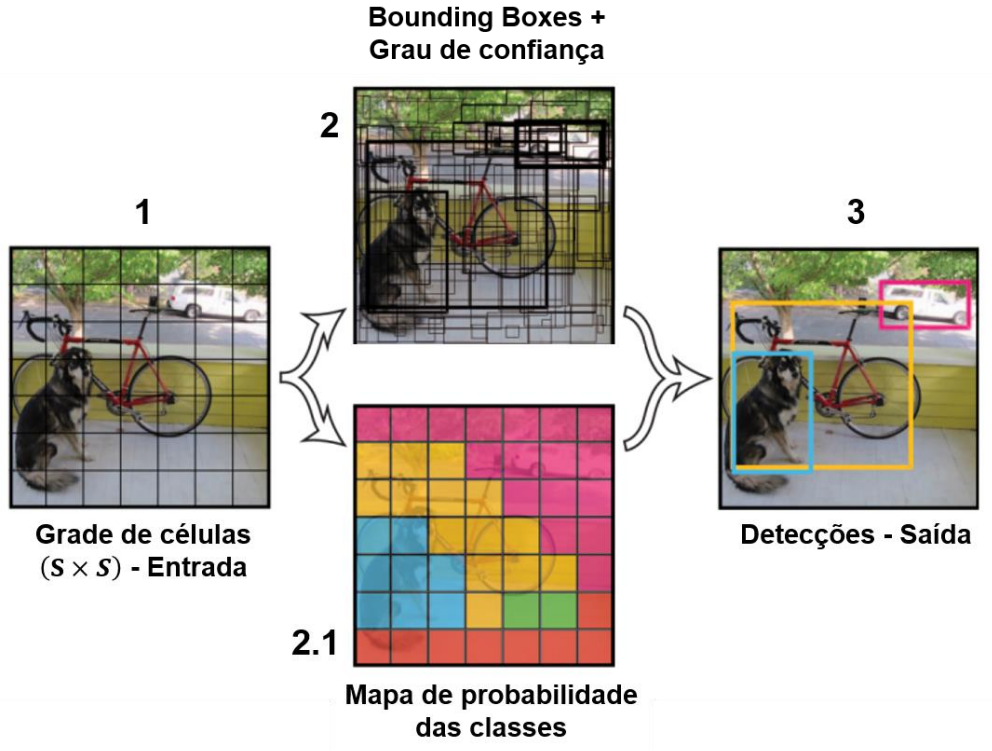

Figura 2.8: Exemplo de funcionamento da YOLO em relação a detecção de objetos em conjunto com o grau de confiança. (Adaptado de Redmon et al., 2017).

Observe-se que, na etapa 2 da Figura 2.8, algumas caixas apresentam uma espessura mais grossas em suas delimitações, apontando que, de acordo a grade de células, os objetos com maior grau de confiança foram destacados na etapa 3.

2.5 Avaliação de Desempenho

Existem diversas medidas de avaliação do desempenho de um classificador. Todas as métricas de avaliação têm o objetivo de medir quão distante o modelo está da classificação ideal, dependem do tipo de dado em questão e da natureza da classificação realizada. Nesta seção apresenta algumas métricas de avaliação de desempenho utilizado em classificações binárias (duas classes de objetos). Elas baseadas de acordo com o conceito de matriz de confusão, de acordo com a tabela:

Tabela 2.1 - Matriz de Confusão

	Valor verdadeiro (Mundo real)	
Resultado da classificação	Classe A (positivo)	Classe B (negativo)
Classe A (positivo)	**VP** (Verdadeiros Positivos)	**FP** (Falsos Positivos)
Classe B (negativo)	**FN** (Falsos Negativos)	**VN** (Verdadeiros Negativos)

Fonte: Adaptado de Motta, 2015.

De acordo com Motta (2015), a Tabela 2.1 confronta os valores gerados por um classificador e dessa forma, podem acontecer quatro situações possíveis, considerando as duas classes:

- VP: um objeto A ser classificado pertencente da classe A;
- FN: um objeto A ser classificado como pertencente da classe B;
- FP: um objeto B ser classificado pertencente a classe A;
- VN: um objeto B ser classificado pertencente a classe B.

Sendo assim, as métricas utilizadas para problemas de detecção e classificação de objetos em imagens são a precisão e o *recall.* Nos parágrafos seguintes apresenta-se a relação matemática e a interpretação das referidas métricas.

A precisão, conhecida também como a taxa de verdadeiros positivos, mede a proporção de classificações positivas dentre todos os objetos realmente positivos, dessa forma sua definição é dada por

$$PR = \frac{VP}{VP + FP}. \tag{2.12}$$

O *recall,* também conhecido como a taxa de verdadeiros negativos, mede a proporção de classificações negativas dentre todos os objetos realmente negativos é definido por

$$Recall = \frac{VF}{VN + FP}. \tag{2.13}$$

Já a métrica F1 é baseado na precisão e no *recall*, é conceituada como a média harmônica entre a precisão e o *recall,* é dada por

$$F1 = \frac{2 \times PR \times Recall}{(\mathrm{PR} + Recall)}. \quad (2.14)$$

De acordo com a Equação (2.14), se a precisão ou recall for igual a zero, o F1 será baixo, fazendo com o que classificador não seja tão bom em relações as proporções.

Há também uma métrica de avaliação utilizada na rede YOLOv8, denominada por *mean Average Precision* (mAP). A mAP é representada pela área debaixo da curva *Precision-Recall*, nomeada por *Average Precision* (AP). A métrica referida, é calculada através da média AP sobre todas as classes determinadas na detecção dos objetos nas imagens.

Para avaliar a modelo treinado, essa métrica realiza duas etapas, uma etapa de localização do objeto e outra de classificação desse objeto, ou seja, ela localiza a posição do objeto na imagem e após isso, ela classifica qual é esse objeto, dessa forma, gerando uma margem de confiança na classificação.

Nesse contexto, a YOLOv8 utiliza o método das *bounding box*. Nessa situação, utiliza-se da métrica IoU (*Intersection over Union*), medindo a sobreposição entre as caixas delimitadoras preditas e a caixa delimitadora real.

O limiar do IoU é definido como 0,5, representado por **mAP50**, caso o valor esteja acima disso, é considerado um acerto, caso contrário, um erro. Da mesma forma, para a métrica **mAP50-95**, que calcula a média considerando vários limiares simultaneamente, sendo eles entre 0,5 e 0,95. Essa abordagem tem o objetivo de gerar maior predição do modelo, ou seja, quanto maior a área de sobreposição das caixas, maior a predição, possibilitando avaliar o que é um acerto ou o que é um erro. Essa métrica é dada por

$$mAP50 - 90 = \frac{1}{10} \times \sum_{IoU=0,5,i=0,05}^{0,95} \left(\sum_{c=1}^{n} PR(Recall)_{IoU} \right), \quad (2.15)$$

sendo, c representando as classes, n representando o total de classes definidas para o estudo, nesse caso da dissertação, utilizou-se apenas uma classe, definida como "Com_manchas"; IoU é o intervalo dos limiares entre 0,5 e 0,95 e $PR(Recall)_{IoU}$ representa o valor da curva AP da classe no limiar específico de IoU.

Capítulo 3

PROBLEMA E SOLUÇÃO

Neste capítulo estão detalhados os atributos e características da problemática exposta, dessa forma, mostra-se o banco de dados baseado na Missão SENTINEL-1, como agente facilitador para a captação dos dados. Evidencia a formulação do problema, assim como também, a solução do problema, baseado no agente detector.

3.1 Descrição do Problema

Os derramamentos de óleos em superfície oceânica representam uma das principais ameaças ambientais, sociais e econômicos para a área atingida. Há distintas técnicas de monitoramento nesses ambientes e uma delas, em específico, é o satélite de radar de abertura sintética (SAR). Sua aplicação nesse cenário é uma ótima escolha, pois, fornece imagens de alta resolução durante o dia e a noite sobre todas as condições climáticas.

Mediante o problema exposto, surgiu a missão SENTINEL-1, desenvolvida pela Agência Espacial Europeia (ESA) no período de 28 de setembro de 2015 a 31 de outubro de 2017. Seu objetivo principal foi a realização do monitoramento em relação ao nível de poluição das áreas terrestres e oceânicas, abrangendo o solo, a vegetação e a costa marinha.

Os satélites destinados para a missão foram equipados com sistema SAR operando em banca C e onde a sua cobertura terrestre é de aproximadamente 250

km com espaçamento de pixel igual a 10m x10m. A partir dessa missão, foram fornecidas imagens capazes de identificar poluições, agentes externos ao meio referido, como manchas de óleos no mar.

Conforme Krestenitis et al. (2019), a base de dados é constituída por 1.112 imagens em tons de cinza e suas respectivas máscaras em RGB (*Red, Green e Blue*). As imagens contêm 5 classes: superfície terrestre; superfície marinha; navios; manchas de óleos e sósias de manchas de óleos.

A Figura 3.1 é a representação de uma imagem da base de dados de treino com sua respectiva máscara.

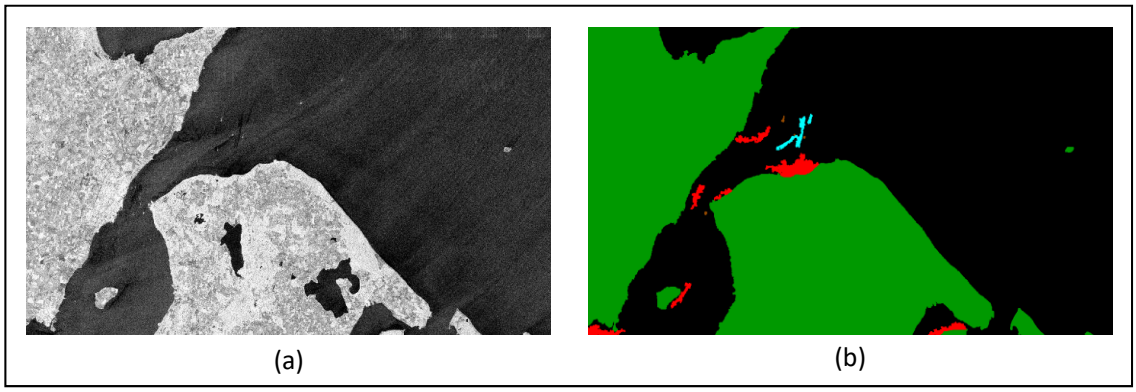

Figura 3.1: Imagem da base de dados: (a) imagem tons de cinza, (b) imagem da máscara em RGB.

De acordo com a Figura 3.1, as máscaras têm a finalidade de identificar através de cinco cores distintas as classes das imagens disponibilizada pelo satélite da missão SENTINEL-1. Portanto, a superfície terrestre é caracterizada pela cor verde; a superfície marinha é representada pela cor preta; os navios são caracterizados pela cor marrom; as manchas de óleos pela cor ciano e por fim, a sósias são caracterizadas pela cor vermelho.

3.1 Formulação do Problema

Os métodos, técnicas e modelos baseados no algoritmo YOLOv8 são apresentados nesta seção. A Figura 3.2 apresenta uma visão geral da metodologia, que consiste no meio a ser monitorado, na aquisição dos dados e no método utilizado para detectar o objeto de interesse.

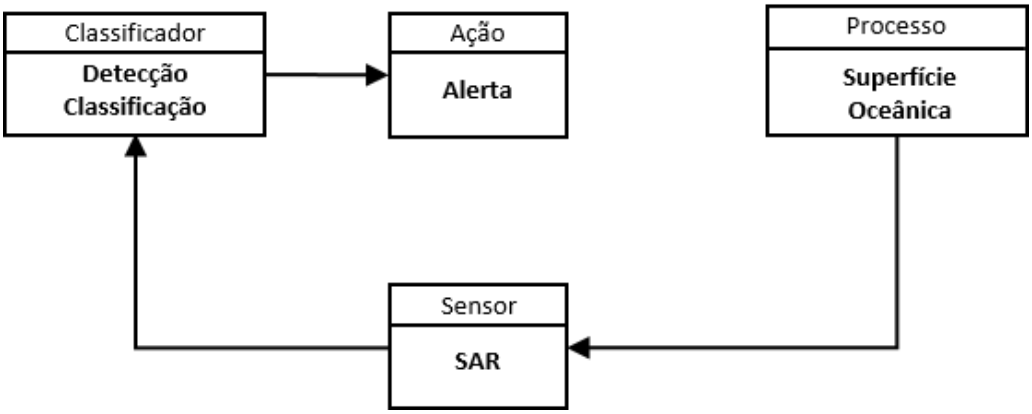

Figura 3.2: Esquema do sistema proposto para detecção de manchas de óleo na superfície do oceano.

Conforme a Figura 3.2, é necessário determinar o local onde será realizado o monitoramento, ou seja, em ambientes onde existem rotas de petroleiros ou em indústrias que utilizam óleos e petróleos como matéria-prima.

Dessa forma, o desenvolvimento da técnica proposta envolve primeiro a análise dos dados para avaliar a distinção entre as duas classes em estudo, seguida do pré-processamento do conjunto de dados. Após esta etapa, o conjunto de dados está preparado para treinar o modelo de classificação. Por fim, são realizados testes para avaliar o desempenho do algoritmo treinado através de métricas de avaliação. Essas etapas são apresentadas no Capítulo 4.

3.2 Classificador-YOLO

De acordo com a Figura 3.2, o classificador, também é considerado como o agente solucionador. Ele é o responsável por detectar e classificar as manchas de óleos em superfície aquática, ou seja, avalia o impacto da perturbação nesse meio. Sendo assim, através da avaliação do classificador, é enviado o sinal de alerta as autoridades competentes.

Portanto, a técnica abordada está relacionada com as etapas da Figura 3.2, onde se tem o processo – sendo a superfície oceânica como objeto a ser monitorado; o sensor – sendo a forma de monitoramento para o meio; o classificador – como o algoritmo proposto de classificar e detectar perturbações e o alerta – como sinal para

a tomada de decisão, sendo uma unidade de ação rápida para os órgãos competentes, de acordo com a avaliação do agente.

Os estudos desta dissertação estão em torno do agente solucionador, visto que ele é apresentado como o solucionador da problemática. Sendo assim, são apresentados ao longo da dissertação o algoritmo YOLOv8 que é proposto como o agente, pois ele tem a capacidade de detectar e classificar tais perturbações nocivas ao meio aquático, sendo capaz de realizar uma leitura do processo de forma rápida e eficaz.

Capítulo 4

SISTEMA DE DETECÇÃO E CLASSIFICAÇÃO PROPOSTO

Neste capítulo estão detalhados a metodologia proposta para o sistema de detecção e classificação de manchas de óleos no mar, evidenciando o passo a passo de cada etapa, como a análise da base de dados, as ferramentas e transformações adotadas no pré-processamento das imagens. Como também, apresenta a arquitetura do classificador-YOLO e os parâmetros utilizados na fase de treinamento

4.1 Metodologia Proposta

A metodologia proposta tem como objetivo analisar a família do algoritmo YOVOv8. Assim, os métodos de transformação de imagens em tons de cinza, corte e redimensionamento são aplicados ao conjunto de dados de treinamento e validação como pré-processamento. No conjunto de dados de teste, apenas o corte e o redimensionamento foram aplicados. Portanto, para uma análise mais detalhada do pré-processamento do conjunto de dados e da metodologia do algoritmo YOLOv8, a Figura 4.1 é uma representação das etapas desse processo.

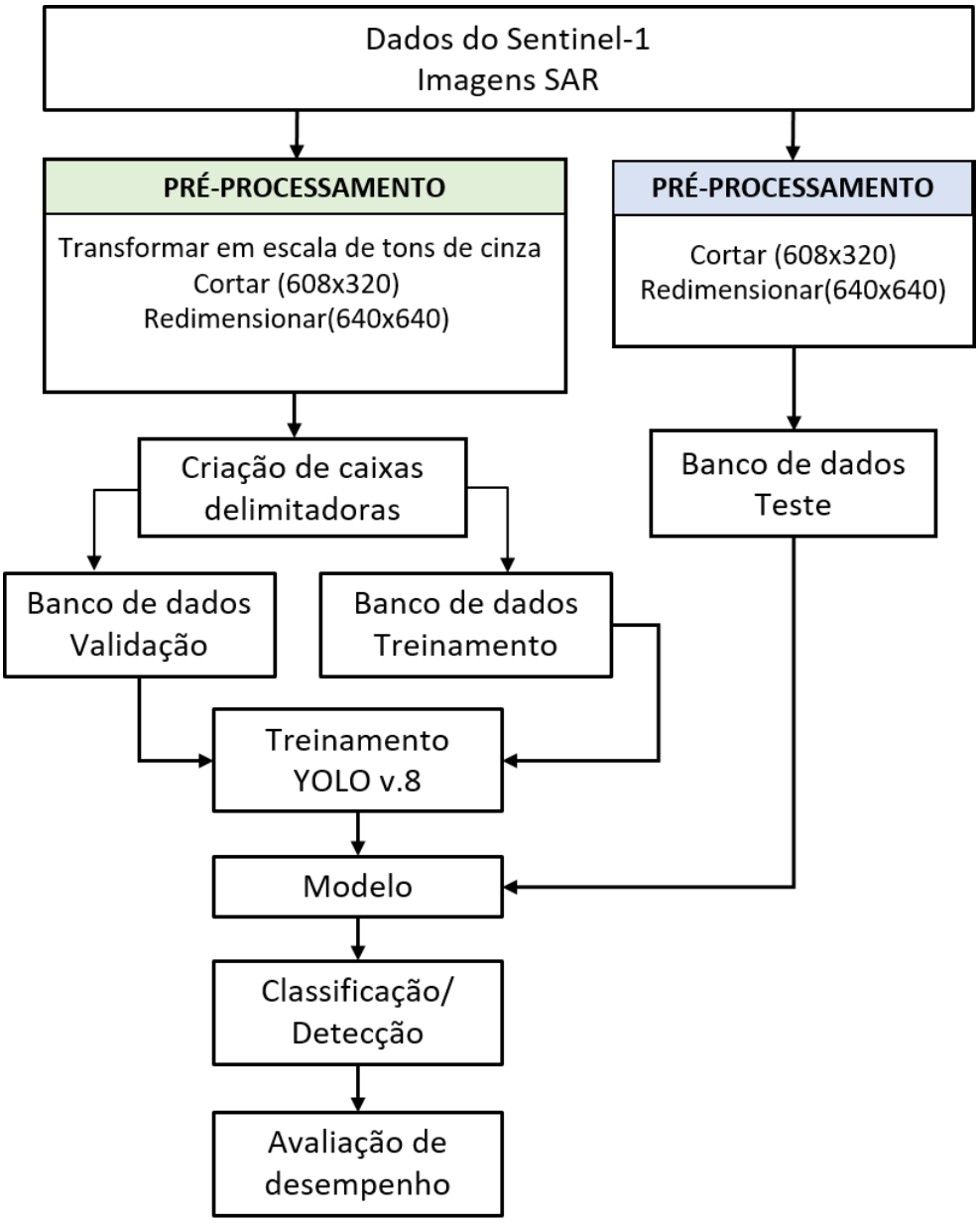

Figura 4.1: Fluxograma do método proposto.

4.1.1 Análise da Base de Dados

Antes de aplicar qualquer método de pré-processamento nas imagens, a análise do conjunto de dados é necessária porque fornece informações sobre o comportamento das classes. Nesta análise, foi realizada uma análise da distribuição de pixels das imagens com e sem manchas de óleo para determinar as semelhanças

entre elas e se é possível separá-las. Esta análise é importante para fazer uma escolha segura entre o modelo e o algoritmo de classificação.

4.1.2 Pré-Processamento

Como pode ser visto na Figura 4.1, para a YOLOv8, foi-se necessário transformar o banco de dados em escalas de tons de cinza, neste caso, toda mancha de óleo antes representada pela cor ciano, ficou em cinza, em pixel de número 178.

Para evitar o *overfitting* e aumentar o desempenho do algoritmo, a imagem antes com dimensão de 1260x650 *pixels* foi cortada em quatro partes, transformando em imagens de dimensões de 625x325 *pixels* e no segundo passo, foi redimensionada em 640x640 pixels.

4.1.3 Anotação de objetos

A etapa seguinte foi de rotular em objetos os derramamentos de óleos, o que é de extrema importância, conforme visto no banco de dados bruto, existem as manchas sósias. E essa etapa de rotulação serve como forma do algoritmo identificar quais são as imagens com manchas de óleos verdadeiras.

Para a etapa de reconhecimento de objetos da YOLOv8, é necessário a realização de um mapeamento dos objetos de interesse, neste caso, denominado como “Com_manchas”. Dessa forma, ela trabalha de forma eficaz na detecção e classificação desses objetos, localizando e identificando as características de cada objeto. Consequentemente, a YOLOv8 utiliza de um formato de arquivo em **.txt* que tenha o mesmo nome do arquivo de imagem, ou seja, cada imagem precisa ter um arquivo em *txt correspondente.

Na mesma vertente, para etapa da metodologia que inclui as criações das caixas demilitadoras, primeiramente, foi-se necessário separar os arquivos de imagens e anotações em duas pastas distintas, sendo elas em “*imagens_Com_manchas”* e *“labels_Com_manchas”,* respectivamente. A pasta das imagens contém a classe “*Com_manchas”* e suas *bounding box* de acordo com as características dessa classe. Já o arquivo que contém as *labels* segue para cada linha uma anotação do objeto de interesse e cada anotação engloba uma determinada sequência, sendo elas: a classe, a posição central no eixo (x), a posição central no eixo (y), a largura e a altura.

Dessa forma, viabilizou-se a realização a extração das regiões de interesse (ROI), a partir desta etapa, criou-se as caixas delimitadoras baseada no contorno de cada mancha de óleo. Dessa maneira, cada imagem que contém uma mancha conforme o pixel de 178, tem uma caixa delimitadora em sua volta, a Figura 4.2 é uma exemplificação do resultado de uma imagem com uma caixa delimitadora.

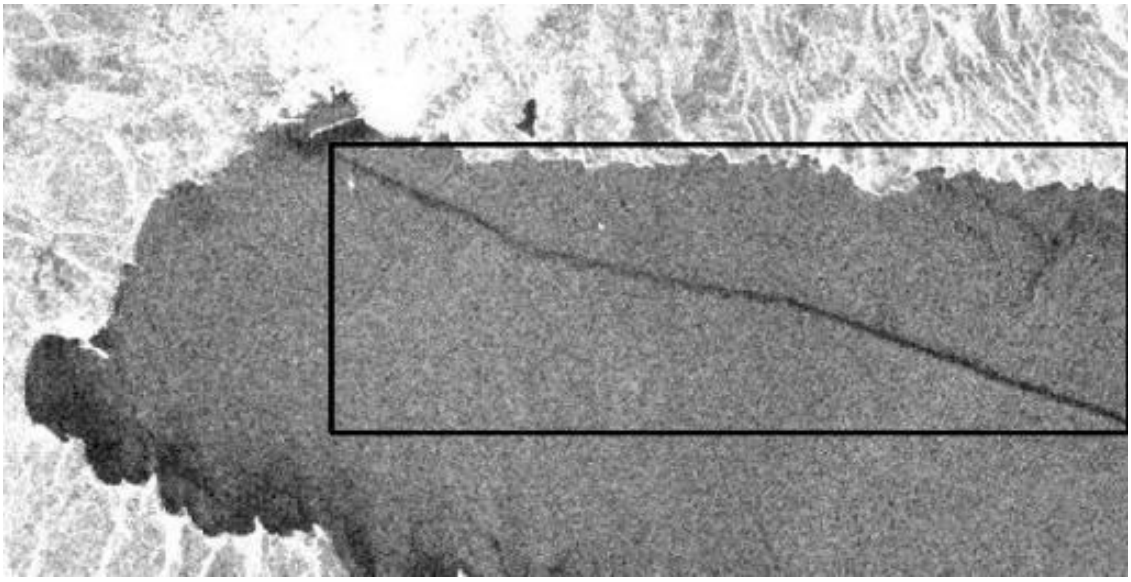

Figura 4.2: Imagem da base de dados de treinamento com uma caixa delimitadora.

Um conjunto de dados de treinamento, composto por 3.376 arquivos foi montado dividido em duas partes: 1.688 imagens de entrada e 1.688 arquivos *(*.txt)* contendo as dimensões das caixas delimitadoras, e o conjunto de dados de validação com 220 arquivos dividido em duas partes: 110 imagens de entrada e 110 arquivos *(*.txt)*. Para o conjunto de dados de teste foi utilizado apenas o recorte de imagens, composto por 440 imagens correspondentes à dimensão de entrada do algoritmo YOLOv8. Da mesma forma, após o pré-processamento do conjunto de dados, conforme mostrado na Figura 4.1, as etapas de treinamento produzem o modelo previsto do classificador, que é então validado e testado para avaliar e comparar o desempenho dos modelos utilizados.

4.1.4 Arquitetura da YOLOv8

A YOLOv8 abrange cinco modelos: YOLOv8n (nano), YOLOv8s (*small*), YOLOv8m (*medium*), YOLOv8l (*large*) e a YOLOv8x (*extra large*). Sua arquitetura é composta por três camadas principais: *backbone*, *neck* e *head*.

De acordo com Seidel (2023), a primeira camada é responsável por extrair características das imagens de entrada durante a fase de treinamento. A segunda camada, refere-se a uma camada intermediária, de acordo com Bochkovskiy, Wang e Liao (2020), sua função é reunir mapas de características de diversos estágios. Esse processo otimiza o desempenho e a propagação de informações semânticas de maneira mais eficaz, viabilizando através de sua ativação, diferentes tipos de percepções. Esse tipo de estrutura da camada intermediária permite uma compreensão das características extraídas das imagens de forma mais abrangente, o que contribui para uma melhor detecção de objetos especificados nas imagens. A terceira camada, é responsável pela saída do modelo, onde está associada à classificação dos rótulos – utiliza a regressão da predição das caixas delimitadoras para definir as classes dos objetos (Wang, Bochkovskiy e Liao (2022).

A visão completa da arquitetura da YOLOv8 pode ser conferida no anexo A. Por conseguinte, serão apresentadas os detalhes das camadas da arquitetura, conforme a divisão vista no anexo A.

O ***backbone* da YOLOv8** é similar ao da YOLOv5, com algumas alterações, utiliza a combinação da CPS-DarkNet53, com 53 camadas convolucionais sucessivas, aumentando a capacidade de aprendizado através da combinação das camadas de transição parcial e das características extraídas das camadas iniciais. Esta combinação reduz o custo computacional e a redundância de gradientes nas CNN (Wang et al., 2020).

Na camada ***Neck* da YOLOv8** também se assemelha com o da YOLOv5. De acordo com Siedel (2023), são utilizadas camadas SPPF (*Spatial Pyramid Pooling – Fast*), sua função é remover as restrições de imagens com tamanhos diferentes da entrada solicitada, ou seja, utiliza como saída as mesmas imagens de entrada, sem precisar deformar ou corta-las. A *neck* tem o intuito de amenizar os custos computacionais, a utilização da SPPF reduz a quantidade de camadas de SPP, otimizando o custo computacional e aumentando a velocidade de treinamento.

Na camada de ***Head* da YOLOv8**, sua função é realizar as detecções em três formas distintas: as *bounding boxes*, a presença de objetos nas imagens e a probabilidades de classe. São nessas camadas que acontece o módulo C2f, ele serve para otimizar a segmentação semântica, também com o mesmo objetivo das camadas anteriores, que é de reduzir o custo computacional e melhorar a etapa de treinamento.

Nos módulos C2f são utilizados o *bottlenecks,* onde sua função é estabelecer uma separação natural entre as camadas de entrada e saída e as camadas de transformação de cada bloco de convolução, portanto, como visto na arquitetura (Anexo A), a cada transformação é utilizado a Conv-BN-SiLU, composta por camadas convolucionais, *Batch Normalization* e a função SiLU (*Sigmoid Linear Unit*) (Seidel, 2023).

4.1.4.1 Modelos Pré-Treinados e Transfer Learning

De acordo com a Figura 4.1, na etapa de treinamento da YOLOv8, foi-se necessário estabelecer temas como modelos pré-treinados e *transfer learning*.

A YOLOv8 fornece modelos pré-treinados, na Tabela 4.1 mostra que quando mais robusto o modelo, maior capacidade computacional ele precisará para ser treinado. Dessa forma, os modelos disponíveis apresentam variações em termos de capacidade de execução e desempenho.

Tabela 4.1 - Resultados dos modelos pré-treinados com base de dados COCO.

Modelo	Tamanho (pixels)	Parâmetros (M)	FLOPs (B)
YOLOv8n	640x640	3.2	8.7
YOLOv8s		11.2	28.6
YOLOv8m		25.9	78.9
YOLOv8l		43.7	165.2
YOLOv8x		68.2	257.8

Fonte: Adaptado de Ultralytics, 2023.

Observa-se que nas colunas parâmetros e FLOPs evidenciam a distinção entre o tempo de execução e desempenho. A coluna parâmetros representa o total de pacotes de características empregados nas camadas da rede neural e na coluna FLOPs representa a quantidade de operações realizadas durante o treinamento.

A diferença entre os modelos pré-treinados está associada com a profundidade e largura das camadas convolucionais. Nesta situação, no Anexo A é apresentado as camadas da arquitetura, onde as referidas profundidades e larguras são salientadas.

A profundidade está inteiramente ligada com a extração de conjunto de características rica em detalhes, minimizando problemas relacionados com aprendizados complexos. Em relação a largura das camadas convolucionais, ela é utilizada no que diz respeito de características aprendidas, ou seja, ela é um parâmetro de definição de aprendizado.

Com os modelos pré-treinados da YOLOv8, utiliza-se um processo chamado de *transfer learning* que visa aumentar o desempenho de novos modelos treinados, ou seja, em vez de treinar um modelo do zero para uma determinada tarefa, ele aproveita o modelo pré-treinado em uma tarefa que seja relacionada. De acordo com Zhuang et al. (2020), esse tipo de transferência de aprendizado não pode ser tão promissor para algumas tarefas de detecção, gerando resultados negativos para o modelo treinado.

Na mesma vertente, essa dissertação utilizado o congelamento das camadas neurais. Os pesos treinados da base de dados originais não são retreinados, permitindo que apenas as últimas camadas do classificador sejam otimizadas de acordo com a função de perda.

A escolha para esse tipo de treinamento é crucial para minimizar o custo computacional em relação ao tempo de treinamento, consumindo assim, menos recursos para as imagens do banco de dados da missão SENTINEL-1.

Os hiperparâmetros principais utilizados para o treinamento do modelo estão descritos na Tabela 4.2.

Tabela 4.2 – Hiperparâmetros utilizados no treinamento da YOLOv8.

Hiperparâmetro	**Valor**
Learning rate inicial - *lr0*	0.01
Learning rate final - *lrf*	0.01
Momentum	0.937
Batch Size	16
Epochs	50
Warmup_momentum	0.8
Weight_decay	0.0005
Warmup_epochs	3.0
box	0.5
Cls	0.5
Dlf	0.5
Iou	0.7

Os parâmetros da Tabela 4.2 são importantes na etapa de treinamento, pois estes têm a capacidade de modelar o treinamento, de acordo com os ajustes da tarefa

para qual será executada e também de otimizar o processo de aprendizado do modelo.

O *lr0* e *lrf* influenciam na taxa de convergência do modelo, determinando a velocidade em que ele se ajusta aos dados, o momentum tem a função de acelerar o gradiente descendente, controla a taxa de aprendizagem do otimizador. O *batch size* e número de *epochs* tem a função de minimizar a demanda por memória e a eficiência computacional. Os parâmetros *warmup_momentum* e *warmup_epochs* contribuem para o começo do treinamento e o *weight_decay* contruibui para controlar a magnitude dos pesos, minimizando o sobreajustes. Os parâmetros *box*, *cls*, *dlf* e *iou* tem relação direta com a função de perda, impactando na detecção e classificação dos objetos nas imagens.

Dentro do próprio modelo, é utilizado *data augmentation,* onde são técnicas que contribuem para o aumento do banco de dados, que auxiliam a melhorar a robustez do modelo, o tornando menos sensível há alguma variação nos dados de entrada. Na Tabela 4.3 são apresentadas as técnicas aplicadas e os seus valores referentes.

Tabela 4.3 – Técnicas utilizadas no data augmentation na etapa de treinamento.

Técnica	Valor
hsv_h	0.015
hsv_s	0.7
hsv_v	0.4
Translate	0.1
Scale	0.5
Fliplr	0.5
Mosaic	1.0

As técnicas *hsv_h*, *hsv_s* e *hsv_v* estão relacionadas com ajustes em matiz, saturação e brilho das imagens. O *translate* em 0.1 é o ajuste da imagem em relação a translação, significa que as imagens podem ser transladadas em 10% da altura ou largura. O *scale* em 0.5 significa que a imagem pode ser reduzida ou aumentada em até 50% da imagem original. O *fliplr* em 0.5 é a técnica de espelhar horizontalmente a imagem, neste caso, com 50% de espelhamento horizontal. Por fim, o *mosaic* combina quatro imagens distintas em apenas uma, formando assim um mosaico.

4.1.4.2 Função de Perda

De acordo com a etapa de classificação e detecção da Figura 4.1, é necessário entender como funciona a função de perda no treinamento da YOLOv8. Ela se destaca pela combinação de três funções, sendo elas, localização (*Location Loss*), identificação de objetos (*Objectness Loss*) e a classificação de objetos (*Classification Loss*). A função da perda é dada por

$$Erro = \lambda_1 E_{loc} + \lambda_2 E_{obj} + \lambda_3 E_{cls}, \quad (3.1)$$

sendo E_{loc}, E_{obj}, E_{cls} denominados como *location loss, objectness loss* e *classification loss*, respectivamente.

Neste caso, para a *location loss*, a métrica utilizada é a IoU, que mede a sobreposição entre a *bounding box* predita e real. Dessa maneira, a IoU é utilizada para o cálculo da função da perda durante a etapa de treinamento que é realizada a regressão do real e o predito.

Para as funções de perda relacionadas com *objectness loss* e *classification loss,* é utilizado método *Binary Cross Entropy Loss* (*BCE Loss*), que mede a distância entre a probabilidade da predição e das manchas de óleos, no caso, o objeto alvo. Esse método *BCE loss* só é possível devido a IoU calculado na etapa de treinamento.

Portanto, seguindo as etapas da metodologia apresentadas na Figura 4.1, as métricas de avaliação são vistas na Seção 2.5.

Todas essas etapas são cruciais para os resultados do modelo treinado. O próximo Capítulo são apresentados os resultados e discussões da metodologia proposta baseada YOLOv8.

Capítulo 5

RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados dos experimentos do modelo YOLOv8 com a base de dados da missão SENTINEL-1. Conforme o objetivo desse trabalho, foram analisados três modelos da YOLOv8 (nano, small e medium) utilizando os mesmos parâmetros e técnicas. Dessa forma, mostrando o modelo mais aceitável para detecção e classificação de manchas em superfície oceânica.

5.1 Análise do Banco de Dados

Primeiramente foi analisada a distribuição das imagens com e sem manchas de óleo. A fim de analisar a distribuição das imagens e analisar se há semelhança na distribuição entre as classes, o que é de extrema importância para prever se elas são separáveis, é selecionado de forma aleatória duas imagens de cada classe e plotado seus histogramas, a Figura 5.1 são esses histogramas, na qual o eixo x representa o valor dos pixels, que vária de 0 a 255 e o eixo y representa a frequência que é o valor de pixel que aparece naquela imagem.

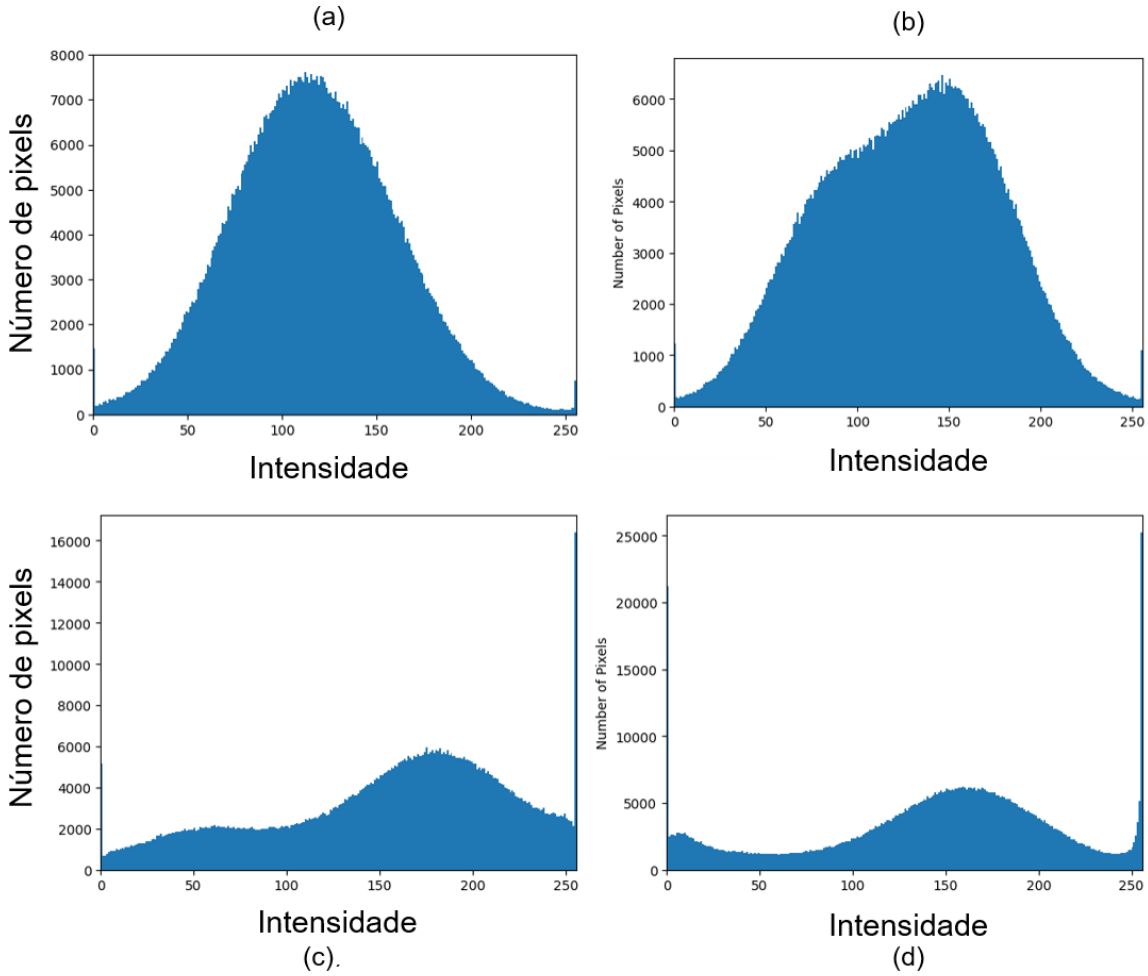

Figura 5.1: Histograma de pixels em relação às classes de imagem: (a) e (b) representam a classe com manchas de óleo; (c) e (d) representam a classe sem manchas de óleo.

Pode-se observar que as imagens (a) e (b) da Figura 5.1 correspondem àquelas com manchas de óleo. Elas possuem um pico de cerca de 100 a 150 pixels, o que representa uma intensidade de pixels em tons de preto. As imagens (c) e (d), por outro lado, possuem um pico de cerca de 150 a 200 pixels, representando imagens coloridas mais próximas do branco.

5.2 Resultados baseados no Modelo YOLOv8

Os modelos da YOLOv8 foram treinados utilizando os mesmos parâmetros, para que a análise comparativa entre eles fosse justa, já que todos têm o mesmo objetivo de atingir a tarefa de detecção e classificação de manchas de óleos em

superfície oceânica. Sendo assim, a rede foi treinada usando o modelo e os parâmetros citados no Capítulo 3, na Subseção 3.2.2.4. Para melhor compreensão da metodologia proposta é apresentada a arquitetura da YOLOv8 no Anexo A.

De acordo com a Tabela 5.1, foram realizados três testes na família YOLO v8: YOLO nano, YOLO *small* e YOLO *medium*, para medir a eficácia de cada algoritmo de acordo com as métricas de desempenho.

Tabela 5.1 - Comparação do tempo de treinamento do algoritmo da família Yolov8.

Modelo YOLOv8	Tempo de treinamento	GFLOPs
Nano	0.665 horas	8.1
Small	0.857 horas	28.4
Medium	0.974 horas	78.7

Com relação ao teste, na Tabela 5.2 é mostrado os resultados das métricas de avaliação da precisão, *recall* e mAP calculadas a partir das Equações 2.12, 2.13 e 2.15, respectivamente.

Tabela 5.2 - Resultados de desempenho na fase de teste.

Métricas / Modelo	Precisão (P)	*Recall* (R)	mAP50 (%)	mAP50-90 (%)
Nano	**0.927**	0.783	0.837	0.697
Small	0.921	0.776	0.849	0.711
Medium	0.893	**0.806**	**0.85**	**0.716**

A Tabela 5.2 fornece uma visão geral dos testes realizados. Observando a métrica mAP50-95, que quantifica a precisão da detecção em todos os níveis de confiança de 50% a 95%, e na métrica mAP50, que é limitada a detecções com nível de confiança de 50%, pode-se observar que o modelo *medium* manifesta o maior valor, correspondendo a 71,6% e 85% dessas métricas de desempenho, respectivamente, assim como manifesta maior valor na métrica de *recall*, equivalente a 80,6%.

5.2.1 Resultados via YOLOv8 nano

As métricas de avaliação da YOLOv8 nano são apresentadas na Tabela 5.2, onde mostra valor significativo na precisão, equivalente a 92,7% quando comparados com os outros modelos, porém, isso se difere nas demais métricas, com valores inferiores, sendo o recall, mAP50 e mAP50-95 equivalente a 78,3%, 83,7% e 69,7%, respectivamente. Dessa maneira, a Figura 5.2 é a representação em formas de gráficos do desenvolvimento das métricas ao longo das épocas treinadas.

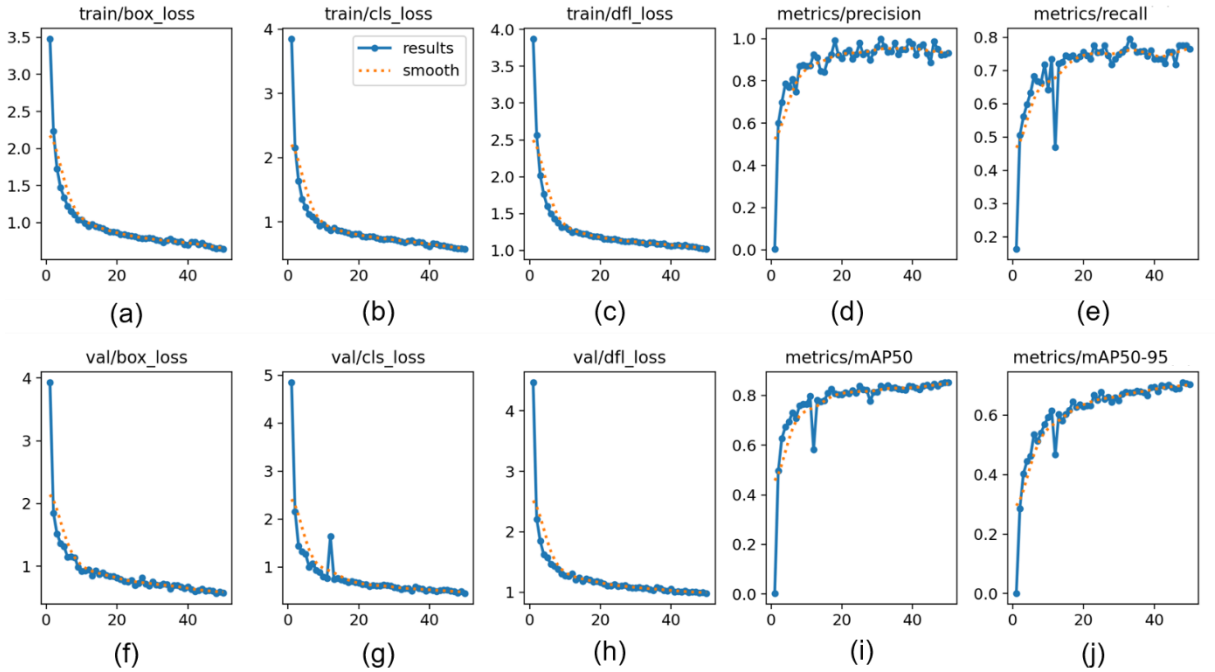

Figura 5.2: Comparação dos resultados das métricas de desempenho de precisão, recall, mAP 50% e mAP 50-95% para o algoritmo YOLOv8 nano.

Conforme a Figura 5.2, as métricas de desempenho do primeiro teste são apresentadas em forma gráfica, onde (a) a (e) referem-se à fase de treinamento e (f) a (j) referem-se à fase de validação. A perda observada na Figura 5.2 (a) está relacionada às caixas delimitadoras em relação aos objetos encontrados pelo algoritmo, onde há uma perda associada às coordenadas em relação ao centro do objeto com as extremidades das caixas. A Figura 5.2 (b) é uma perda associada à classificação das caixas em relação aos objetos encontrados pelo algoritmo, referente ao IoU e por fim, na Figura 5.2 (c) é uma perda associada ao Local Density-Free, onde sua função é ajustar o modelo treinado e regular a densidade dos objetos nas diferentes regiões das caixas delimitadoras, principalmente quando os objetos estão próximos uns dos outros. Assim, a Figura 5.2 (a), (b) e (c) mostram uma relação

inversamente proporcional com o número de épocas treinadas e as perdas, indicando que o desempenho de treinamento da rede melhora com a evolução das 50 épocas.

A Figura 5.2 (d) e (e) referem-se às métricas de precisão e *recall*, respectivamente. Percebe-se que quanto maior o número de épocas, maior o valor das métricas de desempenho, que são diretamente proporcionais. Isso significa que o modelo treinado está mais próximo do classificador ideal conforme Equações 2.12 e 2.13. A Figura 5.2 (e) a (j) seguem o mesmo raciocínio da fase de treinamento, mas isso acontece na fase de validação, que também apresenta um bom desempenho em relação ao classificador, embora as métricas mAP50 e mAP50-95 tenham oscilado ao decorrer das épocas.

Dessa maneira, conforme o conjunto de dados citado no Capítulo 3, a matriz de confusão resultante do teste em relação ao modelo nano, pode ser observada na Figura 5.3.

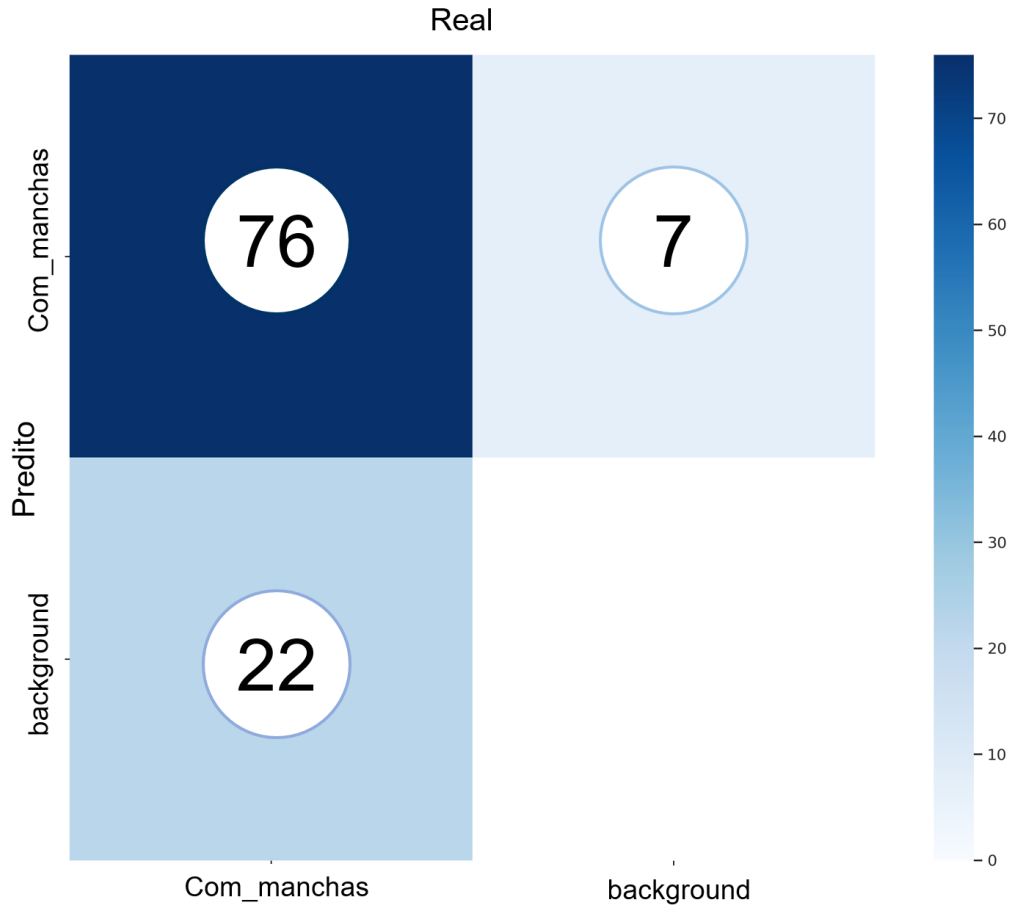

Figura 5.3: Matriz de confusão do modelo nano.

Conforme as anotações dos objetos para a criação das bounding boxes citados no Capítulo 3, a matriz de confusão do modelo nano é analisada no parágrafo a seguir.

Todas as imagens consideradas sem defeitos, ou seja, sem manchas de óleos, são categorizadas como background. Portanto, o classificador detectou 76 objetos denominados “Com_manchas”, 22 imagens não foram detectadas como tendo manchas, mas estavam entre as que deveriam ser classificadas, e 7 imagens foram consideradas como tendo manchas, mas não são desta classificação e poderiam estar associadas a instâncias chamadas sósias de manchas de óleo.

5.2.2 Resultados via YOLOv8 *small*

As métricas de avaliação da YOLOv8 *small* são apresentadas na Tabela 5.2, onde mostra valor significativo na precisão, equivalente a 92,1%, porém, isso se difere nas demais métricas, com valores inferiores, sendo o recall, mAP50 e mAP50-95 equivalente a 77,6%, 84,9% e 71,1%, respectivamente. Quando comparado com o nano, pode-se perceber que os resultados foram de valores bem próximos, mas não superior a ele. Da mesma maneira, a Figura 5.4 é a representação em formas de gráficos do desenvolvimento do modelo treinado ao longo das 50 épocas.

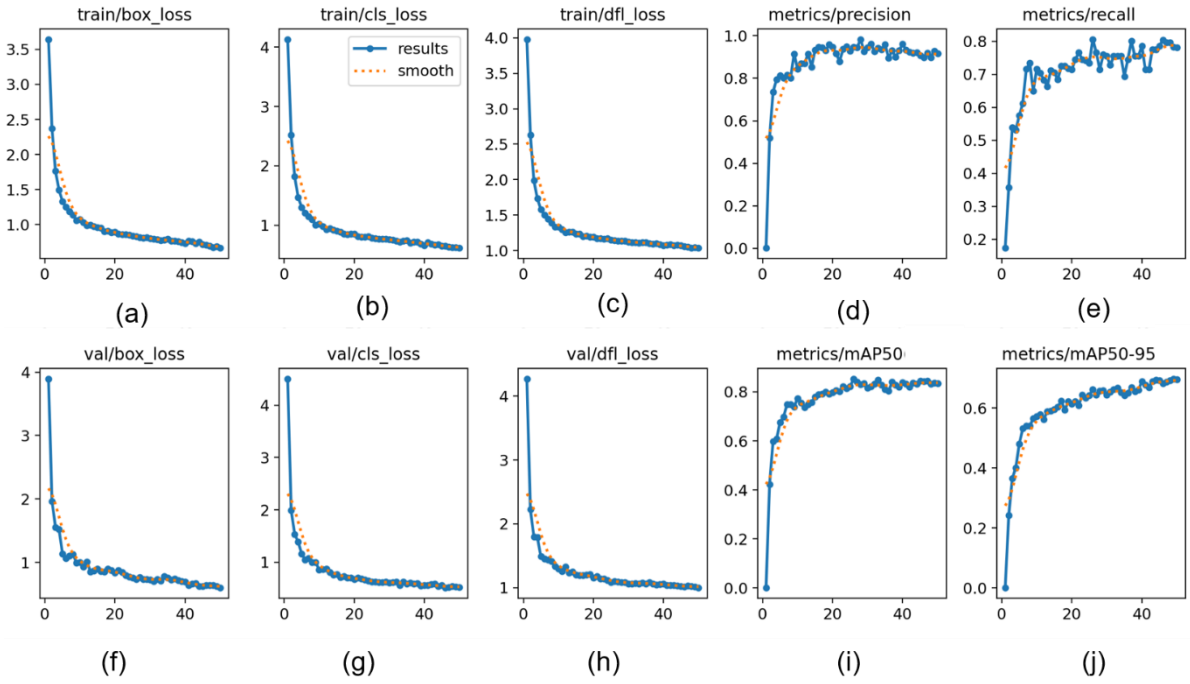

Figura 5.4: Comparação dos resultados das métricas de desempenho de precisão, recall, mAP 50% e mAP 50-95% para o algoritmo YOLOv8 *small*.

O mesmo conceito da Figura 5.3 aplica Figura 5.4. Nesse caso, os gráficos (a), (b) e (c) mostram as perdas associadas às caixas delimitadoras. Porém, os gráficos (d) e (e) mostram que as métricas de precisão e recall oscilaram fortemente com a evolução das épocas, principalmente na curva (e), uma vez que a proporção de amostras verdadeiras positivas classificadas corretamente pelo classificador não foi satisfatória em relação ao número total de amostras positivas, conforme visto na Equação (2). Quanto à fase de validação, os gráficos (f) a (h) seguem o mesmo conceito em relação às perdas das caixas delimitadoras, já os gráficos (i) e (j) referem-se às métricas da Equação (3), ou seja, são satisfatórias com em relação ao primeiro teste, conforme visto nos gráficos (i) e (j) da Figura 5.3

A matriz de confusão para o modelo *small* é apresentado na Figura 5.5 a seguir.

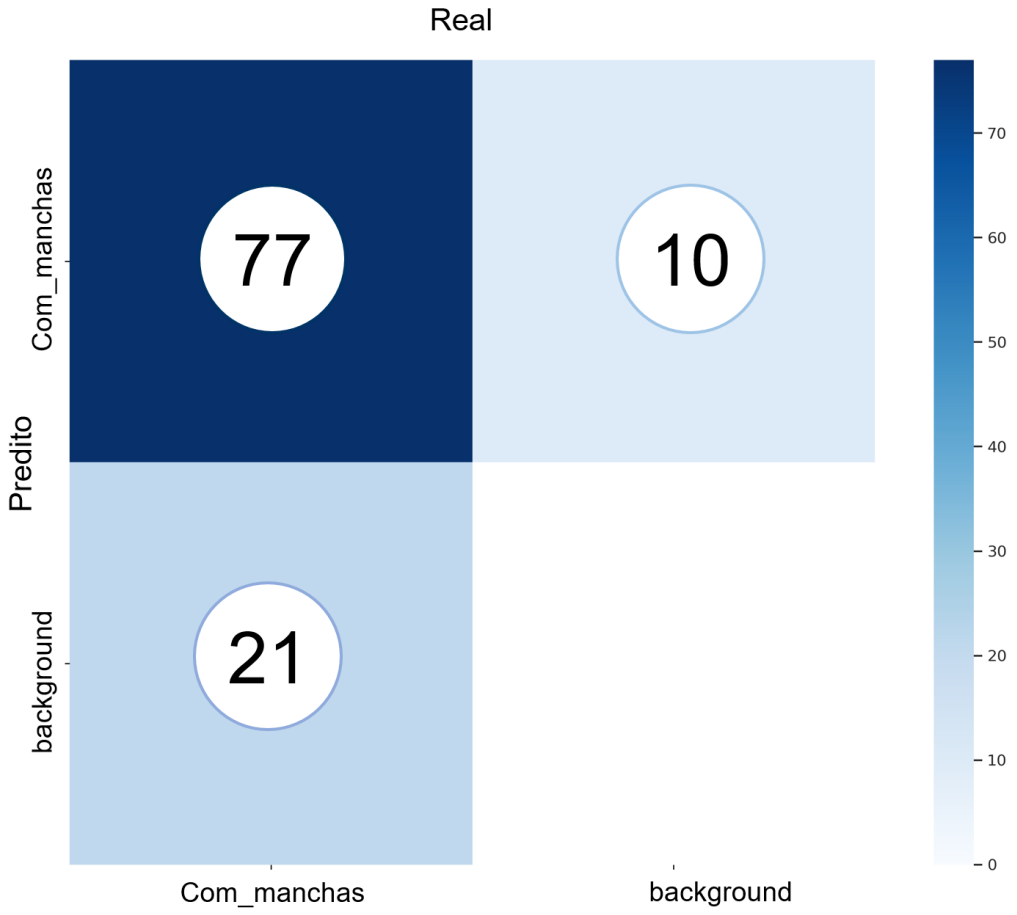

Figura 5.5: Matriz de confusão do modelo *small*.

Da mesma maneira, o classificador detectou 77 objetos denominados “Com_manchas”, 21 imagens foram detectadas como FN, mas estavam entre as que deveriam ser classificadas e 10 imagens foram consideradas como tendo manchas,

mas não são desta classificação, classificando um FP e poderiam estar associadas a instâncias chamadas sósias de manchas de óleo.

5.2.3 Resultados via YOLOv8 *medium*

As métricas de avaliação da YOLOv8 *medium* apresentadas na Tabela 5.2, mostra valor significativo no recall, mAP50 e mAP50-95, equivalente a 80,6%, 85% e 71,6%, respectivamente, quando comparados com os outros modelos, porém, isso se difere na métrica da precisão, com valor inferior ao modelo nano. Da mesma maneira, a Figura 5.6 é a representação em formas de gráficos do desenvolvimento do modelo treinado ao longo das 50 épocas.

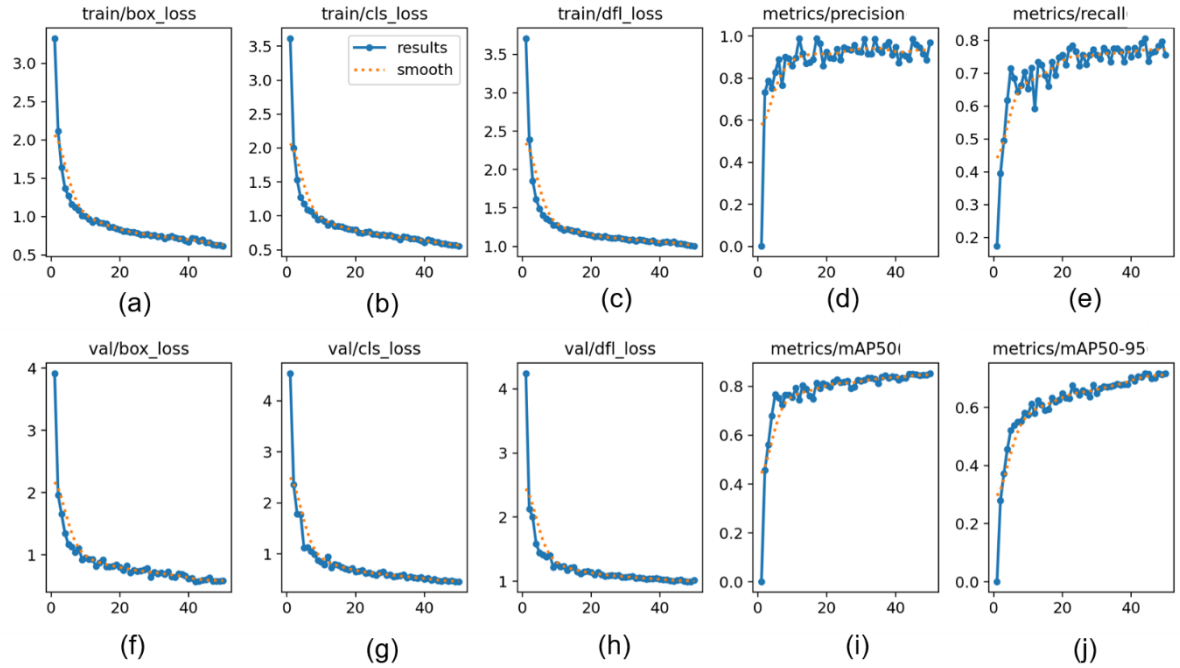

Figura 5.6: Comparação dos resultados das métricas de desempenho de precisão, recall, mAP 50% e mAP 50-95% para o algoritmo YOLOv8 *small*.

As perdas apresentadas na Figura 5.6 (a), (b) e (c) seguem o mesmo conceito dos demais gráficos (a), (b) e (c) das Figuras 5.2 e 5.4, ou seja, a tendência diminui com os períodos de treinamento, o que é satisfatório para o treinamento, isso significa que o classificador está mais assertivo em relação as funções de perda, como visto na Seção 3.2.2.5 do Capítulo 3.

Já as métricas de precisão e recall mostradas nos gráficos (d) e (e) da Figura 5.6 oscilaram ao longo das épocas, mas atingiram um valor correspondente a 89,1% para precisão e 80,6% para recall. Na fase de validação, conforme Figura 5.6 (i) e (j),

atingiu cerca de 85% e 71,6% para mAP50 e mAP50-95, respectivamente. Esses valores foram superiores aos dos testes com YOLOv8 nano e *small*.

Comparando os testes e métricas, percebe-se que o modelo médio YOLOv8 é o mais eficaz na detecção de manchas de óleo em ambientes aquáticos. Assim, utilizando 110 imagens com 98 instâncias de classificação para apenas uma classe: “0: Com_manchas”, podemos observar a matriz de confusão na Figura 5.7.

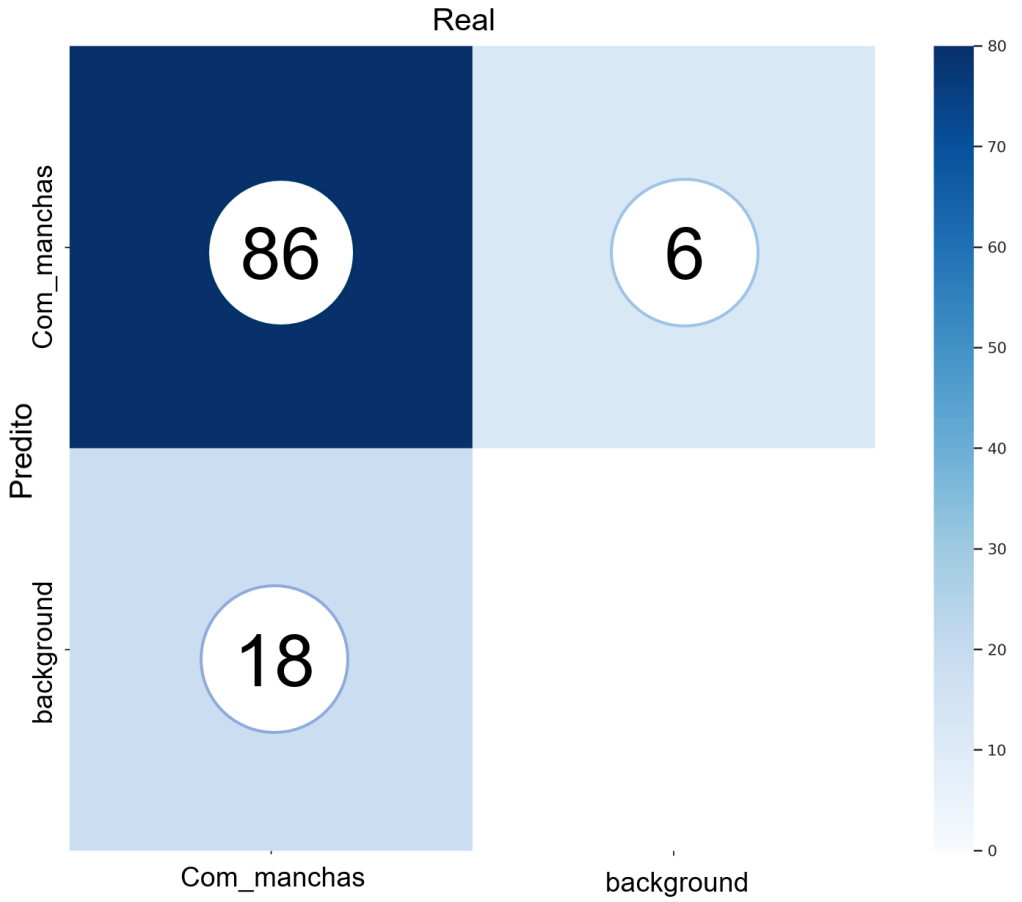

Figura 5.7: Matriz de confusão do modelo *medium*.

Da mesma maneira, o classificador detectou 86 objetos denominados “Com_manchas”, 18 imagens foram detectadas como FN, mas estavam entre as que deveriam ser classificadas e 6 imagens foram consideradas como tendo manchas, mas não são desta classificação, classificando um FP e poderiam estar associadas a instâncias chamadas sósias de manchas de óleo.

Na análise da fase de teste do classificador, a Figura 5.8 (a) mostra a imagem com as *bounding box* e a classificação em (%) do objeto previsto, e a Figura 5.8 (b) é a representação da mancha de óleo detectada pela máscara.

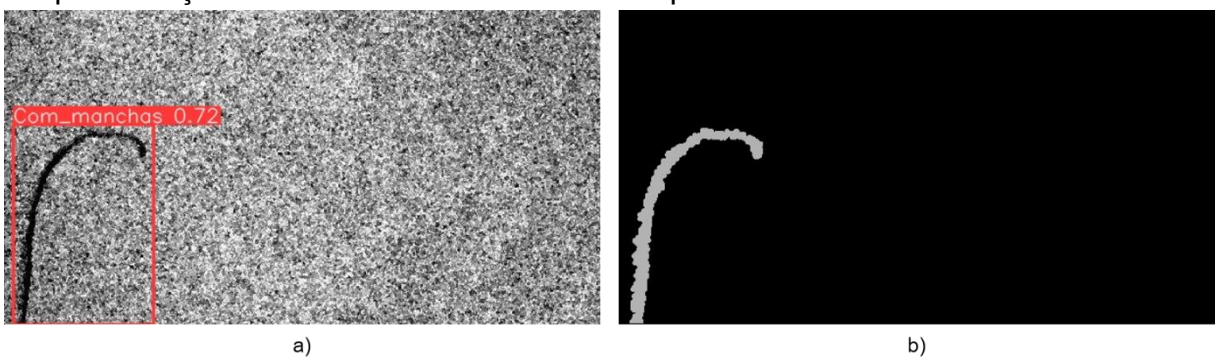

Figura 5.8: Resultados do algoritmo médio YOLOv8: (a) Detecção do resultado previsto pelo classificador; (b) representação da detecção em uma máscara.

Conforme mostrado na Figura 5.8, o modelo médio YOLOv8 reconhece e classifica a imagem de teste conforme esperado nas fases de treinamento e validação. Para mais detalhes sobre o resultado do teste, a Figura 5.9 é um compilado de algumas imagens mostrando a detecção e classificação das manchas de óleos em superfícies oceânicas.

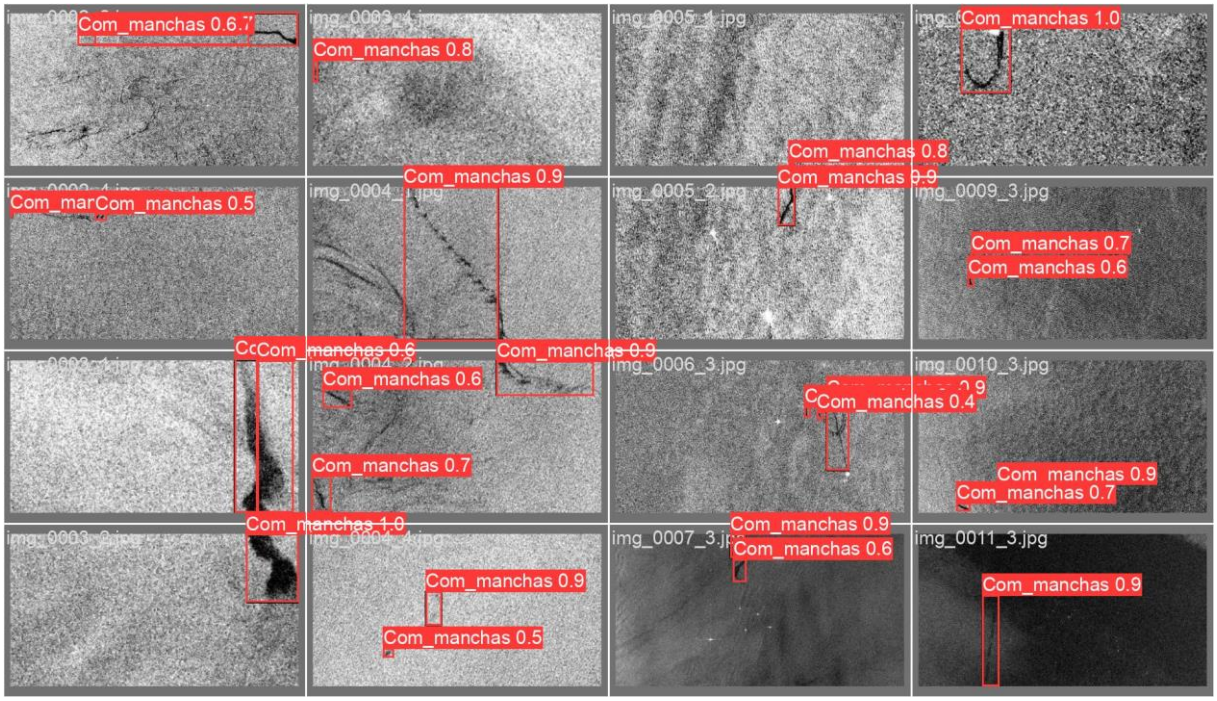

Figura 5.9: Resultados da detecção e classificação do algoritmo YOLOv8 *medium*.

Capítulo 6

CONCLUSÃO

Neste capítulo apresenta-se a conclusão e comentários gerais da dissertação, evidenciando as principais contribuições e propostas. Como também, as sugestões de metodologias e temas para trabalhos futuros relacionados ao monitoramento, detecção e classificação de derrames de óleos em superfície oceânica são expostas neste capítulo.

6.1 Conclusões

Com a crescente globalização do transporte marítimo, surgiu a necessidade de técnicas capazes de monitorar, detectar e classificar o ambiente aquático. Portanto, foi apresentada uma alternativa para mitigar derramamentos de óleo nesses ambientes através da aplicação do algoritmo YOLOv8.

Na análise dos algoritmos da família YOLOv8, as métricas de precisão, recall, mAP50 e mAP50-95, comparado aos demais modelos testados, o modelo *medium* se destacou com melhor desempenho, seu desempenho em termos de tempo de treinamento e velocidade computacional foi superior do que os outros. Portanto, seu desempenho ainda é satisfatório para atender aos requisitos do problema, uma vez que alcançou valores de confiança significativos com relação a objetos rotulados com derramamentos de óleo.

6.2 Contribuições

Esta dissertação tem como objetivo analisar o desempenho de algoritmos de aprendizagem profunda para a detecção e classificação de manchas de óleos em superfície oceânica, dessa maneira, visando monitorar e mitigar os efeitos nocivos causados pelo ser humano, em decorrência da crescente demanda de rotas de transportes marítimos e incidentes no mar.

Dessa forma, com a implementação da YOLOv8, de acordo com as técnicas utilizadas nesta dissertação, além da preocupação dos efeitos socioambiental causados por essa problemática, a solução do problema está baseada em um detector e classificador das manchas de óleos em ambientes aquáticos, como monitoramento eficiente nesses ambientes, consequentemente, gerando alertas para as autoridades competentes para agir de forma rápida e em tempo real para combater os danos causados quando há derrames de óleos.

A metodologia abordada nesta dissertação, pode ser utilizada para a aplicação em *hardwares*, como por exemplos em sistemas portuários, em um nó de sensor para rede de sensores sem fio (RSSF), em veículo aéreo não tripulado (VANT) e em veículo submarino autônomo (AUV).

Portanto, com a contribuição desta dissertação, iniciou-se o processo da viabilização de embarcar o algoritmo desenvolvido. Conforme visto na Seção 1.4, os estudos sobre o tema são promissores em relação a aplicação da metodologia abordada nesta dissertação, como também da sua implementação em sistemas embarcados.

6.3 Trabalhos Futuros

No decorrer do desenvolvimento deste trabalho, várias ideias surgiram como forma de estudos adicionais, porém, em função do tempo para conclusão desta dissertação, recomenda-se uma lista das principais ideias as quais necessitam de um estudo mais aprofundado:

- Realizar testes com os demais modelos do algoritmo da família YOLOv8 e das versões v4, v5.
- Realizar uma comparação entre os algoritmos utilizados nesta dissertação com outras técnicas de redes neurais artificiais.
- Realizar um estudo sobre sistemas embarcados em relação a temas como energia, restrição memória e custo computacional.
- Embarcar o algoritmo proposto em nó de sensor ou VANT ou AUV.

REFERÊNCIAS BIBLIOGRÁFICAS

Almeida, Carlos Caetano de. Identificação e classificação de imagens usando rede neural convolucional e "*machine learning*": implementação em sistema embarcado. 2019. Recurso online (202 p.) Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica, Campinas, SP. Disponível em: https://hdl.handle.net/20.500.12733/1638173. Acesso em: 28 dez. 2023.

Bochkovskiy, Alexey; Wang, Chien-Yao; Liao, Hong-Yuan Mark. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

Cardoso, A. M. Sistema de informações para planejamento e resposta a incidentes de poluição marítima por derramamento de petróleo e derivados. 2007, p. 1-148, MSc, Programa de planejamento energético, COPPE/UFRJ, 2007.

Diana, L.; Xu, J.; Fanucci, L. Identificação de derramamento de óleo a partir de imagens SAR para sistemas embarcados de baixa potência usando CNN. *Sensor Remoto* **2021** , *13* , 3606. https://doi.org/10.3390/rs13183606.

Guo, K. *et al.* Angel-eye: A complete design flow for mapping CNN onto embedded FPGA. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, IEEE, v. 37, n. 1, p. 35–47, 2018.

Huang, Rachel; Pedoeem, Jonathan; Chen, Cuixian. YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE, 2018. p. 2503-2510.

ITOPF – International Tanker Owners Pollution Federation Limited: Oil Tanker Spill Statistics 2020. Disponível em: <https://www.itopf.org/knowledge-resources/data-statistics/statistics/>. Acesso em 05 de agosto. 2023.

Krestenitis, M. O. (2019). Early Identification of Oil Spills in Satellite Images Using Deep CNNs. (C. Springer, Ed.) *In International Conference on Multimedia Modeling*, 424 - 435.

Pena, P. G. L. *et al.* Derramamento de óleo bruto na costa brasileira em 2019: Emergência em saúde pública em questão. **CPS – Caderno de Saúde Pública,** vol. 36, n° 2, 2020.

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 (CVPR). [S.l.: s.n.], 2016.

Samira Ahmed, Tamer ElGharbawi, Mahmoud Salah e Mahmoud ElMewafi (2023). Deep neural network for oil spill detection using Sentinel-1 data: application to Egyptian coastal regions, Geomatics, Natural Hazards and Risk, 14:1, 76-94, DOI: 10.1080/19475705.2022.2155998

Shaban, M.; Salim, R.; Abu Khalifeh, H.; Khelifi, A.; Shalaby, A.; El-Mashad, S.; Mahmoud, A.; Ghazal, M.; El-Baz, A. A Deep-Learning Framework for the Detection of Oil Spills from SAR Data. *Sensors* **2021**, *21*, 2351. https://doi.org/10.3390/s21072351

Siedel, R. Detecção de Defeitos na Fabricação Têxtil Utilizando Yolov5. Dissertação de Mestrado: Computação Aplicada – Instituto Federal do Espírito Santo, Serra – ES. p.77, 2023.

Silva, P.R. Transporte marítimo de petróleo e derivados na costa brasileira: Estrutura e implicações ambientais. MSc, Programa de planejamento energético, COPPE/UFRJ, 2004.

Ultralytics docs 2023. Disponível em: < https://docs.ultralytics.com/pt/models/yolov8/#supported-tasks-and-modes >. Acesso em 15 de fevereiro. 2024.

WA Maawali, A. Al Naabi, M. Al Yaruubi, A. Saleem e AA Maashri, "Projeto e implementação de um veículo de superfície não tripulado para tratamento de derramamentos de óleo", 2019, 1ª Conferência Internacional *sobre Sistemas de Veículos Não Tripulados - Omã (UVS)* , Muscat , Omã, 2019, pp. 1-6, DOI: 10.1109/UVS.2019.8658267.

Wang, Chien-Yao et al. Cspnet: A new backbone that can enhance learning capability of cnn. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. [S.l.: s.n.], 2020. p. 390–391.

Wang, Chien-Yao; Bochkovskiy, Alexey; Liao, Hong-Yuan Mark. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXivpreprint arXiv:2207.02696, 2022.

WU, J. Introduction to convolutional neural networks. National Key Lab for Novel Software Technology Nanjing University, China, 2017.

Yi-Jie Yang, Suman Singha and Roberto Mayerle (2022). A deep learning based oil spill detector using Sentinel-1 SAR imagery, International Journal of Remote Sensing, 43:11, 4287-4314, DOI: 10.1080/01431161.2022.2109445.

Yoo, S.-m., Cho, C., & Lee, K. H. (2019). Structure of Deep Learning Inference Engines for Embedded Systems. *International Conference on Information and Communication Technology Convergence (ICTC)*. doi:10.1109/ictc46691.2019.8939843.

Zeng, K.; Wang, Y. Uma rede neural convolucional profunda para detecção de derramamento de óleo a partir de imagens SAR transmitidas pelo espaço. *Sensor Remoto* 2020 , *12* , 1015. https://doi.org/10.3390/rs12061015.

Zhuang, Fuzhen et al. A comprehensive survey on transfer learning. Proceedings of the IEEE, IEEE, v. 109, n. 1, p. 43–76, 2020.

Anexo A

ARQUITETURA DA YOLOV8

O algoritmo YOLO é utilizado para a detecção de objetos em tempo real. Ele utiliza uma abordagem de regressão e associa probabilidades para detectar objetos usufruindo de única rede neural convolucional para separar espacialmente as caixas delimitadoras e com as probabilidades a cada objeto detectado (Jocher et al., 2022).

A YOLO tem evoluído ao longo dos anos, com versões da v1 à v8 que abordam limitações e melhorias de detecção de objetos. Para o melhor entendimento da YOLOv8, é necessário compreender as limitações, modificações e melhorias ao longo dos lançamentos de cada versão (Ultralytics, 2023).

A primeira versão do YOLO, lançada em 2015, surgiu como uma mudança do básico ao inovador no âmbito da detecção de objetos. No entanto, possui limitações, como a dificuldade em detectar imagens menores dentro de um grupo e a incapacidade de detectar formas novas. Lançada em 2016, a YOLOv2 introduziu uma nova arquitetura de rede chamada Darknet-53, que melhorou a precisão e a velocidade da detecção de objetos. Utilizou de regressão logística para melhorar a previsão da caixa delimitadora e introduziu classificadores logísticos independentes para previsões de classe mais precisas.

A YOLOv3, lançada em 2018, foi desenvolvida com base na YOLOv2, realizando previsões em diferentes escalas, permitindo melhores informações semânticas e uma imagem de saída de maior qualidade. A YOLOv4, superou o YOLOv3 em velocidade e precisão, tornando-o a versão mais avançada do YOLO. Ela alcançou velocidade e precisão ideais em comparação com versões anteriores e outros detectores de objetos de última geração.

A YOLOv5, desenvolvido pela Meituan em 2020, uma empresa chinesa de comércio eletrônico, com foco em aplicações industriais, introduziu um design harmônico ao hardware, um *backbone* desacoplado eficiente e uma estratégia de treinamento mais eficaz, resultando em excelente precisão e velocidade no conjunto de dados COCO. A YOLOv6, também desenvolvido pela Meituan, continuou a

melhorar o design amigável ao hardware da YOLOv5. Introduziu um design de *backbone* e *neck* mais eficiente, aumentando ainda mais a precisão e a velocidade. O algoritmo YOLOv7 teve como objetivo melhorar a precisão da detecção sem aumentar os custos de treinamento, também foi lançado em 2022.

Utilizando as melhorias feitas no YOLOv7 como base, o YOLOv8 apresenta o melhor resultado às versões anteriores. Lançado em meados de 2023, tem se destacado em relação a localização de objetos (sejam por fotos e vídeos), tornando-a mais rápido e eficaz em tempo real.

A arquitetura da YOLOv8 envolve o redimensionamento da imagem de entrada, a aplicação de convoluções e a utilização de técnicas como a *bacth normalization* e o *dropout* para melhorar o desempenho. De acordo com Seidel (2023), a primeira camada é conhecida como *Backbone*, a segunda como *Neck* e a terceira como *Head*.

A Figura A.1 é a representação da arquitetura da YOLOv8, observa-se que está dividida conforme as camadas citadas anteriormente, a *backbone*, *neck* e a *head*.

A camada de *backbone*, está referenciada na arquitetura da YOLOv4, ela é uma modificação da arquitetura da CSPDarkNet53. Esta contempla 53 camadas convolucionais e emprega estáguios parciais para melhorar o fluxo de informações entre as outras camadas. Uma vantagem dessa arquitetura para a YOLOv8 está associada em detectar imagens em multiplas escalas, por isso, utiliza uma rede em forma de pirâmide, ela detecta e prever objetos de diversos tamanhos e formas em apenas uma imagem.

A segunda camada, refere-se a uma camada *neck*, sua função é reunir mapas de características de diversos estágios. Esse processo otimiza o desempenho e a propagação de informações semânticas de maneira mais eficaz, viabilizando através de sua ativação, diferentes tipos de percepções.

A camada *head* consiste em múltiplas camadas convolucionais seguidas por uma série de camadas totalmente conectadas. Essas camadas são responsáveis por prever as *bounding boxes,* probabilidades de classificações para os objetos identificados nas imagens.

A Figura A.1 contem o fluxograma detalhado de acordo com as etapas de cada passo da arquitetura. Será detalhado a seguir esse fluxograma, seguido da explicação dos parâmetros da Tabela A.1 e por fim o passo a passo de toda a arquitetura da YOLOv8.

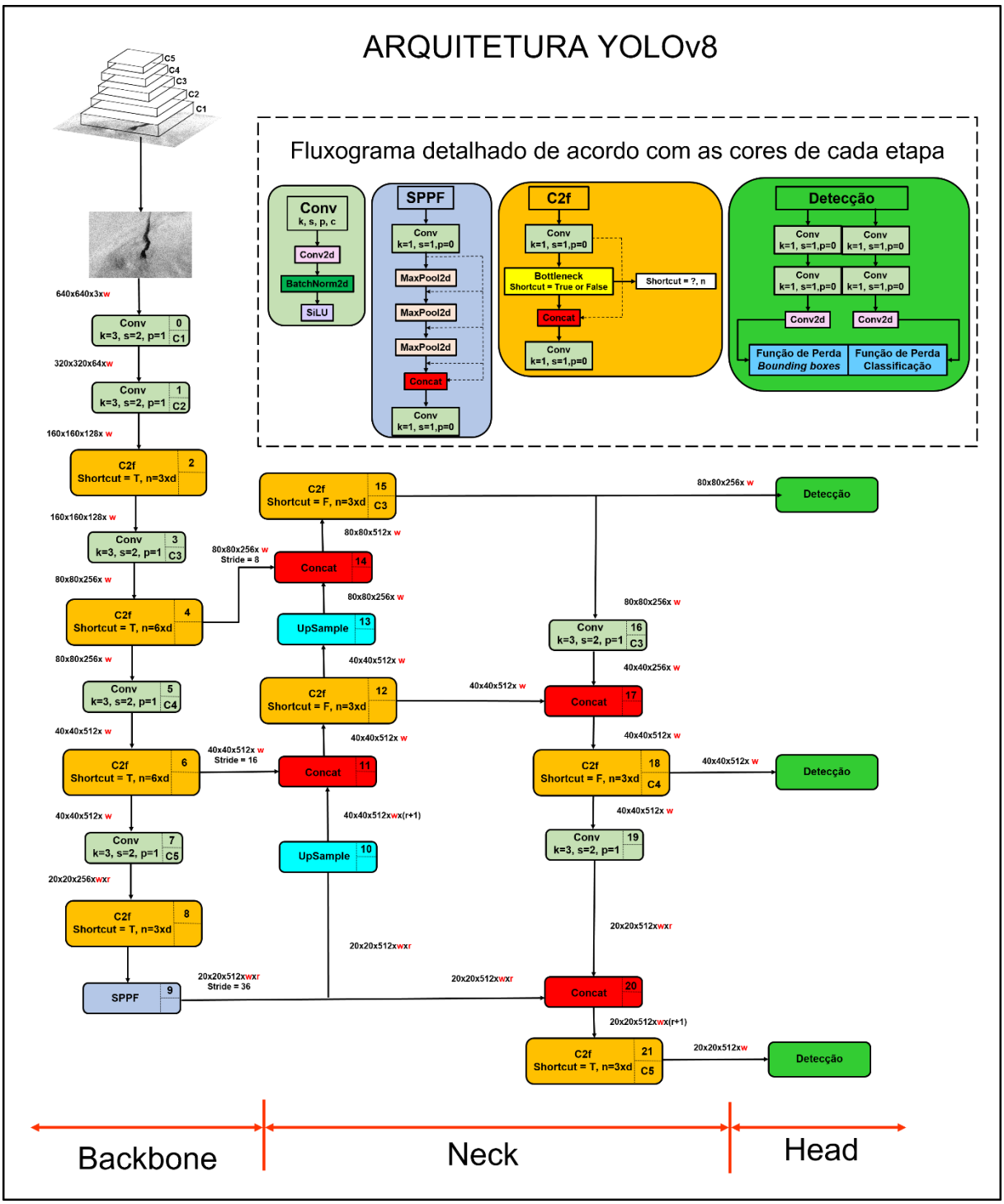

Figura A.1: Arquitetura da YOLOv8

Tabela A.1 – Parâmetros de profundidade, largura e canal de acordo com os modelos da YOLOv8.

Modelo	d (profundidade)	w (largura)	r (canal)
Nano	0.33	0.25	*1024*
Small	0.33	0.50	*1024*
Medium	0.67	0.75	*768*
Large	1.0	1.0	*512*
Extra large	1.0	1.25	*512*

Primeiramente, pode-se chamar por cores, onde o bloco 1 é identificado pela cor verde-musgo e denominado de bloco convolucional, o bloco 2 é identificado pela cor azul e denominado de bloco SPPF, o bloco 3 é caracterizado com a cor amarela e denominado de bloco C2f e por fim, o bloco 4 é representado pela cor verde-bandeira e denominado como bloco de detecção.

O bloco 1, consiste em uma camada convolucional caracterizado por *kernel* (k), *stride* (s) e *padding* (p), uma camada de normalização em lotes, denominado de *BatchNorm2d* e seguida por uma camada de função de ativação, denominada SiLU.

O bloco 2, denominado de SPPF (pooling de pirâmide espacial) é caracterizado como uma modificação de uma pirâmide espacial, ela contém com uma maior velocidade referente a SP. Dentro desse bloco, consiste em uma camada convolucional, seguido por camadas de extrações de caracteristicas, denominadas como *MaxPool2d*. Após as extrações, o resultado é concatenado pela camada *Concat*, e por fim, o resultado da concatenação é a entrada para mais uma convolução.

O bloco 3, contém em uma camada de convolução, onde sua saída é dividida para as seguintes camadas: *BottleNeck* e para *Concat* onde servirá como um atalho para o mapa de caracteristicas extraídos na camada da *backbone*. Após essa etapa, sua saída é utilizada como entrada para mais uma camada convolucional.

O bloco 4, acontece a detecção dos objetos nas imagens, o que a difere de outros blocos de detecção das versões anteriores é que na versão da YOLOv8 não é utilizado o conceito de âncoras, ou seja, as previsões acontecem nas células de cada grade dividida nas imagens de entrada. Neste bloco, contém duas trilhas de camadas, sendo a primeira para a predição das bounding boxes e a segunda como predição da

classe dos objetos. As duas trilhas seguem a mesma sequências de camadas, dois blocos de camadas convolucionais e uma única camada de bloco convolucional2d.

Outro fator importante para o entendimento do funcionamento da arquitetura do algoritmo, são os parâmetros expostos na Tabela A.1, que de acordo com cada modelo é apresentado a profundidade, largura e canal para cada convolucão.

Esses parâmetros auxiliam no cálculo gerado através das resoluções das imagens ao decorrer das convoluções. A Figura A.2 é uma exemplificação desses parâmetros.

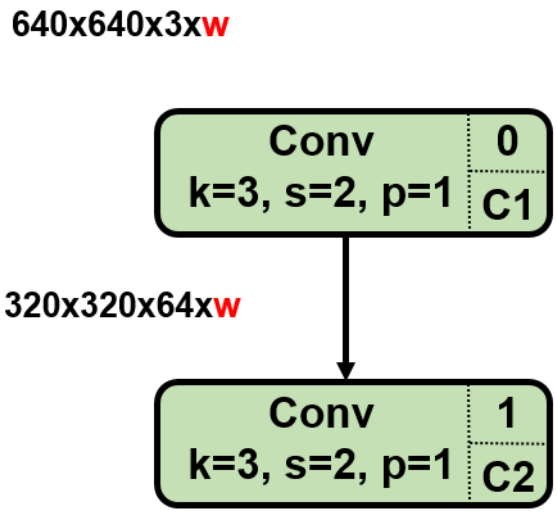

Figura A.2: Exemplo de parâmetros de uma camada convolucional da arquitetura da YOLOv8.

De acordo com a Figura A.2, o índice 0 indica que o bloco de convulução é o primeiro da arquitetura da rede, o k=3 é indica que o bloco convulucional tem *kernel* 3, s=2 indica que a distância desse bloco tem passo 2, ou seja, quanto menor a saída resultante, maior será essa camada de convolução e o p=1 indica que o *padding* para este bloco agrega valores 1 aos elementos da borda das imagens.

Dessa forma, a entrada dessa camada de convolução tem resolução 640x640x3. De acordo com o *stride*, essa convolução é reduzida quando o passo dois é utilizado, o que resulta em uma resolução na segunda convolução de 320x320x64xw, para obter o canal dessa resolução é necessário utilizar os parâmetros da Tabela A.1. E esse cálculo é referenciado em toda arquitetura da rede.

A entrada da YOLOv8 é uma imagem com 3 canais na camada de backbone e cada uma dessas camadas convolucionais do *backbone* extraem recursos distintos em vários níveis de resolução.

Do mesmo modo, o passo a passo da arquitetura é explicado nesse parágrafo. Como já mencionado, de acordo Figura A.2 existem índices e parâmetros de entendimento do passo a passo de funcionamento da arquitetura. Dessa forma, nas camadas da backbone, neck e head a estruturação continua a mesma.

Após as duas primeiras camadas de convoluções da *backbone*, segue para a camada C2f, ela contém dois parâmetros de atalho, denominados como *ShortCut* = *True* (T) ou *False* (F). Caso for T, servirá como atalho na camada *BottleNeck* (explicado no bloco 3), caso contrário não utilizará o atalho. O valor n da camada C2f é calculado através do valor da profundidade multiplicado por 3.

Em seguida, ocorre a terceira camada convolucional, representado como índice 3 e mais uma camada C2f, com índice 4 e assim sucessivamente, até na camada SPPF. A camada SPPF é usada após a última camada de convolução da *backbone*. Sua principal função é gerar uma representação de objetos de diversos tamanhos sem redimensionar a imagem e nem introduzir uma perda de caracteristicas importantes dessa imagem.

Na camada *Neck*, apresentada também na Figura A.1, ocorre a camada *UpSample*, sua função é aumentar a resolução do mapa de caracteristicas da camada SPPF para corresponder a à resolução da camada C2f de índice 6 da *backbone*. Em seguida, a camada *Concat* é utilizado para concatenar e combinar os parâmetros da C2f de índice 6, o que reduzirá o número de canais da camada C2f de índice 12, dessa maneira apenas a resolução permanecerá inalterada. A Figura A.3 é uma exemplificação do processo entre as camadas *backbone* e *neck* (pois são camadas totalmente conectadas).

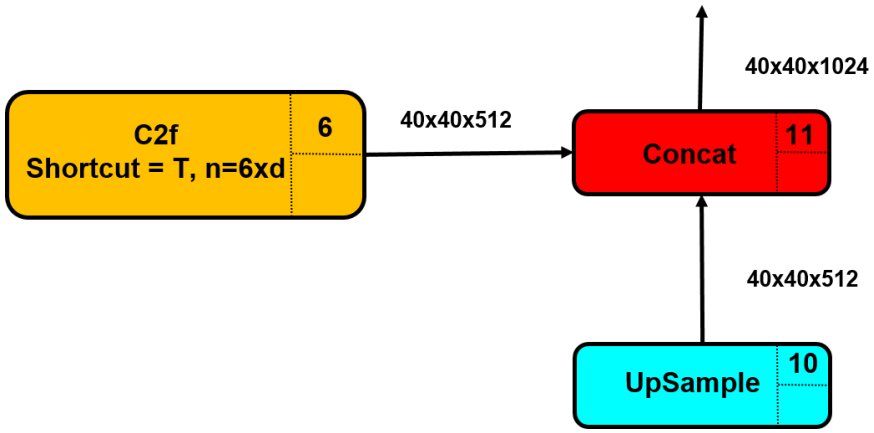

Figura A.3: Exemplificação do processo das camadas totalmente conectadas entre a *backbone* e *neck*.

Na Figura A.3 é possível a realização dos cálculos da Tabela A.1, dessa maneira, a saída da C2f de índice 6 é 40x40x512, a saída da *UpSample* de índice 10 é 40x40x512 e o resultado da concatenação é 40x40x1024.

A camada C2f da *neck* não implica em atalhos, pois o *ShortCut* = F, e isso permite a mesma linha de raciocínio até na camada C2f de índice 15. Esta camada é utilizada como entrada para a camada de detecção, esta camada de detecção, por sua vez, éespecializada para detectar objetos de pequeno porte. A saída da camada C2f de índice 15, também é utilizada como entrada para a camada convolucional de índice 16, o que reduzirá pela metade o número de canal utilizando a camada *concat*. Além disso, a camada *concat* combina o C2f de índice 12 com a camada C2f de índice 18, isso permite que a saída da C2f de índice 18 é usado como entrada para a segunda camada de detecção.

A segunda camada de detecção é especializada para detectar objetos de médio porte em uma image. A mesma saída da camada C2f de índice 18 servirá como entrada para o bloco convolucional de índice 19, seguido de mais uma camada de concatenação que por sua vez, é totalmente conectado com a camada SPPF, gerando assim mapa de caracteristicas para a camada de C2f de índice 21. A camada de C2f é a entrada da terceira camada de detecção do bloco head, gerando assim detecções de objetos de grande porte em uma imagem.