ISEL – Instituto Superior de Engenharia de Lisboa

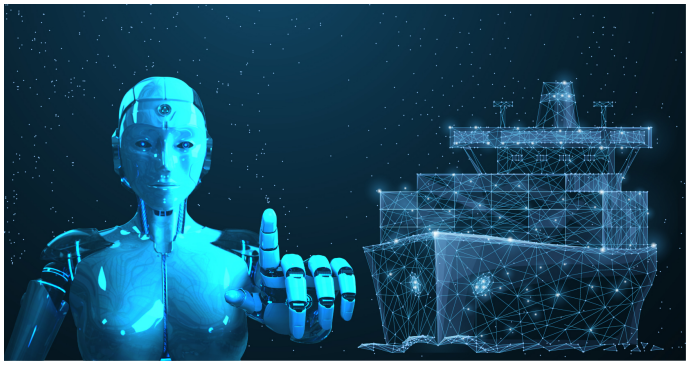
DEETC – Departamento de Engenharia de Eletrónica e Telecomunicações e de Computadores

MEIM

Mestrado Engenharia informática e multimédia

Aprendizagem Automática

Deteção e Reconhecimento Inteligente de Embarcações



Pedro Rodrigo Santarém Ferreira

Orientadores

Professor Doutor	Gonçalo Marques
Professor Doutor	Pedro Teodoro

Júri

Presidente *Professor Doutor*	Rui Jesus
Vogal *Professor Doutor*	Pedro Jorge
Vogal *Professor Doutor*	Gonçalo Marques

Setembro, 2023

ISEL – Instituto Superior de Engenharia de Lisboa

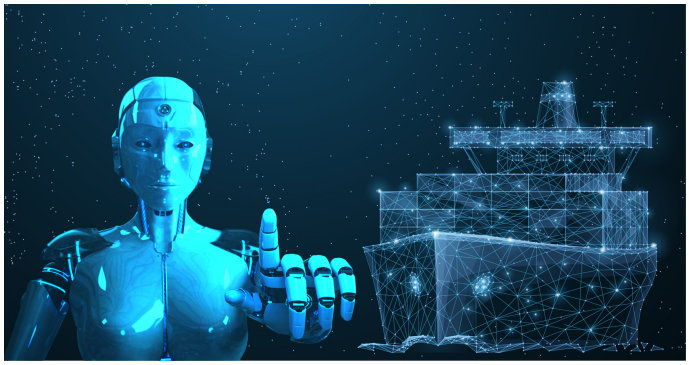
DEETC – Departamento de Engenharia de Eletrónica e Telecomunicações e de Computadores

MEIM

Mestrado Engenharia informática e multimédia

Aprendizagem Automática

Deteção e Reconhecimento Inteligente de Embarcações



Pedro Rodrigo Santarém Ferreira

Orientadores

Professor Doutor	Gonçalo Marques
Professor Doutor	Pedro Teodoro

Júri

Presidente *Professor Doutor*	Rui Jesus
Vogal *Professor Doutor*	Pedro Jorge
Vogal *Professor Doutor*	Gonçalo Marques

Setembro, 2023

Resumo

Este projeto foca-se na pesquisa e desenvolvimento de sistemas avançados para a deteção automática de embarcações. Nesta perspetiva, o trabalho está centrado na conceção e otimização de uma arquitetura denominada *Single Shot MultiBox Detector* (SSD), um algoritmo empregue na deteção de objetos em imagens. Este método é utilizado em visão computacional e sistemas de inteligência artificial para identificar e localizar objetos numa cena específica.

Com o intuito de alcançar esse objetivo, foi proposta uma nova arquitetura baseada no SSD. O objetivo primordial dessas modificações centrou-se na redução do custo computacional, visando tornar o sistema apropriado para a deteção em tempo real, sem comprometer o nível de precisão exigido.

Os resultados obtidos revelam que a nova arquitetura proposta do SSD demonstra uma redução no consumo de recursos computacionais, mantendo o seu desempenho na deteção. A arquitetura proposta consiste numa rede base mais otimizada, com um menor número de camadas de previsão e menos âncoras empregues por camada. Alcançou-se um mAP de 79.3% a uma taxa de 2 fps num sistema de computação com recursos limitados, como é o caso do Raspberry Pi 4. Esses 2 fps representam uma deteção em tempo real, considerando a natureza do problema, no qual as embarcações se movem a velocidades reduzidas.

Este trabalho representa um avanço no campo da deteção e classificação de embarcações, detendo o potencial para ser utilizado em aplicações práticas, incluindo a monitorização do tráfego marítimo e a segurança costeira, além disso, possui a capacidade de armazenar imagens devidamente categorizadas para a criação autónoma de conjuntos de dados de embarcações. Os resultados alcançados abrem portas para futuros desenvolvimentos e aperfeiçoamentos nesta área de aprendizagem automática que se encontra numa constante evolução.

Palavras-chaves: Deteção de Objetos, SSD, Aprendizagem Automática.

Abstract

This project focuses on the research and development of advanced systems for the automatic detection of vessels. From this perspective, the work is centered around the design and optimization of an architecture called Single Shot MultiBox Detector (SSD), an algorithm employed in detecting objects in images. This method is used in computer vision and artificial intelligence systems to identify and locate objects in a specific scene.

In order to achieve this goal, a new architecture based on SSD was proposed. The primary objective of these modifications focused on reducing computational costs, aiming to make the system suitable for real-time detection without compromising the required level of accuracy.

The results obtained reveal that the proposed new architecture of SSD demonstrates a reduction in computational resource consumption while maintaining its performance in detection. The proposed architecture consists of a more optimized base network, with a smaller number of prediction layers and fewer anchors employed per layer. An mAP of 79.3% was achieved at a rate of 2 fps on a computing system with limited resources, such as the Raspberry Pi 4. These 2 fps represent real-time detection, considering the nature of the problem in which vessels move at reduced speeds.

This work represents an advancement in the field of vessel detection and classification, holding the potential for practical applications, including maritime traffic monitoring and coastal security. Moreover, it has the capability to store properly categorized images for the autonomous creation of vessel datasets. The achieved results pave the way for future developments and enhancements in this constantly evolving field of machine learning.

Keywords: Object Detection, SSD, Machine Learning.

Agradecimentos

Gostaria de expressar os meus sinceros agradecimentos ao Professor Gonçalo Marques e ao Professor Pedro Teodoro pela orientação e apoio inestimáveis ao longo deste projeto. A vossa experiência e orientação foram fundamentais para o sucesso deste trabalho. Um agradecimento especial ao Dr. Gonçalo Marques por me ter introduzido à área de aprendizagem automática e ter despertado em mim um interesse profundo por este campo. Ao Dr. Pedro Teodoro, quero agradecer especialmente pelo apoio individual, não apenas no âmbito do projeto, mas também em diversas outras áreas, satisfazendo a minha curiosidade.

Além disso, quero manifestar a minha profunda gratidão à Escola Superior Náutica Infante D. Henrique por ter disponibilizado todo o material necessário, bem como o espaço essencial para a realização deste projeto.

Aos meus amigos, quero agradecer pelo apoio emocional constante e pelas respostas às minhas dúvidas de português, mostrando sempre uma disponibilidade incrível.

Por fim, não posso deixar de expressar o meu agradecimento a todos os meus professores ao longo destes anos. As vossas lições e orientações foram fundamentais para o meu crescimento académico e pessoal.

A todos, o meu muito obrigado por fazerem parte deste percurso e por terem contribuído para o meu sucesso neste projeto.

Pai e Mãe.

Índice

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Os sistemas de deteção e classificação automática de embarcações desempenham um papel de crescente importância nas operações marítimas e portuárias, bem como na gestão eficiente do tráfego aquático e na segurança marítima. A capacidade de identificar e detetar embarcações de forma automática e precisa é fundamental para garantir um funcionamento seguro e eficaz das atividades marítimas. Neste contexto, este trabalho propõe a implementação de um sistema avançado de deteção e classificação automática de embarcações, baseado em técnicas de redes neuronais profundas, tais como *Region-based Convolutional Neural Networks* (R-CNN), *Single Shot MultiBox Detector* (SSD) e *You Only Look Once* (Yolo).

Este trabalho propõe alcançar vários objetivos, tais como a expansão do conjunto de dados de embarcações, a conceção e a avaliação de um sistema de deteção e classificação automática de embarcações, explorando as capacidades de redes neuronais profundas. Além disso, é objetivo a execução da inferência desse sistema em tempo real num Raspberry Pi 4. Dado que a presença de embarcações é limitada na maioria do tempo, a execução contínua de um detector de objetos torna-se desnecessária. É nesse contexto que foi implementado um classificador binário, visando realizar uma filtragem inicial com o intuito de reduzir o constante uso computacional antes da ativação do detetor de objetos. Este método pretende estabelecer um sistema capaz de identificar com precisão uma variedade de tipos de embarcações, incluindo fragatas marítimas, graneleiros, lanchas de pilotos, navios-tanque, NPOs (Navios de Patrulha Oceânica) e porta-contentores, entre outros.

Para atingir estes objetivos, serão utilizadas técnicas de aprendizagem

profunda, que têm demonstrado um desempenho notável em tarefas de visão computacional, como a deteção de objetos. Os modelos serão treinados num conjunto de dados disponibilizado pelos docentes. O ênfase será colocada na otimização do desempenho e na eficiência computacional do sistema.

O sistema será testado no ambiente real do rio Tejo, próximo à Escola Superior Náutica Infante D. Henrique, proporcionando um cenário prático para avaliar a sua eficácia em condições de operação reais.

A estrutura deste trabalho está organizada da seguinte forma: o segundo capítulo apresenta uma revisão da literatura, destacando os principais conceitos e técnicas relacionadas com a deteção e classificação automática de embarcações. O terceiro capítulo é dedicado à exposição dos fundamentos teóricos relacionados com a aprendizagem automática, abordando desde os conceitos básicos de redes neuronais até à análise detalhada da deteção de objetos. O quarto capítulo detalha a metodologia adotada e a implementação, incluindo a descrição dos modelos de redes neuronais profundas utilizados e a preparação dos dados. O quinto capítulo expõe os resultados obtidos nos testes em tempo real, enquanto o sexto capítulo discute e analisa os resultados e apresenta conclusões finais.

Capítulo 2

Trabalho Relacionado

A revisão da literatura revela uma série de artigos de pesquisa relevantes que abordam a deteção de embarcações marítimas usando técnicas de aprendizagem profunda.

No âmbito das embarcações marítimas, têm sido conduzidos diversos estudos e projetos baseados em imagens de Radar de Abertura Sintética (SAR). Estas imagens oferecem uma representação minuciosa da topografia e textura da superfície. Devido ao conjunto de dados de embarcações utilizado no trabalho consistir em imagens de luz visível, serão abordados os estudos e artigos que se concentram no mesmo tipo de imagens. As pesquisas a seguir apresentadas exploram abordagens para a deteção e classificação de objetos em contextos marítimos.

No primeiro artigo estudado denominado *Deep Learning Based Multi-Modal Fusion Architectures for Maritime Vessel Detection* [1] são apresentados duas arquiteturas de fusão multimodal para deteção de embarcações usando dados de câmaras RGB e infravermelhas. A *Early Fusion Architecture* concatena imagens RGB e IR para entrada em uma rede de detetores, enquanto a *Middle Fusion Architecture* combina informações das duas câmaras por meio de métodos de fusão de imagem. O estudo avalia sete métodos de fusão e conclui que a *Middle Fusion Architecture* superou a *Early Fusion Architecture* em termos de precisão. Isso sugere que a fusão de informações de modalidades diferentes pode melhorar a deteção de objetos.

De forma a investigar o desempenho de detetores de objetos relativos a embarcações marítimas o artigo *ABOships—An Inshore and Offshore Maritime Vessel Detection Dataset with Precise Annotations* [2] foi consultado,

onde foram utilizados detetores de um e dois estágios, como SSD, EfficientDet, Faster R-CNN e R-FCN, previamente treinados em dados do Microsoft COCO. Os resultados mostram que o tamanho do objeto afeta a precisão da deteção e a precisão da deteção diminui com o tamanho do objeto. Para objetos pequenos, o detetor de melhor desempenho foi o Faster R-CNN. Para objetos de tamanho médio, o melhor desempenho foi obtido pelo SSD como extrator de recursos.

Uma outra fonte relevante no contexto da deteção de embarcações é o artigo *Image-based ship detection using deep learning* [3], onde é escolhido o algoritmo YOLO v2. É destacada a importância de modelos de CNN para extrair características e resolver problemas de deteção de objetos. Além disso, é mencionado que o algoritmo YOLO v2 supera os métodos de dois estágios em termos de velocidade e precisão (R-CNN). O artigo também discute o uso de transferência de pesos para melhorar o desempenho em cenários com dados limitados.

Por último é apresentado o artigo *SeaShips: A Large-Scale Precisely Annotated Dataset for Ship Detection* [4], onde descreve a criação do conjunto de dados "SeaShips", que contém imagens de navios capturadas por várias câmaras em cenários marítimos diversificados. Os detetores Faster R-CNN, YOLO e SSD, são avaliados em relação a esse conjunto de dados. Os resultados mostram que Faster R-CNN supera outros algoritmos, especialmente para objetos de tamanho médio. A discussão também inclui detalhes sobre a função de perda e o processo de treino dos modelos.

Em resumo, os artigos analisados abordam vários aspetos da deteção de embarcações marítimas usando técnicas de aprendizagem profundas, o desempenho de diferentes detetores e a criação de conjuntos de dados de referência. Proporcionam uma base de conhecimento específico e fundamental para investigadores e profissionais deste campo, enfatizando a relevância crítica da seleção de modelos e conjuntos de dados, crucial para alcançar deteções precisas em ambientes marítimos complexos.

Capítulo 3

Aprendizagem Automática

É um ramo da inteligência artificial que se foca no uso de dados e algoritmos para imitar a forma como os seres humanos aprendem, aprimorando gradualmente a sua aprendizagem. Este campo envolve a criação de modelos computacionais que podem aprender com os dados disponíveis e tomar decisões ou fazer previsões sem serem explicitamente programados para cada tarefa específica. O processo de aprendizagem é guiado pela análise dos dados, identificação de padrões e ajuste dos modelos para melhorar o desempenho ao longo do tempo. Isso tem aplicações numa ampla gama de áreas, incluindo reconhecimento de padrões, processamento de linguagem natural, visão computacional, medicina, finanças e muito mais.

3.1 Redes neuronais

As redes neuronais artificiais são constituídas por estruturas em camadas, compreendendo uma entrada, uma ou múltiplas camadas intermédias (também conhecidas como camadas escondidas) e uma camada de saída. Cada camada mantém conexões com a camada subsequente, e essas conexões individuais são atribuídas com pesos. Através de uma ponderação desses pesos e das entradas, os neurónios em cada camada efetuam cálculos e produzem saídas. Em seguida, é aplicada uma função de ativação, a qual decide se o neurónio deve ser ativado e passar informação para a camada seguinte.

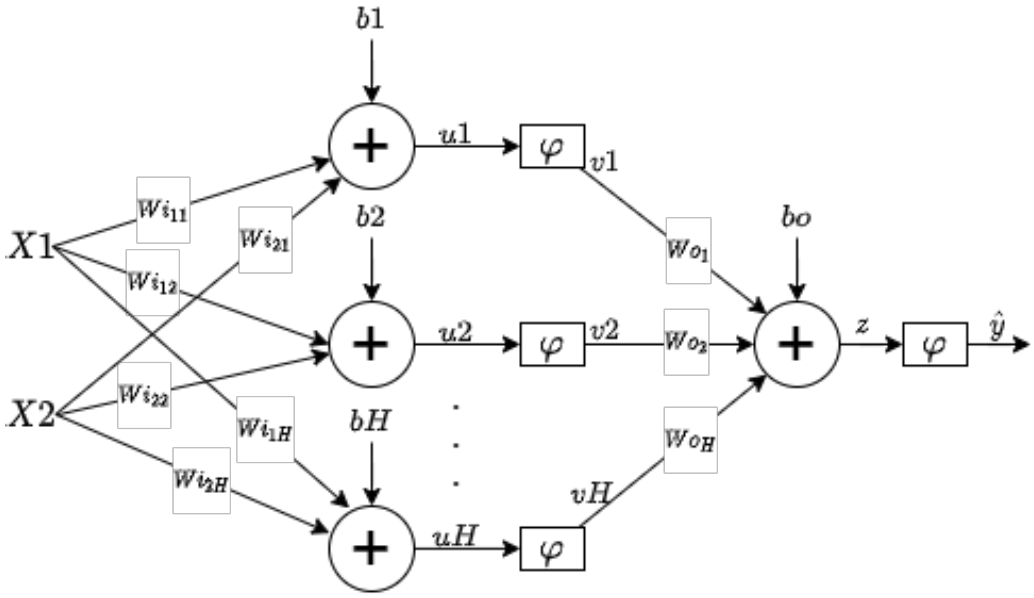


Figura 3.1: Rede neuronal

3.2 Treinar redes neuronais

No final de cada iteração de treino, é avaliado o valor da função de perda, indicando o desempenho da rede. Em seguida, recorre-se ao processo de *backpropagation* para calcular os gradientes da função de perda em relação aos pesos da rede. Esse procedimento possibilita à rede ajustar os pesos de modo a minimizar a função de perda.

3.3 Redes convolucionais

As Redes Neuronais Convolucionais (ConvNets ou CNNs) [5] são uma especialização de redes neuronais profundas construídas para processar dados que possuem uma estrutura espacial, como imagens. Revolucionaram a área de visão computacional e são amplamente utilizadas em tarefas como classificação de imagens, deteção de objetos e segmentação semântica. Uma das características distintivas das CNNs é a sua capacidade de capturar automaticamente características relevantes de um conjunto de dados, tornando-as altamente eficazes em questões relacionadas com a classificações de imagens.

3.3.1 Entrada das CNN

Para que as imagens possam ser recebidas por uma rede neural MLP, as matrizes são convertidas em vetores, enquanto para uma CNN, a imagem é recebida na sua estrutura original. Manter a matriz preserva as relações espaciais entre os píxeis, permitindo que a rede capture padrões em diferentes regiões da imagem [6]. Isso significaria que uma imagem a cor, onde possui três canais (RGB) com tamanho 32 x 32, teria 32 x 32 x 3 = 3072 entradas na rede.

3.3.2 Camada de convolução

As camadas de convolução são responsáveis por extrair características relevantes das imagens de entrada. Aplicam operações de convolução que envolvem uma janela deslizante, também chamada de filtro ou *kernel*, que percorre a imagem original [5]. O filtro é multiplicado elemento a elemento com os valores da região correspondente da imagem e, em seguida, esses produtos são somados para produzir um único valor na saída. Ao mover o filtro por toda a imagem, obtém-se um mapa de características que destaca diferentes padrões ou características presentes na imagem, como bordas, texturas e formas.

Neste exemplo (figura 3.2) é demonstrada a operação convolução numa imagem, aplica-se o filtro 2x2 nas regiões da imagem, formando uma matriz 2x2 de pesos [6]. Devido ao número de pesos não depender das dimensões da imagem mas sim do tamanho do filtros e do número destes, existe um

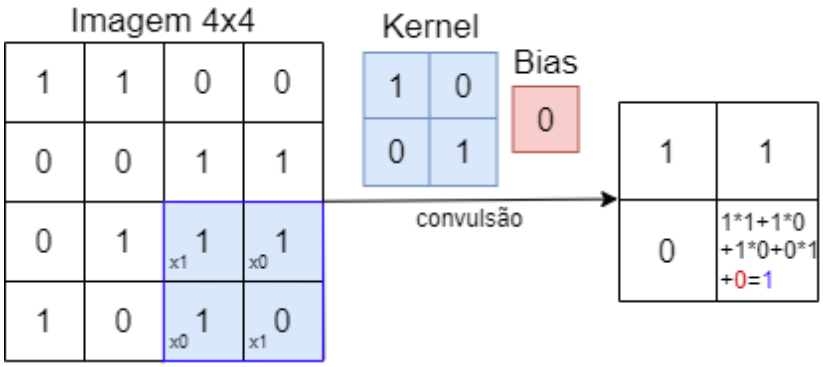


Figura 3.2: Operação de convolução

número de pesos exponencialmente reduzido em comparação com uma MLP. Quando os pesos da rede estão treinados, o que normalmente acontece é que as camadas iniciais detetam formas simples como contornos e já nas camadas mais profundas serão detetadas formas mais complexas. Além disso, a introdução de camadas de *pooling* (ou sub-amostragem) depois das camadas de convolução ajuda a reduzir a dimensionalidade do mapa de características, preservando as informações mais importantes. Isso torna a rede mais eficiente e reduz a possibilidade de *overfitting*.

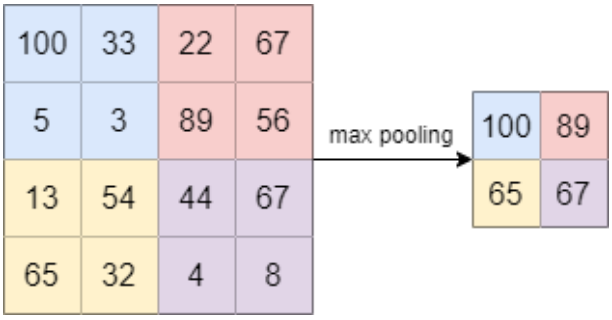


Figura 3.3: Max-pooling

Em suma, as redes neuronais convolucionais aplicam operações de convolução para extrair padrões e características relevantes das imagens de entrada. Esse processo de extração de características em várias camadas permite que as ConvNets capturem informações significativas e realizem tarefas de visão computacional com alto desempenho.

3.3.3 CNN com *Fully Connected Layer*

As Redes Neuronais Convolucionais (CNNs) têm a capacidade de identificar características específicas numa imagem. Entretanto, esse processo por si só possui um alcance limitado, pois não viabiliza a classificação, a qual é um dos propósitos centrais dentro do âmbito *Machine Learning*. É por essa razão que, após a última camada de convolução, é necessário conectá-la a uma rede MLP a fim de gerar um resultado de classificação fundamentado nas informações extraídas pelas CNNs.

Na figura 3.4, está presente uma imagem de dimensões 32 por 32 píxeis [6], que serve como entrada para uma camada convolucionais composta por um conjunto de três filtros. Posteriormente, essa camada é conectada a uma

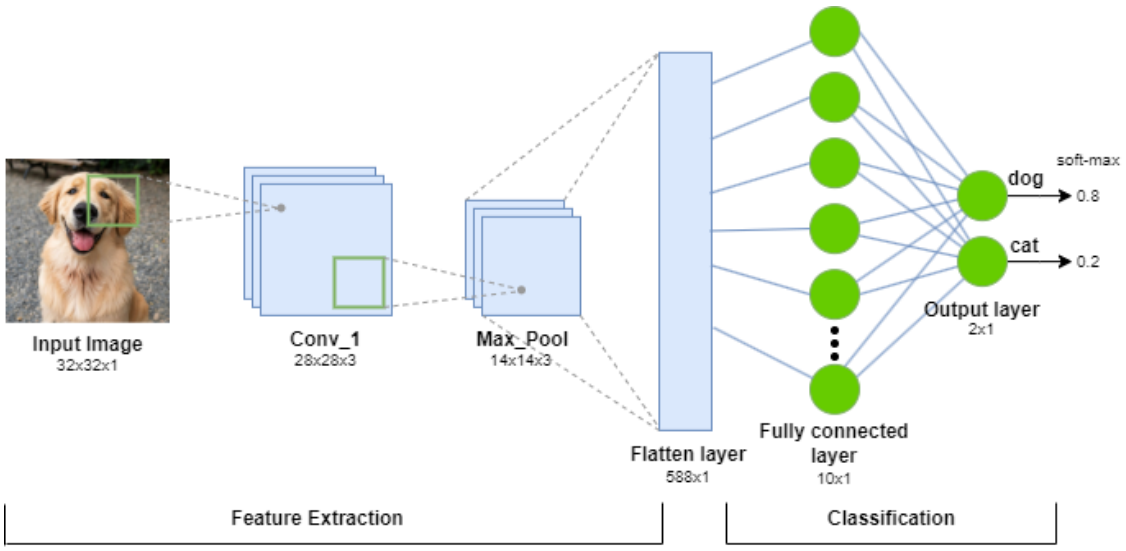


Figura 3.4: Arquitetura completa de CNN + Fully Connected Layer

camada de agrupamento (*pooling*) que visa reduzir a dimensão da matriz resultante pela metade. Na sequência, ocorre a conversão das matrizes num vetor, de modo a ser entrada para a camada de neurónios. Por fim, essa camada de neurónios é conectada à camada de saída, a qual, por meio da aplicação da função de ativação "Softmax", proporciona como retorno os valores probabilísticos associados a cada classe em questão.

A conexão entre uma CNN e uma MLP é crucial porque a CNN é especializada na extração de características, enquanto o MLP é especializado na tarefa de classificação. Isso resulta num sistema de classificação de imagens poderoso e capaz de reconhecer padrões complexos nas imagens de entrada.

3.3.4 Treino CNN

O processo de treino de uma CNN começa com a inicialização dos seus pesos de forma aleatória. Durante a fase de treino, a rede neuronal é alimentada com um amplo conjunto de imagens, cada uma delas associada a uma classe específica (por exemplo, gato, cão).

A CNN processa a imagem usando pesos iniciais aleatórios e ajusta os pesos para corresponder aos rótulos de classe. Isso é feito através da retro-propagação [5], que otimiza os pesos com base na diferença entre a saída esperada e a atual, cada iteração de treino, é conhecida como "época".

Ao longo de várias épocas, a CNN passa por sucessivas iterações de treino. A cada iteração, os pesos são ajustados em pequenas quantidades para refinar a precisão das previsões. Com o decorrer das épocas, a rede torna-se progressivamente mais precisa na classificação e previsão das classes das imagens de treino.

Após a conclusão do treino, a rede é avaliada usando um conjunto de dados de teste, que consiste em imagens rotuladas não utilizadas durante o treino. Essa avaliação permite medir a precisão e o desempenho da CNN em fazer previsões precisas para imagens não vistas anteriormente.

3.4 Deteção de objetos

Até o presente momento, foi delineada a forma como é possível utilizar redes neuronais com o propósito de realizar tarefas de classificação, especialmente no contexto de imagens. Embora essa abordagem de classificação possa solucionar uma ampla gama de problemas com notáveis taxas de desempenho, é importante reconhecer que ela pressupõe que cada imagem corresponde exclusivamente a uma única classe.

Surge, então, uma indagação: e se uma imagem obtiver múltiplas classes? Foi com essa ponderação em mente que emergiu a necessidade de conceber um modelo que fosse capaz de efetuar a classificação de múltiplas classes numa única imagem. Além desse requisito, o modelo também proporciona a determinação das coordenadas dentro da imagem que estão vinculadas a cada classe identificada.

É relevante observar que a etapa de localização pode ser considerada relativamente direta, dada a necessidade de acrescentar apenas quatro neurónios de saída além dos já presentes para fins de classificação. Esses neurónios adicionais seriam designados para representar as coordenadas x e y da localização, bem como a largura e a altura da caixa delimitadora do objeto. Tal disposição proporcionaria a informação essencial para situar um objeto numa imagem. No entanto, essa abordagem é limitada à circunstância em que cada imagem está estritamente associada a uma única classe. Esta limitação decorre da necessidade de acomodar diferentes números de saídas para cada imagem.

Essa visão conduziu à criação de um modelo que não somente realiza a

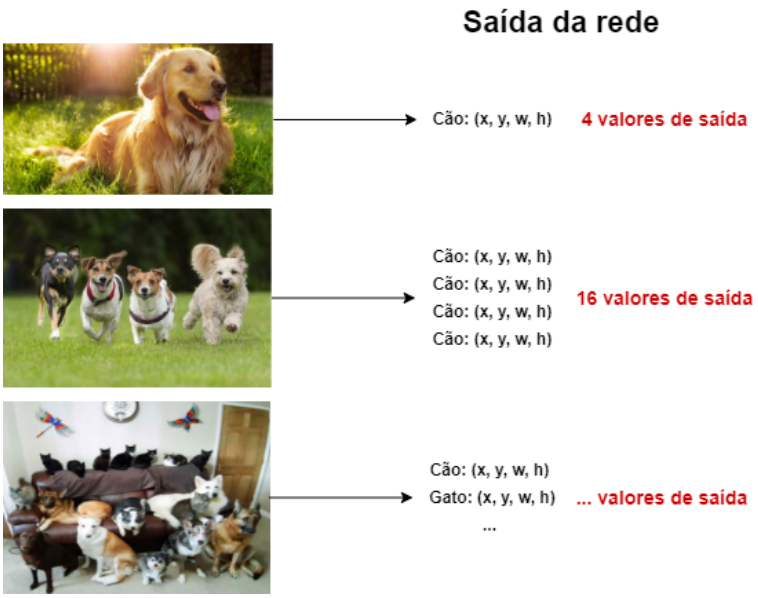


Figura 3.5: Diferentes valores de saída dependendo do número de objetos

classificação multi-classe, mas também oferece a capacidade de localização precisa dos objetos na imagem. Essa aptidão é manifestada por meio da delimitação visual, onde um retângulo é traçado em torno da região da imagem que se estima conter a classe específica em questão. No âmbito da visão computacional, essa abordagem é reconhecida como deteção de objetos.

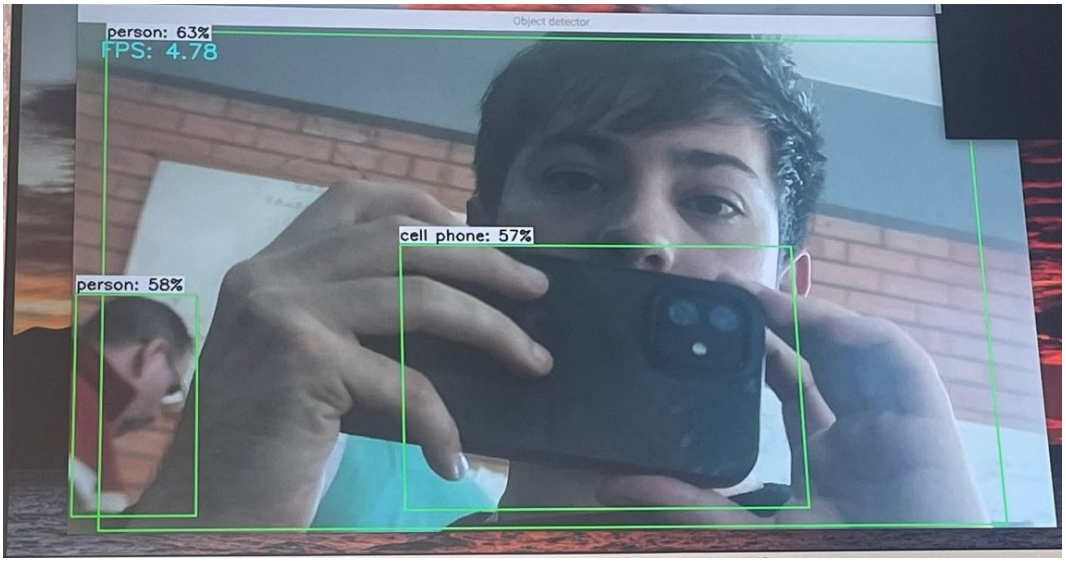


Figura 3.6: Exemplo de deteção de objetos

Conforme é evidenciado na figura 3.6, três objetos foram devidamente classificados e as suas respetivas localizações foram determinadas: duas pessoas e um telemóvel. Caso esta imagem fosse dada como entrada para um processo de classificação de imagens convencional, apenas uma das classes seria identificada. De seguida serão apresentados os principais métodos utilizados na deteção de objetos.

3.4.1 Janela deslizante

Uma abordagem simplista é a utilização de uma janela retangular de diversos tamanhos e escalas que varre a imagem. Essa janela atua como um "recorte" que é deslocado sobre a imagem original, classificando cada posição para determinar se contém ou não um objeto de interesse [7]. Esta técnica envolve a aplicação repetida de um classificador a cada posição da janela, a fim de identificar se existe ou não um objeto naquela região.

A abordagem de janela deslizante [15] apresenta algumas limitações significativas em termos de desempenho e eficiência, especialmente em cenários que requerem deteção de objetos em tempo real ou em imagens de alta resolução. Algumas dessas limitações são as seguintes:

1. **Complexidade Computacional**: O processo de percorrer a imagem

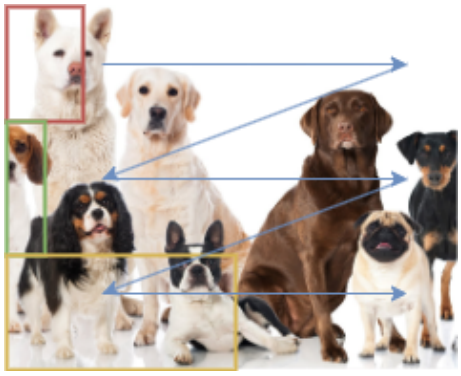


Figura 3.7: Metodologia de janela deslizante

usando janelas retangulares e a subsequente aplicação repetida de um classificador em cada posição pode ser computacionalmente exigente.

2. **Redundância de Cálculos**: Conforme a janela é deslocada ao longo da totalidade da imagem, diversas regiões sobrepostas são avaliadas de maneira iterativa. Esse procedimento resulta em cálculos redundantes, originando um incremento da carga computacional.

3. **Variação de Tamanhos**: A necessidade de aplicar essa técnica com diferentes tamanhos de janelas amplia ainda mais o custo computacional, tornando o processo mais demorado e intensivo em recursos.

4. **Desempenho**: Esta técnica enfrenta desafios em cenários onde objetos são muito grandes para a janela de deteção ou quando dois objetos se sobrepõem em várias janelas adjacentes, dificultando a sua deteção individual.

3.4.2 Faster R-CNN

O [23] Faster R-CNN é uma das últimas versões da família R-CNN, foi desenvolvido com o propósito de aprimorar o antigo Fast R-CNN, com ênfase particular na melhoria de sua velocidade, daí o seu nome. Essa abordagem [15] inovadora adota a utilização de redes neuronais para a geração de sugestões de regiões de interesse (RoIs), em substituição ao *selective search*, que é uma abordagem anteriormente usada no R-CNN e Fast R-CNN. Como resultado, a arquitetura pode ser descrita em duas fases principais [10]:

1. ***Region proposal network* (RPN)**: Aplica o RPN nos mapas de características provenientes das camadas de convolução iniciais. Atua como um classificador, identificando potenciais locais de objetos na imagem.

2. **Fast R-CNN**: É composta pelos mesmos princípios do Fast R-CNN, incluindo uma CNN pré-treinada inicial para a extração de características da imagem. Em seguida, são aplicadas camadas de *pooling* para redimensionar os mapas resultantes para tamanhos fixos. Por fim, a camada de saída é constituída por duas redes neuronais distintas: um classificador para determinar a classe e uma regressão linear para estimar a localização.

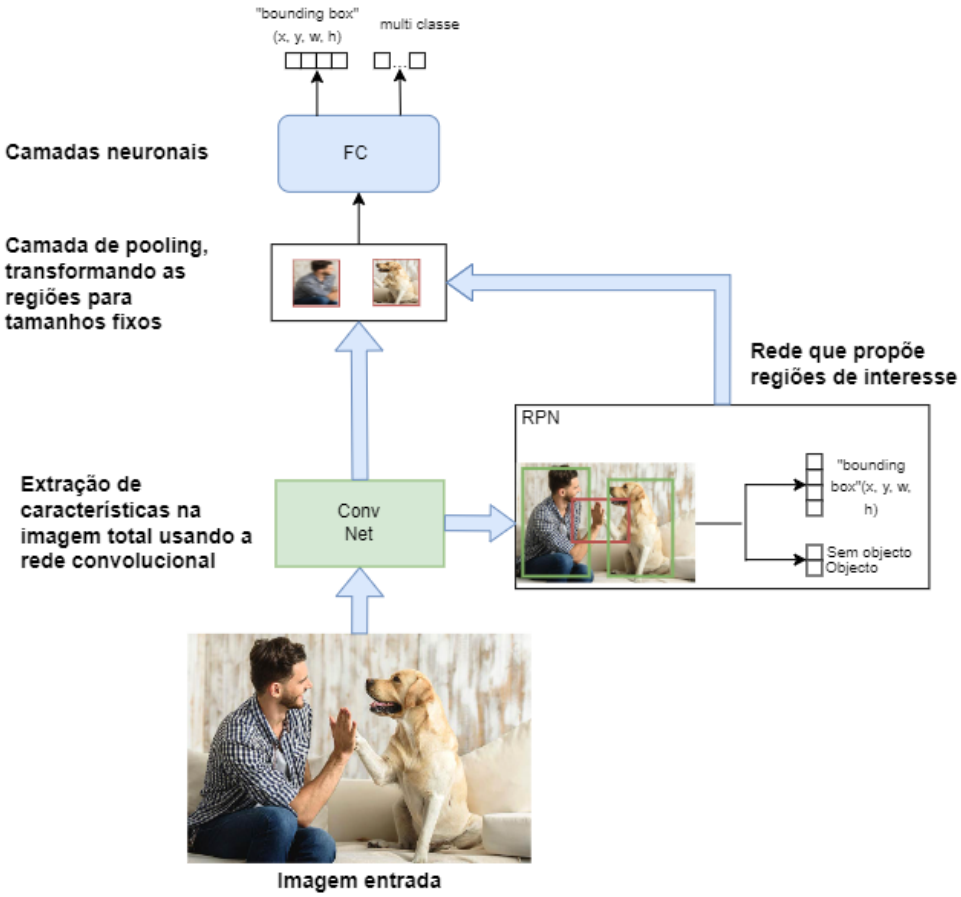


Figura 3.8: Arquitetura Faster-CNN

RPN

O RPN (*Region Proposal Network*) é encarregado de gerar regiões de interesse (RoIs) na imagem, que têm uma alta probabilidade de conter objetos de interesse. Essa rede é composta por uma camada de convolução que recebe os mapas de características da CNN de base e está conectada a duas camadas de saída [10]: uma camada de classificação e outra camada de previsão das *bounding-boxes*.

As regiões de interesse são identificadas mediante da utilização de caixas âncora, que são caixas delimitadoras com diferentes tamanhos e proporções (*aspect ratios*) predefinidas. Essa diversificação de caixas âncora visa acomodar objetos de diversos tipos e tamanhos em várias partes da imagem. Essa abordagem assemelha-se ao conceito de janela deslizante previamente explicado. Cada caixa âncora possui um ponto central denominado âncora, que é deslocado ao longo da imagem durante o processo.

Figura 3.9: Âncoras no Faster-RCNN

O processo de treino da RPN envolve uma avaliação minuciosa de cada caixa âncora. Em outras palavras, para cada caixa âncora, esta é comparada com a caixa delimitadora real da imagem (conhecida como *ground truth* ou gt). Durante essa comparação, é realizada uma operação chamada *Intersection over Union* (IoU), que quantifica o nível de sobreposição entre a proposta da caixa âncora e o objeto real na imagem.

Um IoU acima de 0.7 indica que a caixa âncora contém um objeto, enquanto um IoU abaixo de 0.3 sugere que a caixa âncora representa o fundo da imagem. Esse processo é crucial para identificar regiões de interesse que

precisam de análise adicional.

Figura 3.10: IoU entre âncora e *ground truth*

No estágio subsequente, as âncoras com pontuações indicativas de presença ou ausência de objetos passam por um classificador binário e um regressor de localização. Isso permite aprimorar a precisão e a adequação das propostas de regiões para deteção de objetos, ajustando as propostas iniciais para se alinharem melhor com as posições reais dos objetos na imagem.

Loss

A função de perda do algotimo é uma *multi-task loss* onde resulta na combinação da função de perda de classificação e da regressão de *bounding box*.

Portanto, o objetivo do treino consiste na minimização desta função de perda total, através do ajuste dos pesos do modelo [10]. Esse ajuste tem como propósito capacitar o modelo a realizar de forma precisa a classificação das regiões propostas e a predição exata das coordenadas das *bounding boxes*.

Limitações

Embora seja possível executar aplicações em tempo real com alta precisão, desde que os objetos não se movam em altas velocidades, a abordagem utilizada no R-CNN, em particular no Faster R-CNN, apresenta limitações quando se trata de lidar com uma ampla variedade de cenários em tempo

real. Isso ocorre devido à natureza de duas fases desse modelo, que envolve a geração de propostas de regiões seguida da classificação dessas regiões propostas. Essa abordagem de duas fases resulta num aumento significativo no custo computacional, o que, por sua vez, leva a tempos de inferência mais longos.

Portanto, de seguida, serão apresentados modelos de uma única fase que não incluem a etapa de geração de propostas de regiões. Embora esses modelos possam sacrificar um pouco a precisão, oferecem em contrapartida uma redução significativa no tempo de execução, tornando-os mais adequados para cenários em tempo real.

3.4.3 YOLO

O modelo YOLO (*You Only Look Once*) [19] representa uma inovação significativa no campo da deteção de objetos em imagens e vídeos. A sua abordagem [10] é caracterizada pela utilização de uma rede neuronal *end-to-end* que realiza previsões dentro de um número limitado de *bounding boxes* (caixas delimitadoras) e das probabilidades associadas às classes dos objetos detetados. Esta metodologia unificada simplifica consideravelmente o processo de deteção de objetos, resultando numa melhoria notável tanto em termos de velocidade.

Ao contrário do Faster R-CNN, o YOLO é um modelo de deteção de objetos de uma única fase, que não requer a geração prévia de propostas de regiões. Em vez disso, adota uma abordagem de localização direta [12], prevendo as *bounding boxes* diretamente em relação a um número fixo destas. Com apenas um *forward pass* pela rede, o YOLO é capaz de identificar objetos de interesse, eliminando a necessidade de múltiplas passagens na imagem. Essa característica é refletida no nome "*You Only Look Once*", que descreve a eficiência do modelo.

O principal objetivo do YOLO é fornecer inferências de alta velocidade, superando o Faster R-CNN em termos de tempo de processamento, mesmo que isso envolva um ligeiro compromisso na precisão.

O modelo YOLO utiliza uma única CNN para produzir todas as previsões. O primeiro passo no algoritmo inicializa-se pela divisão da imagem original em células de uma grelha SxS de formato uniforme [10]. Cada célula na grelha é responsável por localizar e prever a classe do objeto, juntamente

com o valor de probabilidade.

A notação Y = [Pc, Bx, By, Bh, Bw, C1, ..., CK] corresponde à representação de uma deteção [11] numa das células da grelha. Cada componente possui uma interpretação específica:

- **Pc**: Reflete a probabilidade/confiança da presença de um objeto na célula. Demonstra o grau de certeza do modelo em relação à existência de um objeto na célula.
- **Bx, By**: Indicam as coordenadas x e y do centro da *bounding box* em relação às dimensões da célula. Estas coordenadas definem a posição do objeto dentro da célula. De notar as dimensões de cada caixa é 1 por 1.
- **Bh, Bw**: Representam a altura e largura da caixa delimitadora, tendo em consideração as dimensões da célula.
- **C**: Corresponde às probabilidades associadas às classes possíveis para o objeto na célula. Por exemplo, se o modelo estiver a realizar uma tarefa de deteção de objetos com duas classes (por exemplo, pessoa e cão), C1 representará a probabilidade de ser uma pessoa, enquanto C2 representará a probabilidade de ser um cão.

Na figura 3.11, a primeira célula resultará num vetor onde o valor de Pc é igual a 0, e os valores subsequentes são marcados como "não relevantes". Isso indica que essa célula não contém nenhum objeto e, portanto, a sua saída será desconsiderada.

Na célula localizada na ultima linha, onde o vetor é apresentado, a probabilidade de conter um objeto é 1. Os quatro atributos subsequentes definem as características da caixa delimitadora associada à célula. Neste caso, a célula possui uma *bounding box* com largura e altura iguais a 1. Vale realçar que os valores de Bw e Bh podem exceder o valor de 1, indicando que as dimensões da *bounding box* ultrapassam as dimensões da célula. Por exemplo, se Bw=2, isso significa que a *bounding box* que descreve o objeto tem o dobro da largura da célula.

Por fim, são definidas as probabilidades associadas às classes presentes na célula. No exemplo dado, C1=0 indica que não há uma pessoa na célula, enquanto o valor C2=1 indica que há a presença de um cão.

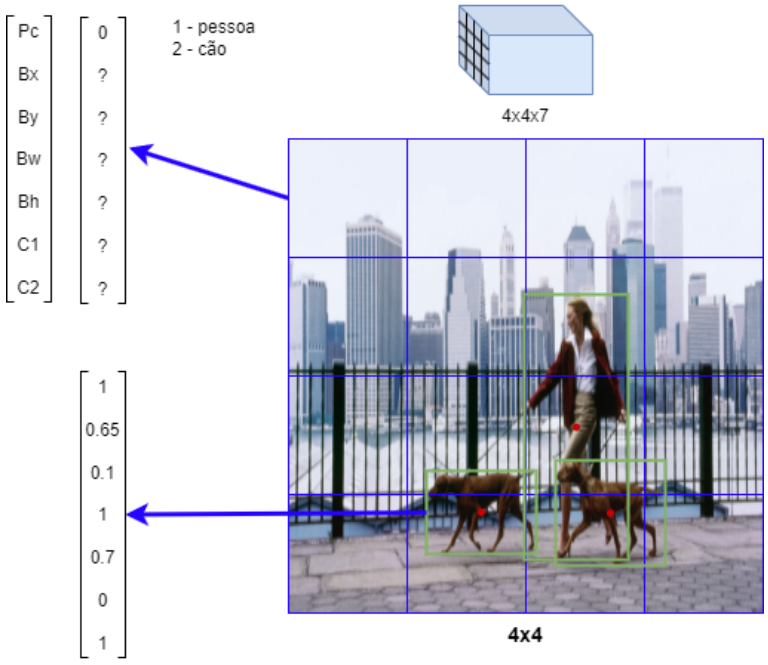


Figura 3.11: Exemplo da definição dos vetores para cada célula

É importante mencionar que o valor de S (tamanho da grelha) pode ser definido e variável para acomodar objetos de diferentes tamanhos na imagem de entrada. Quando S=4, significa que a imagem de entrada é dividida numa grelha 4x4 [10], onde cada célula tem associado um vetor de 7 atributos. Essa flexibilidade em relação a S permite que o modelo seja adaptado para objetos de diversos tamanhos.

Caixas de âncoras

Uma restrição da técnica anterior surge quando múltiplos objetos têm centros na mesma célula, onde só é possível representar uma classe. A introdução das "caixas de âncoras" (*anchors*) resolveu essa limitação. Com a concatenação de dois vetores, é possível representar duas classes na mesma célula, quando o número de âncoras é definido como 2. A escolha do número de âncoras deve ser ponderada em relação ao tamanho das imagens e dos objetos. A saída Y é definida como S x S x número de âncoras x (5 + número de classes), permitindo lidar com múltiplos centros de objetos na mesma célula e melhorar a deteção de objetos.

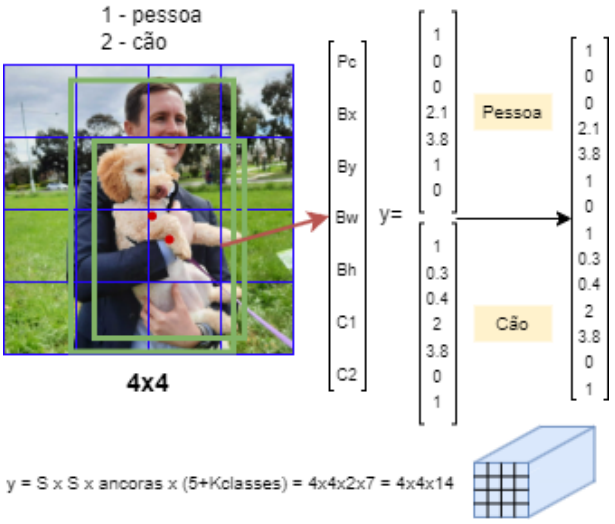


Figura 3.12: Vetor com duas âncoras

Escalas

Uma das melhorias propostas nas versões posteriores do YOLO [10] é a capacidade de realizar deteções em diferentes escalas por célula. Isso é representado na figura 3.13, onde a mesma imagem é processada usando grelhas de três escalas diferentes: 4x4, 8x8 e 16x16. Totalizando ((4x4) + (8x8) + (16x16) x 2) = 672 caixas previstas por imagem. Cada escala de grelha é projetada com um número de âncoras igual a 2.

Esta melhoria oferece uma maior sensibilidade a objetos de diferentes tamanhos. Por exemplo, a grelha 4x4 é responsável pela deteção de objetos maiores, a grelha 8x8 por objetos de tamanho médio e a grelha 16x16 para objetos menores. Embora essa abordagem ofereça uma melhoria significativa em termos de deteção em várias escalas, é necessário ter cuidado ao definir o uso de várias grelhas, os valores de divisão (S) e o número de âncoras [10]. Todos esses hiperparâmetros afetam a complexidade do modelo e o custo computacional, pois cada célula precisa processar informações para deteção. Assim, a seleção de quantas e quais divisões de grade utilizar depende das necessidades específicas da aplicação e dos recursos disponíveis para inferência.

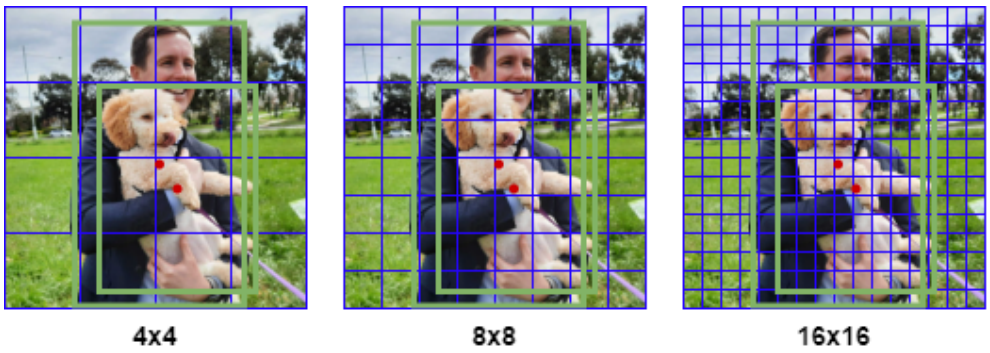


Figura 3.13: Deteções em diferentes escalas

NMS

Após gerar várias caixas delimitadoras para cada objeto, é essencial filtrar e selecionar a caixa delimitadora mais adequada para cada objeto na imagem, removendo caixas redundantes e mantendo apenas uma por objeto identificado.

O processo de filtragem é composto por duas etapas principais [11]: a primeira etapa envolve a eliminação de caixas delimitadoras com baixa probabilidade/confiança, geralmente abaixo de um limite predefinido (por exemplo, 0,4); a segunda etapa utiliza um método chamado *Non-Maximum Suppression* (NMS), que é aplicado a cada classe prevista individualmente. O NMS funciona calculando o IoU entre todas as caixas delimitadoras da mesma classe. O IoU é uma medida que varia de 0 a 1, onde 0 significa nenhuma sobreposição e 1 significa sobreposição perfeita.

Para cada par de caixas delimitadoras que se sobrepõem, o NMS compara o valor do IoU com um limite predefinido (por exemplo, 0,65). Se o valor do IoU for maior que o limite, apenas a caixa delimitadora com a maior probabilidade é mantida, enquanto a outra é descartada. Isso garante que apenas a caixa delimitadora mais confiável seja mantida quando há uma sobreposição significativa. Se o valor do IoU entre as caixas não exceder o limite, ambas as caixas delimitadoras são mantidas no resultado final, pois isso sugere que elas podem delimitar objetos diferentes da mesma classe na imagem. O uso do NMS é importante porque evita a redundância de caixas delimitadoras e garante que apenas as deteções mais confiáveis sejam consideradas. Isso é particularmente útil em cenários de deteção de objetos, onde podem ocorrer

sobreposições entre objetos ou múltiplas deteções da mesma classe na mesma imagem.

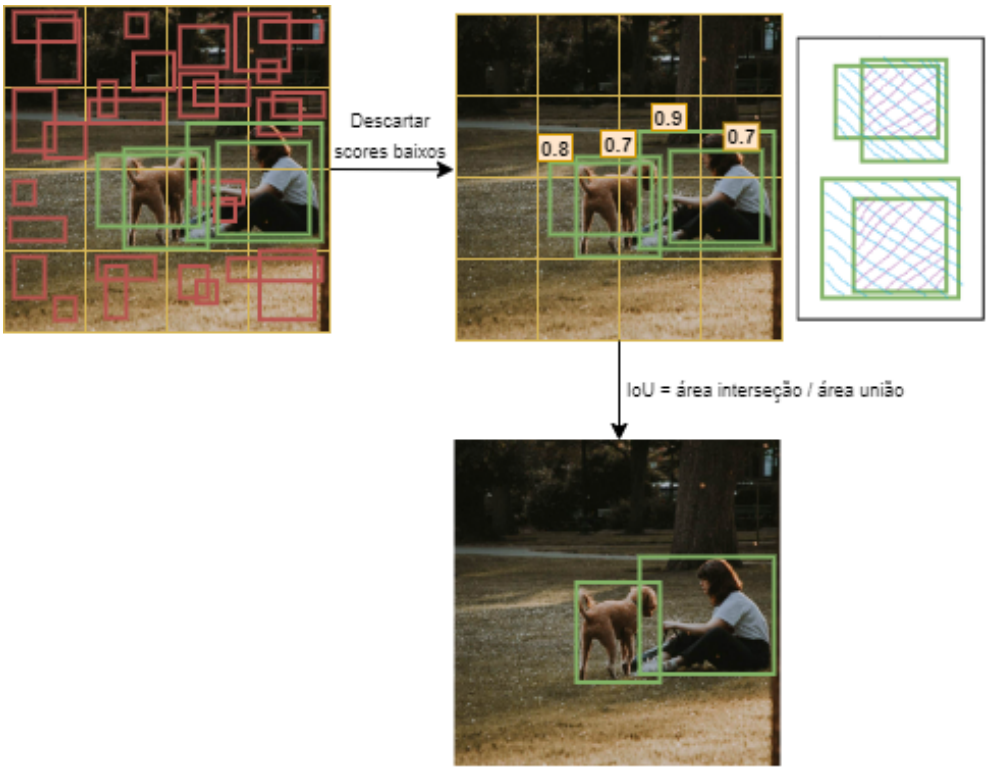


Figura 3.14: Processo de NMS

Vantagens e Desvantagens

O YOLO é notável pela sua eficiência temporal e deteção de objetos em várias escalas por célula. Oferece inferências em tempo real e uma abordagem unificada para *bounding boxes* e probabilidades de classe. No entanto, a sua precisão é inferior ao Faster R-CNN para objetos pequenos e requer configurações detalhadas de hiperparâmetros, como tamanho de grelha e número de âncoras. Apesar disso, o YOLO é uma escolha poderosa para aplicações de deteção rápida, desde que as suas limitações sejam consideradas.

3.4.4 SSD

O *Single Shot Detector* (SSD), assim como o YOLO, [14] é um sistema de deteção de objetos de uma única fase, que não requer um gerador de regiões,

como no caso do Faster R-CNN. Essa característica permite que todo o processo de deteção seja realizado numa única rede, o que lhe confere o nome "*Single Shot*". O SSD utiliza mapas de características de diferentes escalas para acomodar objetos de vários tamanhos, tornando-o altamente flexível.

O SSD é composto por três partes fundamentais: a primeira fase lida com a extração dos mapas de características a partir da imagem de entrada, a segunda, denominada "*detection head*", consiste nas camadas responsáveis pelas previsões do modelo, e, por fim, uma fase de pós-processamento para refinar os resultados e gerar as deteções finais. Essa abordagem unificada torna o SSD eficiente e eficaz na deteção de objetos em imagens e vídeos.

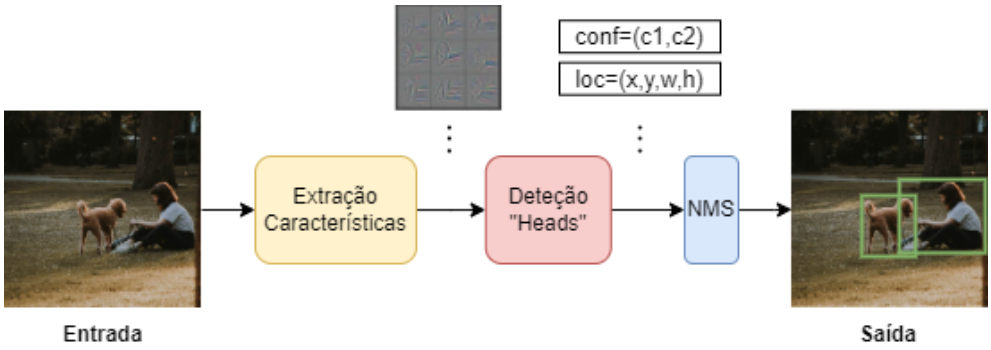


Figura 3.15: Arquitetura SSD de alto nível

Extração de características

Nesta primeira fase do SSD, a imagem é utilizada como entrada para a arquitetura da rede VGG16 [10]. Trata-se de uma CNN conhecida por uma arquitetura composta por blocos repetitivos (5), que diminuem a resolução espacial, mas aumentam a quantidade de características extraídas. Cada bloco consiste em duas ou três camadas de convolução seguidas por uma camada de *max-pooling*. A escolha do VGG16 para esta aplicação deve-se ao seu desempenho sólido na extração de características relevantes em comparação com outros modelos disponíveis na época (2016). É importante mencionar que, para os propósitos do SSD, a parte superior do VGG16, responsável pela classificação, é removida, mantendo apenas as camadas de convolução para a extração de mapas de características da imagem.

Com o objetivo de realizar deteções em diferentes escalas, foram incorporadas camadas de convolução adicionais, formando uma pirâmide de mapas de características que diminuem progressivamente em escala. As saídas de cada bloco da pirâmide servem como entrada para a fase seguinte, gerando as previsões de objetos em várias escalas. Portanto, considerando uma imagem de entrada de 300x300 píxeis, são gerados 6 mapas de características [16] com as seguintes dimensões: 38x38, 19x19, 10x10, 5x5, 3x3 e 1x1.

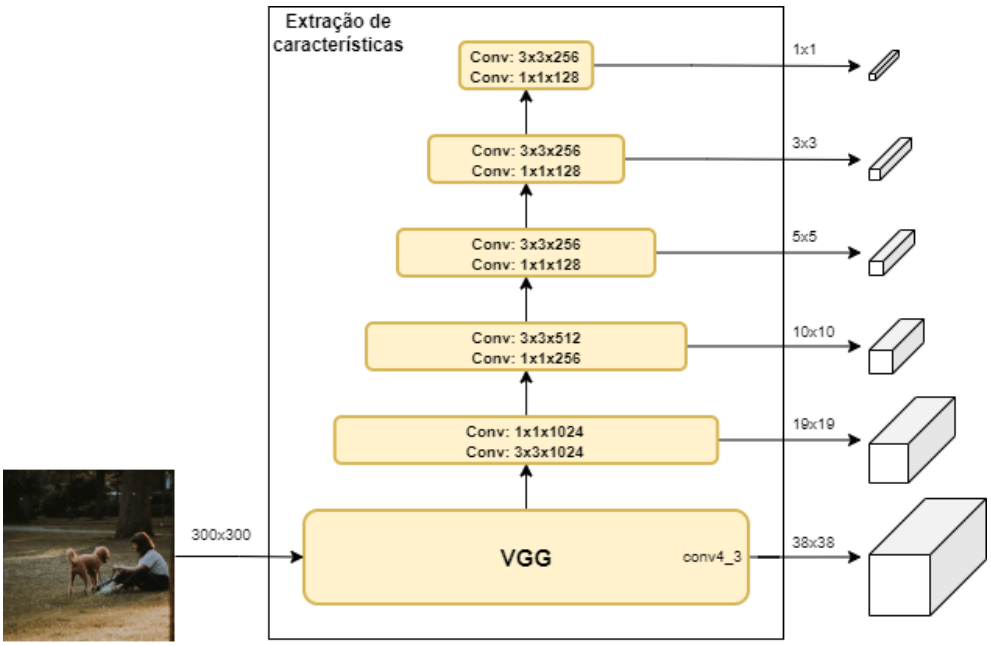


Figura 3.16: Arquitetura do módulo de extração de características

Camadas de deteção

Após a obtenção da pirâmide composta por seis mapas de características na fase anterior, nesta etapa, esses mapas são convertidos em previsões de localização [10]. Para essa finalidade, existem seis "cabeças de deteção" conectadas aos seis mapas correspondentes. Cada "cabeça de deteção" consiste num par de camadas: a primeira camada tem a responsabilidade de gerar a pontuação de confiança (*confidence scores*) para cada classe em relação a cada caixa delimitadora possível. A segunda camada é encarregada de ajustar as âncoras de modo a otimizar a precisão da localização das caixas delimitadoras.

- **Camada confiança:** Considerando o exemplo ilustrado na figura 3.17, na segunda "cabeça de deteção" (3x3), a saída dessa camada é definida como 3 x 3 x (nA x nC), onde "nC" representa o número de classes e "nA" o número de âncoras. Em outras palavras, para cada célula na matriz 3x3, haverá um conjunto de valores de confiança correspondentes a cada combinação de classe e âncora. Assumindo que nC=3 e nA=2, o tensor de saída terá dimensões de 3x3x6, totalizando 54 valores de confiança. É relevante destacar que, para cada âncora, são apresentados valores de confiança para cada classe (por exemplo, *background*: 0.1, c1: 0.7 e c2: 0.2). No exemplo fornecido, a pontuação para a classe c1 é superior à do *background*, o que resulta na identificação positiva dessa caixa delimitadora. Caso a confiança em *background* fosse maior do que as demais, a caixa seria classificada como negativa.

- **Camada localização:** No *Single Shot Detector* (SSD), as âncoras convencionais são denominadas de "caixas padrão" ou "*default boxes*". Essas caixas representam um conjunto predefinido de caixas delimitadoras com diferentes tamanhos, de maneira semelhante ao YOLO e ao Faster R-CNN. O detetor, portanto, sobrepõe uma grelha de caixas padrão ao redor dos mapas de características na tentativa de identificar objetos contidos nessas caixas. Portanto, a saída da camada de localização consiste nos parâmetros relativos aos deslocamentos. Esses deslocamentos são posteriormente utilizados nas âncoras para prever uma caixa delimitadora. A razão pela qual essa camada não retorna diretamente as coordenadas absolutas de uma caixa, mas sim os deslocamentos que resultarão numa caixa, está relacionada ao facto de que as camadas de convolução teriam que produzir valores de saída diferentes para a mesma instância de objeto em diferentes locais dentro da imagem de entrada, isso não é possível devido à partilha de pesos.

  Nesta etapa, um ajuste é aplicado a uma caixa padrão específica para melhorar a precisão da localização. Isso gera informações de localização para cada âncora associada a cada célula, representada por quatro valores que descrevem a posição e o tamanho em relação à célula da grelha. Por exemplo, considerando a segunda "cabeça de deteção" (3x3) ilustrada na figura 3.17 com "nA" igual a 2, cada célula gera duas previsões de localização (x, y, largura e altura). Esse processo é repetido para as

demais "cabeças", permitindo a deteção de objetos em várias escalas. Os mapas de características menores visam identificar objetos maiores, enquanto os mapas maiores concentram-se na deteção de objetos menores.

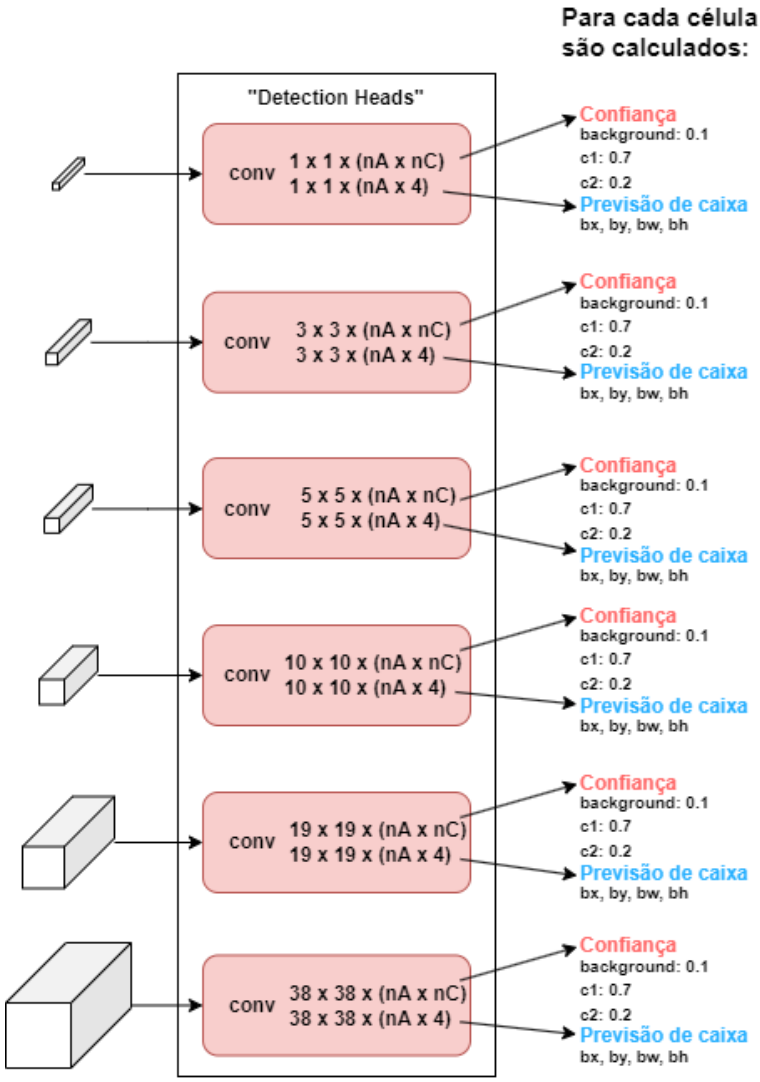


Figura 3.17: Camadas de previsão

Escalas

O SSD é especializado na deteção de objetos de diferentes tamanhos, escalas e formas, alcançado por meio de uma coleção de mapas de características

com dimensões variadas. Isso é exemplificado na figura 3.18, onde a rede redimensiona os mapas de características para dimensões de 8x8 e 4x4 ao identificar dois cães da mesma classe, mas de tamanhos diferentes na imagem.

Ao aplicar caixas padrão a mapas de características de 8x8 e 4x4 [10], estas são redimensionadas para escalas diferentes em relação à imagem original. O mapa de características de 8x8 com mais células oferece uma granularidade mais elevada, ideal para identificar objetos menores, enquanto o mapa de 4x4 com menos células é mais eficaz na deteção de objetos maiores devido à sua granularidade reduzida.

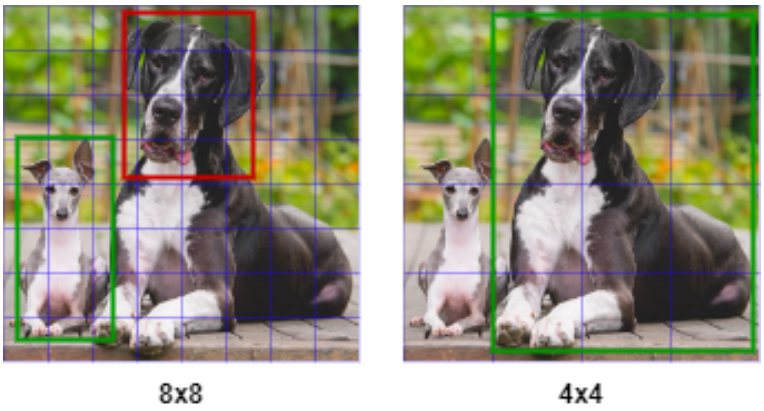


Figura 3.18: Diferentes resoluções de mapas de características

O exemplo na figura 3.18 é meramente ilustrativo, visando facilitar a compreensão e visualização. Os mapas de características, embora abstratos para nossa perceção, preservam as características fundamentais da imagem.

NMS

A terceira e última fase do SSD é dedicada à filtragem e remoção das deteções realizadas, com o objetivo de reter apenas aquelas que melhor descrevem a localização dos objetos presentes na imagem. A importância dessa fase reside no fato de que a fase anterior prevê múltiplas *bounding boxes* que são repetidas e redundantes. Assim como no YOLO, a abordagem utiliza o NMS, descartando inicialmente as caixas delimitadoras com baixa confiança. Em seguida, o processo de IoU é aplicado entre as caixas restantes, descartando aquelas com sobreposição significativa e mantendo as deteções

mais confiáveis. Se não houver sobreposição suficiente, ambas as caixas são mantidas como deteções finais dos objetos.

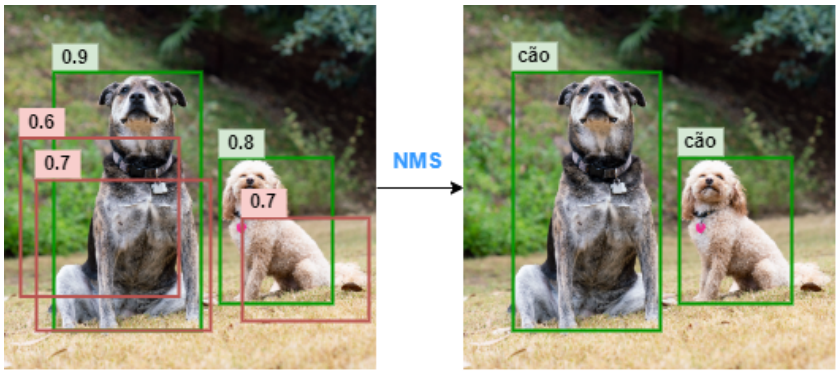


Figura 3.19: Fase NMS

Treino

Durante o treino, é necessário identificar quais "caixas padrão" que correspondem a deteções de qualidade, permitindo que o modelo se adapte e alcance um nível adequado de aprendizagem para a deteção de objetos durante a inferência. Isso é realizado por meio da comparação com as "caixas verdadeiras" (*ground truth*) que representam as localizações reais dos objetos na imagem.

A determinação [16] da precisão de uma "caixa padrão" como representação fiel de um objeto baseia-se na métrica de sobreposição de caixas (IoU). Na sua essência, as "caixas padrão"são minuciosamente comparadas com todas as "caixas verdadeiras" presentes na imagem. Aquelas "caixas padrão" que exibem um valor de IoU superior a 0.5 em relação a alguma "caixa verdadeira" são consideradas deteções positivas. Por outro lado, as "caixas padrão" com IoU menor ou igual a 0.5 são categorizadas como deteções negativas, sugerindo que retratam o plano de fundo da imagem.

Para avaliar o desempenho das deteções de forma rigorosa [14], torna-se essencial estabelecer uma função de perda adequada. Semelhante a outros algoritmos de deteção, o SSD implementa uma função de perda composta, conhecida como *multi loss*. A função de perda total é obtida através da combinação ponderada das perdas de confiança e de localização.

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = log(\frac{g_j^w}{d_i^w}) \qquad \hat{g}_j^h = log(\frac{g_j^h}{d_i^h})$$

A função de perda de localização [16] realiza o cálculo da perda de localização, efetuando uma comparação entre as previsões das coordenadas das caixas delimitadoras (l) e as coordenadas verdadeiras (g). Nesse processo, é aplicada a função *Smooth L1* para atenuar o erro. A soma ocorre exclusivamente para as caixas verdadeiras positivas (Pos), ou seja, aquelas que verdadeiramente contêm objetos. Os termos $\hat{g}_j^{cx}, \hat{g}_j^{cy}, \hat{g}_j^w, \hat{g}_j^h$ representam as coordenadas normalizadas do centro e as dimensões da caixa delimitadora do objeto, calculadas em relação ao tamanho da caixa padrão ($d_i^{cx}, d_i^{cy}, d_i^w, d_i^h$). Permite que o modelo se aproxime das deteções previstas em relação às caixas verdadeiras.

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^p log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \text{ onde } \hat{c}_i^p = \frac{exp(c_i^p)}{\sum_p exp(c_i^p)}$$

A função de perda de confiança [16] avalia o quão precisa é a previsão do modelo em relação às probabilidades das diferentes classes. Esta função de perda é composta por dois termos: o primeiro é aplicado às classes positivas (que representam objetos), e o segundo é para as classes negativas (fundo da imagem). Para calcular esta perda, é utilizada uma operação softmax sobre as confianças de várias classes (c). Ao minimizar esta função, as previsões de confiança aproximam-se das classes corretas para cada exemplo.

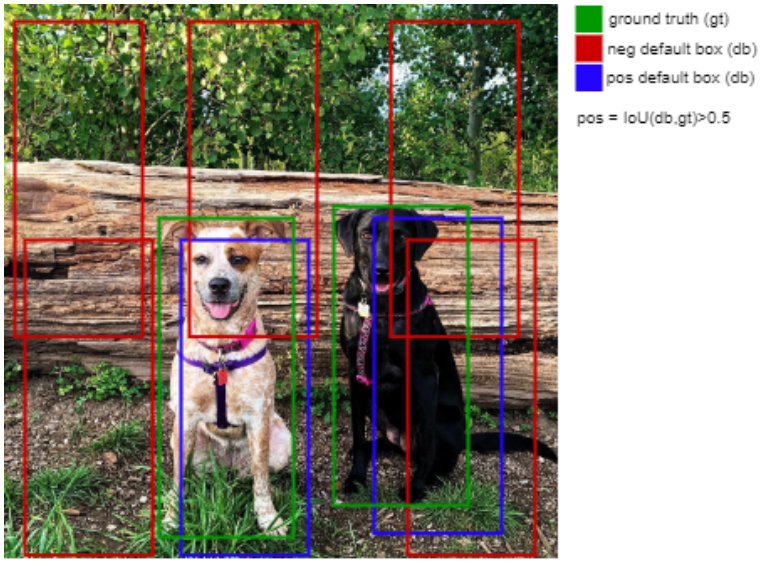


Figura 3.20: Definição de caixas positivas e negativas

Arquitetura

Anteriormente, foi descrito o funcionamento dos módulos individuais do SSD, é agora discutido como a arquitetura do modelo como um todo é estruturada. [16] Na figura 3.21 ilustra que, para cada classe, são geradas 8732 deteções. Para uma imagem de entrada de 300x300 píxeis, resultará em diferentes tamanhos de mapas de características: 38x38, 19x19, 10x10, 5x5, 3x3 e 1x1. É importante observar que cada mapa de características tem um número predefinido de *default boxes*, que pode ser 4 ou 6, dependendo do mapa. Portanto, o total de deteções por classe é calculado como a soma das deteções em cada mapa de características, ou seja, 38x38x4 + 19x19x6 + 10x10x6 + 5x5x6 + 3x3x4 + 1x1x4. A escolha dos valores predefinidos para o número de *default boxes* em cada mapa foi resultado de extensos testes e otimizações, equilibrando o custo computacional com o desempenho do modelo. Devido à grande quantidade de *bounding boxes* geradas, é essencial aplicar o processo de NMS para reduzir o número de deteções na saída final do modelo.

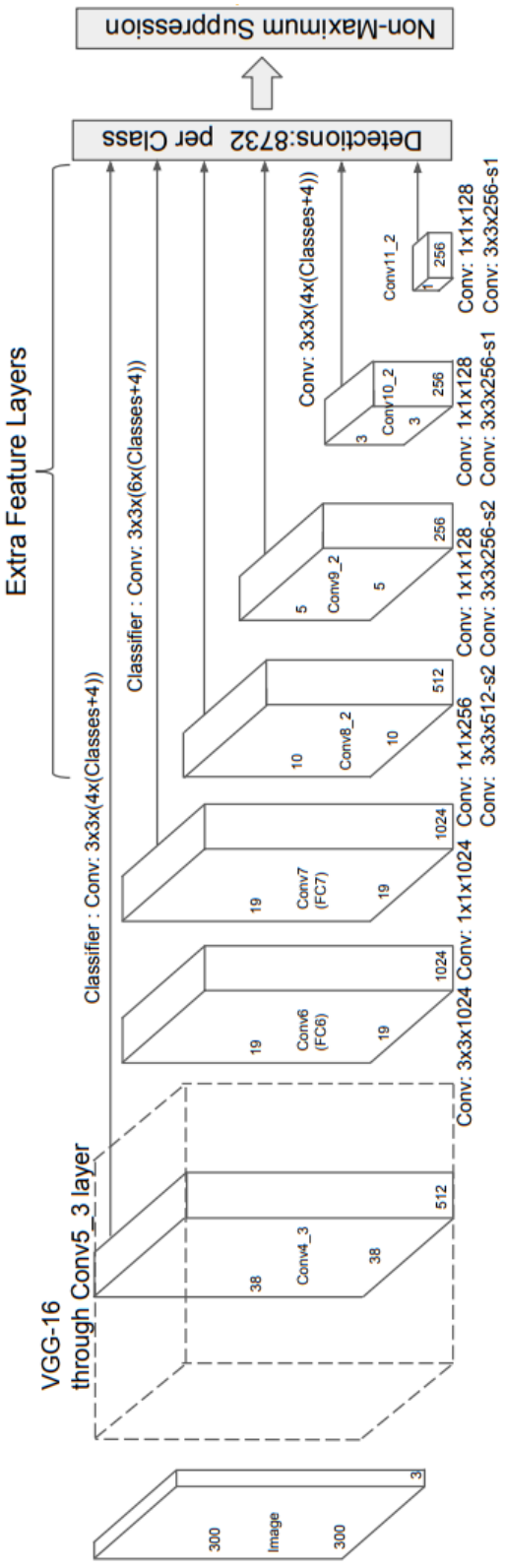


Figura 3.21: Arquitetura original do SSD [16]

Capítulo 4

Implementação

O propósito fundamental deste trabalho reside no desenvolvimento de um sistema de deteção automática de embarcações a ser executado num Raspberry Pi, visando a ampliação do conjunto de dados automaticamente, viabilizando o desenvolvimento e o aperfeiçoamento de futuros sistemas de deteção de embarcações. Além disso, busca-se aprimorar a eficiência da gestão portuária e o controle do tráfego marítimo, com o intuito de reduzir a dependência da intervenção humana. Uma característica adicional notável é a capacidade de implementar o sistema em diversos locais com relativa simplicidade.

Nesta secção, será apresentado em detalhe o processo de desenvolvimento passo a passo. Inicialmente, será apresentado o estudo detalhado do conjunto de dados de embarcações adotado, a fim de possibilitar a aprendizagem dos modelos criados. Em seguida, será abordada a implementação do classificador binário, o qual desempenha o papel inicial no sistema, servindo como uma etapa de filtragem para reduzir o custo computacional. Por último, serão delineados os pormenores referentes ao desenvolvimento do detetor de embarcações, bem como à construção do sistema completo.

4.1 Conjunto de dados

Como em todos os problemas de aprendizagem automática, a disponibilidade de um conjunto de dados é de vital importância para permitir que o modelo aprenda e, posteriormente, execute classificações/deteções com base em informações de entrada não vistas anteriormente. A qualidade e a quantidade dos dados desempenham um papel crucial no desempenho do modelo.

É amplamente reconhecido que a diversidade e a amplitude dos dados contribuem significativamente para o potencial máximo de uma rede neuronal. Isso diferencia a verdadeira aprendizagem do mero processo de memorização. Alguns dos motivos fundamentais para a importância dos conjuntos de dados incluem:

- **Representatividade:** Os dados devem representar fielmente o cenário ou problema que o modelo enfrentará na aplicação do mundo real. Se o conjunto de dados não for representativo, o modelo pode aprender relações incorretas, resultando em classificações também incorretas.
- **Generalização:** A diversificado no conjunto de dados permite que o modelo generalize melhor para novos exemplos não vistos durante o treino. Quanto mais variedade existir nas amostras de dados, mais robusto será o modelo em situações inesperadas.
- **Quantidade:** Um conjunto de dados suficientemente amplo desempenha um papel fundamental na prevenção do *overfitting*, uma questão comum em *machine learning*. O *overfitting* ocorre quando um modelo ajusta-se em excesso aos dados de treino.
- **Sem ruído:** A qualidade dos dados desempenha um papel fundamental no processo de aprendizagem. A correção e a integridade dos dados são de extrema importância, uma vez que dados ruidosos ou incorretos podem resultar em conclusões equivocadas por parte do modelo.

Para desenvolver uma solução para o problema proposto, foi disponibilizado um conjunto de dados devidamente rotulados pelos orientadores. Este conjunto de dados abrange múltiplas embarcações que transitam pela área em frente à Escola Náutica Infante D. Henrique, situada na entrada do rio Tejo. É relevante destacar que essa região apresenta um elevado volume de tráfego marítimo, uma vez que serve como rota para o porto de Lisboa.

Esses dados foram adquiridos utilizando um sistema composto por um Raspberry Pi 4 conectado a uma câmara. Este sistema captura e armazena uma imagem de alta resolução sempre que recebe um sinal do Sistema de Identificação Automática (AIS) emitido pelas embarcações. Além disso, também foram registadas imagens que não continham nenhuma embarcação, a fim de proporcionar um equilíbrio no conjunto de dados.

categoria	**nº imagens**	**subcategorias**	**nº ocorrências**
com barco	2875	fragata marinha portuguesa	129
		graneleiro	625
		lancha de pilotos	84
		navio tanque	175
		NPO	221
		porta contentores	603
		outro	1336
sem barco	3219	-	-

Tabela 4.1: Distribuição das classes no conjunto de dados

No conjunto de dados em questão, existem 2.875 imagens que contêm embarcações (objetos presentes) e 3.219 imagens sem embarcações (imagens de fundo). Dentro do conjunto de imagens que possuem embarcações, estão distribuídas em sete classes distintas, cada uma indicando o tipo de embarcação presente. É importante salientar que a soma das ocorrências nas sete classes não corresponde necessariamente ao número total de 2.875 imagens, uma vez que várias imagens podem conter múltiplas embarcações. Além das imagens fornecidas, também me foi disponibilizado informações sobre todas as imagens do conjunto de dados.

De seguida, são apresentados três exemplos de cada classe, a fim de fornecer uma representação visual das características de cada classe.

Figura 4.1: Imagens da classe 0

Figura 4.2: Imagens da classe 1

Figura 4.3: Imagens da classe 2

Figura 4.4: Imagens da classe 3

Figura 4.5: Imagens da classe 4

Figura 4.6: Imagens da classe 5

Figura 4.7: Imagens da classe 6

Figura 4.8: Imagens da classe 7

Na figura 4.9, são apresentadas três imagens nas quais ocorre a presença de mais de uma embarcação:

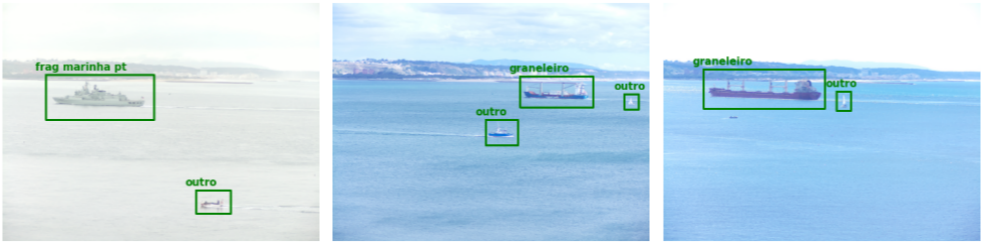


Figura 4.9: Imagens com múltiplos objetos

4.2 Classificação binária

Nesta fase, foi desenvolvida uma rede neuronal de classificação binária com o propósito de identificar a presença ou ausência de embarcações nas imagens. Este primeiro estágio do projeto serve como uma introdução e teste do conjunto de dados nas redes neuronais. Além disso, será utilizado na fase final como a primeira camada de filtragem antes de aplicar a rede de deteção implementada. Em outras palavras, somente se a classificação binária detetar uma embarcação, a rede de deteção realizará a inferência. Isso é feito para reduzir o custo computacional e evitar que o detetor processe os dados continuamente, já que isso teria um escusado custo computacional mais alto.

Para a constituição do conjunto de dados, foram empregadas todas as imagens disponíveis, abrangendo aquelas que retratam a presença de embarcações, bem como as que não apresentam.

O tamanho total do conjunto de dados consiste na soma das imagens com

```
Tamanho total dos dados: 6094
==============================

Tamanho treino:  4875
==============================

Tamanho teste:  1219
==============================
```

Figura 4.10: Distribuição dos dados para classificação binária

barco e das imagens sem barco, totalizando 6094 (2875+3219) amostras. O conjunto de dados de teste representa 20% desse total. Dado que o problema em questão é de baixa complexidade e não requer um *fine tunning* extenso do modelo, a estratégia de implementação foi centrada em utilizar o máximo de dados disponíveis para o treino, visando um desempenho otimizado do modelo. No entanto, é necessário adotar precauções para evitar o fenómeno de *overfitting*.

Devido ao volume e alta resolução das imagens, armazená-las diretamente em memória torna-se impraticável. Para lidar com isso, foi adotada a estratégia de utilizar um gerador proveniente da biblioteca TensorFlow, que permite o pré-processamento e o fornecimento dos dados em lotes, reduzindo significativamente o consumo de memória.

De seguida é necessário aplicar o pré-processamento das imagens, isto é aplicado nas imagens em *batches* usando o gerador anteriormente referido, a ordem de pré-processamento é:

1. **Histograma:** A aplicação de uma equalização de histograma [24] tem como objetivo aprimorar o contraste e realçar detalhes subtis presentes nas imagens. Nesse contexto, utiliza-se o método CLAHE, o qual efetua a equalização de histograma em regiões individuais. Esse procedimento culmina numa maior capacidade do modelo em identificar características significantes.

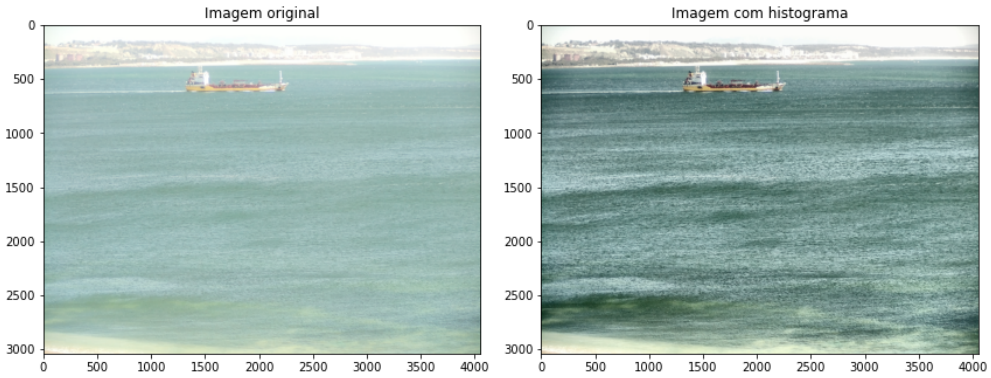


Figura 4.11: Aplicação da equalização

2. **Redimensionamento:** O redimensionamento das imagens para dimensões fixas é uma prática essencial para garantir a uniformidade das dimensões das imagens de entrada e a sua compatibilidade com a arquitetura da CNN utilizada. Neste contexto, optou-se por padronizar o tamanho das imagens para 224x224 píxeis. A escolha desse tamanho específico é motivada pelo fato de que a arquitetura utilizada para a técnica de *transfer learning*, que será abordada em detalhes posteriormente, ter sido treinada com imagens redimensionadas para 224x224 píxeis.

3. **Normalização:** Foi aplicada uma normalização aos valores dos píxeis das imagens, restringindo-os ao intervalo entre -1 e 1. Essa abordagem tem diversas vantagens, incluindo a aceleração da convergência do modelo, a prevenção de oscilações nos valores dos pixeis e a compatibilidade com a arquitetura da CNN que será utilizada.

Com a divisão do conjunto de dados em conjuntos de treino e teste, juntamente com o pré-processamento das imagens, o próximo passo envolve a criação do modelo de aprendizagem. Para alcançar isso, optou-se pela aplicação da técnica de *transfer learning*. Essa abordagem envolve a reutilização do conhecimento adquirido por uma arquitetura complexa previamente treinada num vasto conjunto de dados composto por milhões de imagens, abrangendo diversas classes. Isso resulta na criação de um modelo com melhor desempenho, superando as redes neuronais desenvolvidas e treinadas do zero.

Nesse contexto, o processo envolve a aplicação do conhecimento substancial adquirido por meio de uma rede pré-treinada com a finalidade de abordar um problema específico. A arquitetura escolhida para tal é a MobileNetV2, que se destaca por uma eficiência computacional, mantendo um elevado nível de precisão em comparação com outros modelos (VGG16, ResNet). A seleção dessa arquitetura é particularmente justificada quando se leva em consideração a intenção de realizar inferências num dispositivo com recursos limitados, como o Raspberry Pi 4.

É importante salientar que o MobileNetV2 foi treinado utilizando um dos maiores conjuntos de dados disponíveis, conhecido como "ImageNet" [20], que contém mais de 1 milhão de imagens e abrange mais de mil classes.

A arquitetura foi adaptada ao remover o topo da rede, que originalmente continha as camadas treinadas com o conjunto de dados "ImageNet". Além disso, os parâmetros das camadas da rede foram congelados, impedindo que sejam atualizados durante o treino com os dados das embarcações. Foram introduzidas duas camadas de convolução, seguidas por uma camada de saída com um único neurónio com ativação "sigmoid". Essas modificações foram feitas para reconfigurar a arquitetura, direcionando a saída do modelo para um intervalo entre 0 e 1, com o objetivo de obter a classificação binária desejada. Dessa forma, um valor de decisão de 0.5 foi estabelecido, classificando uma imagem como positiva se o valor de saída for superior ao limite e como negativa se for inferior.

É importante observar que diversas combinações de arquiteturas foram testadas até a obtenção de uma rede final que equilibra desempenho e custo. Na figura 4.12, é exposto o progresso do treino desta rede, a qual foi treinada ao longo de 30 épocas. Os resultados apresentaram os seguintes comportamentos em relação à precisão de treino, teste e à função de perda de treino e teste:

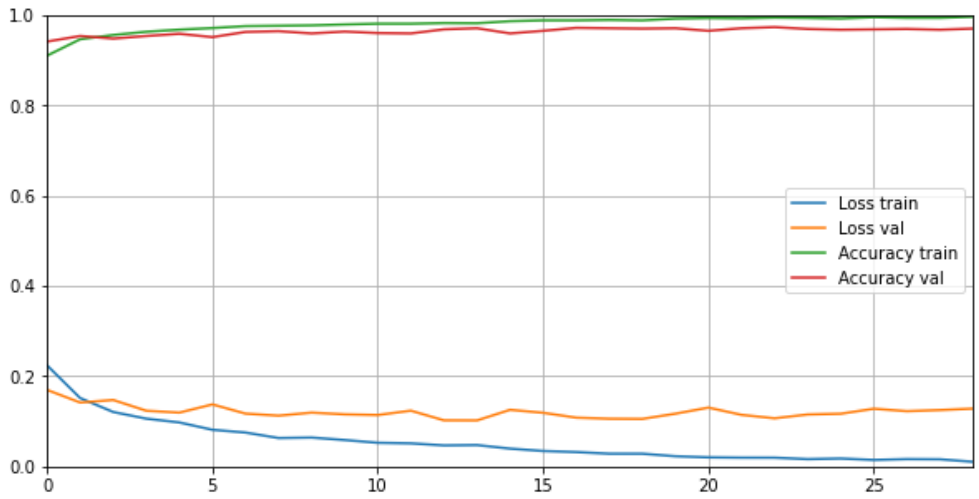


Figura 4.12: Historio de treino da classificação binária

O gráfico referente à figura 4.12 exibe um comportamento típico e esperado durante o treino de uma rede. As precisões no treino e no teste aumentam progressivamente ao longo das épocas, enquanto as perdas, tanto no treino quanto na validação, diminuem. O indicativo mais relevante do desempenho do modelo é observado na função de perda do conjunto de teste, pois minimizá-la é crucial para obter um modelo generalizado e adaptável a dados não previamente vistos.

Analisando a curva de perda no conjunto de teste, observa-se que diminui gradualmente até a época 6, momento em que se estabiliza, com pouca redução adicional. Além disso, começa a surgir uma discrepância entre as curvas de perda no treino e no teste. A curva de treino continua a diminuir, indicando que o modelo está a se adaptar cada vez mais aos dados de treino. No entanto, essa adaptação excessiva aos dados de treino resulta num desempenho inferior nos dados de teste, caracterizando um fenómeno conhecido como *overfitting*.

É importante destacar que, mesmo na época 1, as precisões já são altas, e as perdas são baixas, graças ao processo de *transfer learning*, no qual as camadas de convolução já possuem pesos previamente treinados.

Ao realizar o treino de uma rede utilizando a técnica de *transfer learning*, é comum realizar ajustes posteriores no treino para melhorar ainda mais o desempenho do modelo. Um desses ajustes envolve descongelar algumas das últimas camadas da arquitetura pré-treinada, como a MobileNetV2, permi-

tindo que essas camadas se adaptem ao conjunto de dados específico. Além disso, ao realizar esses ajustes no treino, é aconselhável reduzir a taxa de aprendizagem para evitar o risco de *overfitting*.

No entanto, ao analisar o desempenho atual do modelo e observar as funções de perda, verifica-se que já existe um desempenho bastante elevado e que as funções de perda apresentam algumas tendências significativas de *overfitting* caso o treino continue. Portanto, neste contexto, não é justificável aplicar essa estratégia de ajuste no treino do modelo.

4.3 Deteção de objetos

Esta fase representa um marco crucial no desenvolvimento do projeto, onde a seleção e implementação do modelo de deteção de objetos desempenham um papel fundamental. Com o objetivo de executar inferências num Raspberry Pi4, que possui recursos computacionais limitados, a escolha de um modelo de 2 estágios, como o Faster R-CNN, não se revela viável. Portanto, as opções viáveis restringem-se ao YOLO e ao SSD. Ambas as arquiteturas têm se destacado na resolução de problemas semelhantes de controlo de tráfego, mas optei pelo *Single Shot Detector* (SSD) devido a uma ligeira vantagem de precisão em múltiplas escalas, mesmo que seja um pouco mais lento. Tal fator não representa um problema significativo, pois as embarcações geralmente não se deslocam a altas velocidades, e um valor de FPS moderado/ligeiro é suficiente para um sistema de deteção eficaz.

Outra razão [12] que influenciou essa escolha é a constante evolução e atualização do YOLO, com a versão mais recente, a versão 8, lançada no início de 2023. Embora essas atualizações contribuam para melhorias e resultados mais precisos, ao implementar um sistema a partir do zero, pode tornar-se desafiador acompanhar as mudanças. Por exemplo, na versão 8, não são mais usadas caixas de âncoras, uma característica comum nas versões anteriores. Essas intenções de mudanças podem levar a uma falta de continuidade e dificultar a motivação do estudo e implementação de uma versão específica à medida que novas versões são lançadas.

4.3.1 SSD VOC

No desenvolvimento original do SSD [16], o sistema foi submetido a testes utilizando dois conjuntos de dados: o [21] PASCAL VOC e o [22] COCO. Como parte da fase de experimentação e análise do modelo, antes de utilizar o conjunto de dados específico para deteção de embarcações, escolhi por treinar o modelo inicialmente com o conjunto de dados PASCAL VOC. Essa escolha justifica-se pela intenção de posteriormente realizar uma comparação direta entre o desempenho do sistema treinado com o PASCAL VOC e do mesmo treinado com o conjunto de dados das embarcações.

O conjunto de dados PASCAL VOC é composto por 9.963 imagens, que cobrem 20 classes de objetos distintas, como "cão", "gato", "carro", "avião" e outras. Cada imagem é acompanhada por anotações detalhadas que descrevem a presença e a localização dos objetos. Essas anotações incluem informações sobre as coordenadas das caixas delimitadoras dos objetos, a classe e outros detalhes relevantes.

Carregamento dos dados e pré-processamento

O primeiro passo é extrair e armazenar a informação relativa ao conjunto de dados, tal como referido anteriormente as informações relativas às imagens estão guardados num ficheiro de anotações em xml, foi portanto necessário extrair a informação contida no xml. Na figura 4.13 está demonstrado o resultado da extração.

O conjunto de dados foi dividido de acordo com as recomendações das anotações, com metade dos dados destinada ao treino e a outra metade para o teste. Não foi reservado um conjunto de validação devido à limitação de recursos para experimentações detalhadas com várias combinações de hiperparâmetros durante o treino.

Figura 4.13: Imagens do conjunto de dados VOC, com a informação extraída

Figura 4.14: Distribuição dos dados VOC para SSD

Uma prática comum em *machine learning* é a aplicação de técnicas de *augmentation* nos dados. Em geral, essa abordagem é considerada benéfica, pois contribui para enriquecer o conjunto de dados em termos de diversidade, principalmente quando o número de exemplos é baixo. Em outras palavras, cada exemplo fornecido à rede de treino é submetido a diversas transformações, principalmente de natureza geométrica e de cor. Isso resulta na criação de novas imagens que introduzem uma variabilidade de transformações, simulando assim diferentes situações do mundo real que podem não estar representadas no conjunto de dados original.

A técnica de *augmentation* desempenha um papel importante na prevenção do *overfitting* do modelo, uma vez que enriquece os dados de treino, permitindo que o modelo generalize de maneira mais eficaz e, consequentemente, melhore a precisão. É fundamental realçar que a aplicação de *augmentation* aos dados deve ser feita com discernimento, dependendo do problema em questão. Não faz sentido gerar exemplos que descrevam situações irreais, como, por exemplo, aplicar uma rotação de 180^{o} numa imagem de carros. No entanto, em outros contextos, como imagens de flores, essa abordagem pode ser justificável.

Tendo isso em conta, [17] os criadores do SSD incorporaram uma técnica de *augmentation* específica. É importante observar que as transformações a seguir são controladas por um intervalo de valores, a partir do qual uma seleção aleatória é feita. Portanto, a aplicação de *augmentation* na mesma imagem duas vezes nunca resultará nos mesmos resultados.

1. **Distorção fotométrica:** São alterações ou manipulações na intensidade de luz [17], cores e contrastes de uma imagem.

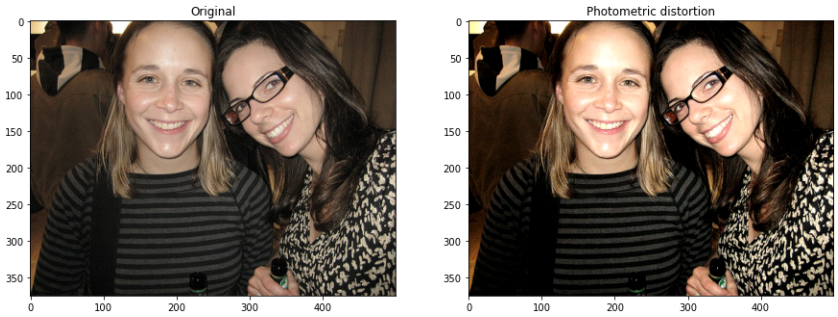


Figura 4.15: Resultado de distorção fotométrica

2. **Expansão:** Estende a área da imagem original preenchendo o espaço extra criado com o valor médio de cor. [17] Simula o efeito de fotografias tiradas a partir de ângulos ou distâncias ligeiramente diferentes.

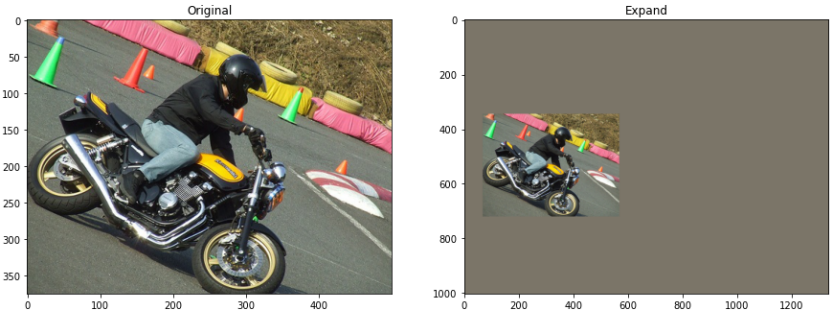


Figura 4.16: Resultado de expansão

3. **Corte:** Recorta uma porção da imagem [17] com a condição de que essa área tenha uma sobreposição (IoU) com pelo menos uma caixa delimitadora de referência (*ground truth*) e que o centroide de pelo menos uma caixa delimitadora de referência esteja contido nesta área. Esta operação faz sentido por remover informação que não é relevante do fundo da imagem.

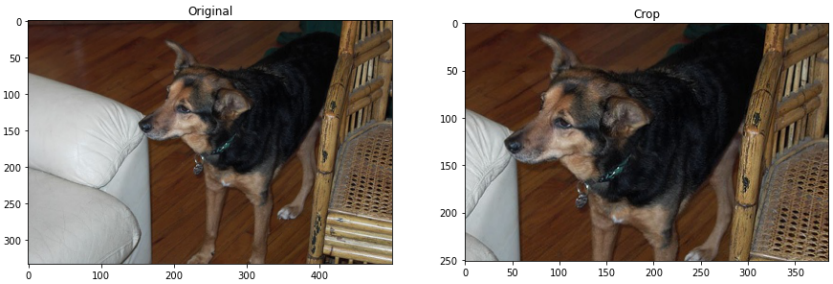


Figura 4.17: Resultado de crop

4. **Espelhamento:** Todos os elementos da imagem sofrem uma inversão horizontal, isto é,[17] o que originalmente se encontrava à esquerda passa a situar-se à direita, e vice-versa. Essa operação tem a finalidade de simular a representação do objeto na orientação horizontal oposta.

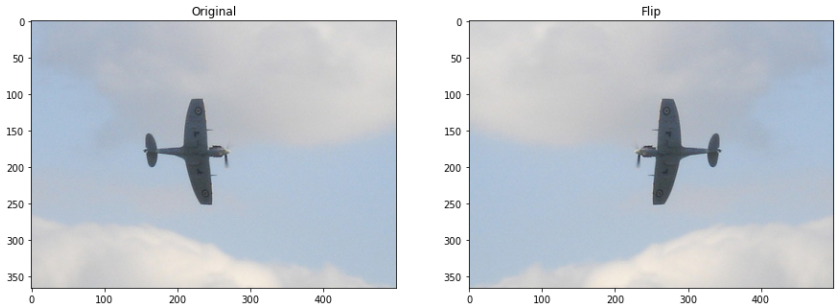


Figura 4.18: Resultado de espelhamento

Integrando todas essas etapas de *augmentation* na ordem apropriada numa imagem com valores de decisão aleatórios, seguida por um redimensionamento fixo, neste caso, para largura e altura de 300, são criadas novas instâncias que o conjunto de dados original não representava. Isso permite que o modelo generalize melhor. É importante observar que a *augmentation* é aplicada apenas aos dados de treino, pois é essencial avaliar o modelo com os dados de teste, mantendo a natureza original e real.

Conforme previamente mencionado, devido à utilização das caixas padrão em mapas de características de diversos tamanhos, o modelo não consegue prever parâmetros absolutos para definir as caixas delimitadoras. Como resultado, o modelo prevê 4 parâmetros de deslocamento em relação às referidas "caixas padrão". É, portanto, imperativo a presença de um codificador que converta as coordenadas das caixas verdadeiras em deslocamentos em relação às "caixas padrão" existentes. Este formato desempenha um papel crucial no cálculo da função de perda durante o treino. No âmbito da implementação do codificador, este também assume a responsabilidade de calcular e selecionar as "caixas padrão" que melhor representam o objeto, determinando a sobreposição IoU entre estas e as caixas delimitadoras verdadeiras para, posteriormente, efetuar o cálculo da perda de localização.

O codificador [18] resume-se então na geração das "caixas padrão"para cada um dos 6 mapas de características gerados pelo módulo de extração de características, com valores de coordenadas absolutas. Em seguida, identifica a "caixa padrão"que apresenta a maior sobreposição com cada caixa delimitadora verdadeira e, por fim, calcula os deslocamentos entre a caixa verdadeira e a caixa padrão correspondente, portanto, esses deslocamentos (codificações) correspondem às *labels* de entrada do SSD.

É necessário agora criar dois geradores, um para o conjunto de treino e outro para o conjunto de teste. Ambos os geradores recebem os respetivos conjuntos de dados (denominados "*filenames*" e "*labels*"). Além disso, também recebem o codificador previamente mencionado e as transformações que serão aplicadas nas imagens. Similarmente ao gerador usado na classificação binária disponível pelo TensorFlow, a função principal desses geradores é produzir lotes de amostras indefinidamente. Neste caso, também desempenham o papel de aplicar as transformações desejadas nas imagens e realizar a codificação dos dados.

É importante observar que, durante o treino, os dados são baralhados para evitar que a ordem dos exemplos influencie o processo de aprendizagem dos pesos da rede neuronal. Isso é particularmente relevante quando existe alguma ordem intrínseca nas classes dos dados. Por outro lado, no conjunto de teste, o baralhamento não é apropriado, pois esse conjunto é usado exclusivamente para avaliar o desempenho do modelo. Nesse contexto, é mais conveniente manter a ordem original dos dados, o que facilita a realização de testes.

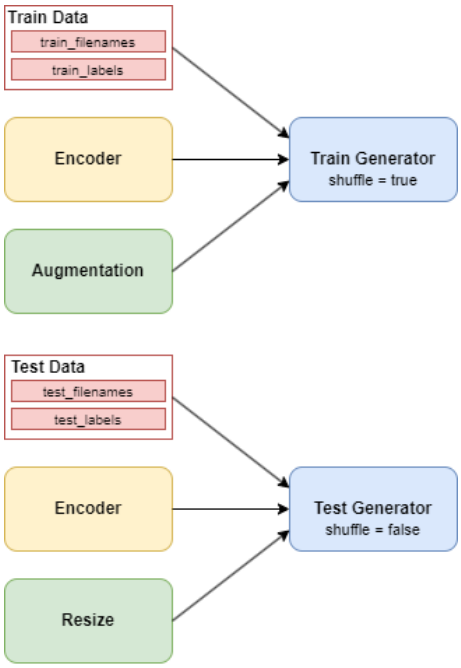


Figura 4.19: Geradores conjuntos de treino e teste

Construção do modelo

Com base no repositório estabelecido por Pierluigi Ferrari [18], procedeu-se ao desenvolvimento e à adaptação do modelo SSD VGG-16 para efeitos de comparação e referência, conforme exposto na teoria e exemplificado na figura 3.21. O modelo recebe os parâmetros [18] que definem as *default boxes*, como o número destas por cada camada de predição, as proporções de *aspect ratio* e as escalas. Além disso, são fornecidas as dimensões da imagem e o número de classes presentes no conjunto de dados. Adicionalmente, foram incorporados na rede base VGG16 os pesos previamente treinados e no conjunto de dados ImageNet. Em outras palavras, foi utilizado a técnica de *transfer learning* com o propósito de agilizar a extração de características, resultando, assim, num modelo de deteção de objetos aprimorado. Finalmente, é necessário estabelecer um otimizador e uma função de perda. Foi escolhido o Adam, uma vez que o SGD resultou em dificuldades de convergência durante o processo de treino. Quanto à função de perda, foi implementada a abordagem de múltiplas tarefa (classificação e localização), referente à perda do SSD, conforme descrito na teoria. O modelo desenvolvido é composto por aproximadamente 26 milhões de parâmetros.

Treino

O modelo implementado foi submetido a uma série de testes, nos quais foram realizados ajustes nos hiperparâmetros. Em seguida, serão expostos os resultados finais provenientes da investigação realizada.

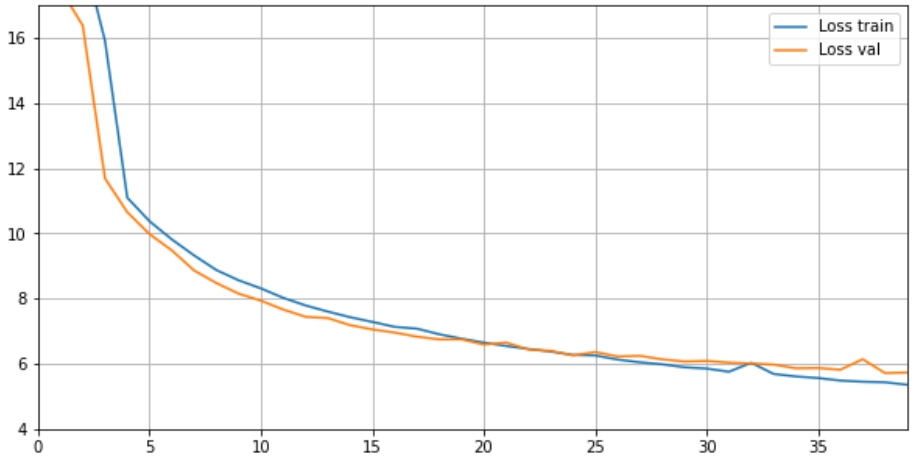


Figura 4.20: Histórico de treino do SSD VOC

Observando o comportamento da função de perda durante o treino (figura 4.20), constata-se o comportamento esperado, no qual tanto a perda de treino quanto a perda de teste diminuem progressivamente à medida que o número de épocas aumenta. A diferença inicial entre o valor da perda de validação, que é inferior à perda de treino, pode possivelmente ser atribuída ao facto de que os exemplos no conjunto de validação podem ser ligeiramente mais fáceis de serem detetados do que os do conjunto de treino.

Importa salientar que não foram identificados sinais de *overfitting*, uma vez que ambas as funções de perda continuam a diminuir sem que ocorra uma grande discrepância entre elas. Além disso, o gráfico indica que ambas as funções de perda continuarão a diminuir com o aumento do número de épocas, o que resultaria num modelo ainda mais aprimorado. No entanto, é relevante mencionar que esta fase representa apenas um teste num conjunto de dados experimental (VOC).

Na figura 4.21, é apresentado o resultado da saída da rede após a aplicação do NMS num exemplo de teste. Nesse exemplo, foram previstas duas caixas delimitadoras, uma correspondente à deteção de um cavalo e a outra à deteção de uma pessoa.

```
Predicted boxes:

   class    conf xmin   ymin   xmax   ymax
[[ 13.     0.82  47.   121.   286.   418.  ]
 [ 15.     0.98 126.    71.   258.   294.  ]]
```

Figura 4.21: Resultados da saída do SSD para um exemplo de entrada

Na figura 4.22, são apresentadas as previsões juntamente com as caixas verdadeiras (indicadas a verde) para o mesmo exemplo. Observa-se que a previsão da caixa correspondente ao cavalo não é perfeita, especialmente em relação à largura. No entanto, a deteção da pessoa é correta. Este exemplo não é considerado de deteção fácil devido à sobreposição entre as duas classes.

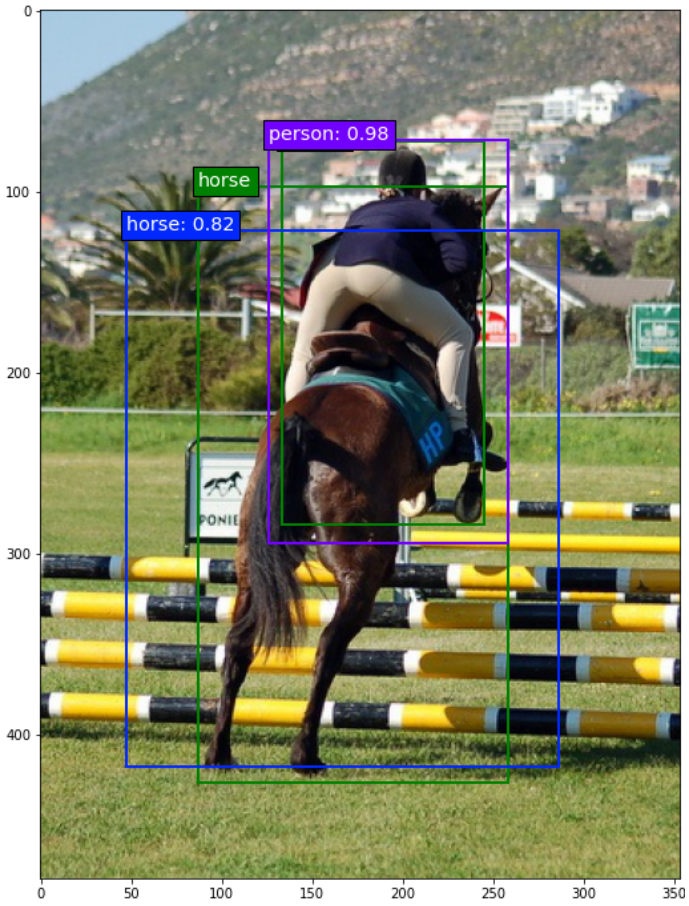


Figura 4.22: Ilustração dos resultados de previsão e das caixas verdadeiras.

4.3.2 SSD Embarcações

Concluindo o bom funcionamento do SSD para o conjunto de dados VOC, é agora necessário treinar o modelo com o conjunto de dados das embarcações sendo este o principal objetivo do trabalho.

Carregamento dos dados

O primeiro procedimento consiste em extrair as informações desse conjunto de dados e formatá-las de acordo com a estrutura necessária para o gerador, semelhante ao processo realizado para o conjunto de dados VOC, de notar que apenas serão utilizados os exemplos que contêm embarcações.

A divisão do conjunto de dados das embarcações foi efetuada em dois subconjuntos distintos: o conjunto de treino, que representa 70% do total de exemplos disponíveis, e o conjunto de teste, que compreende os restantes 30%. Assim como nos casos anteriores, não foi criado um subconjunto de validação, uma vez que ajustar hiperparâmetros neste modelo não é uma abordagem realista.

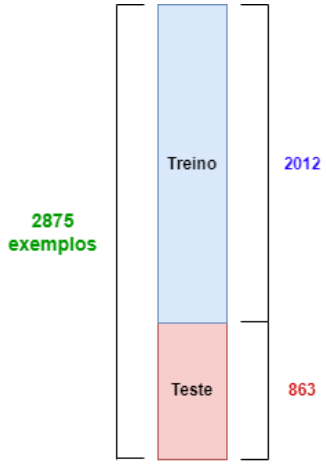


Figura 4.23: Distribuição do conjunto de dados para o conjunto de dados das embarcações

De seguida são apresentados três exemplos da *augmentation* no conjunto de dados das embarcações:

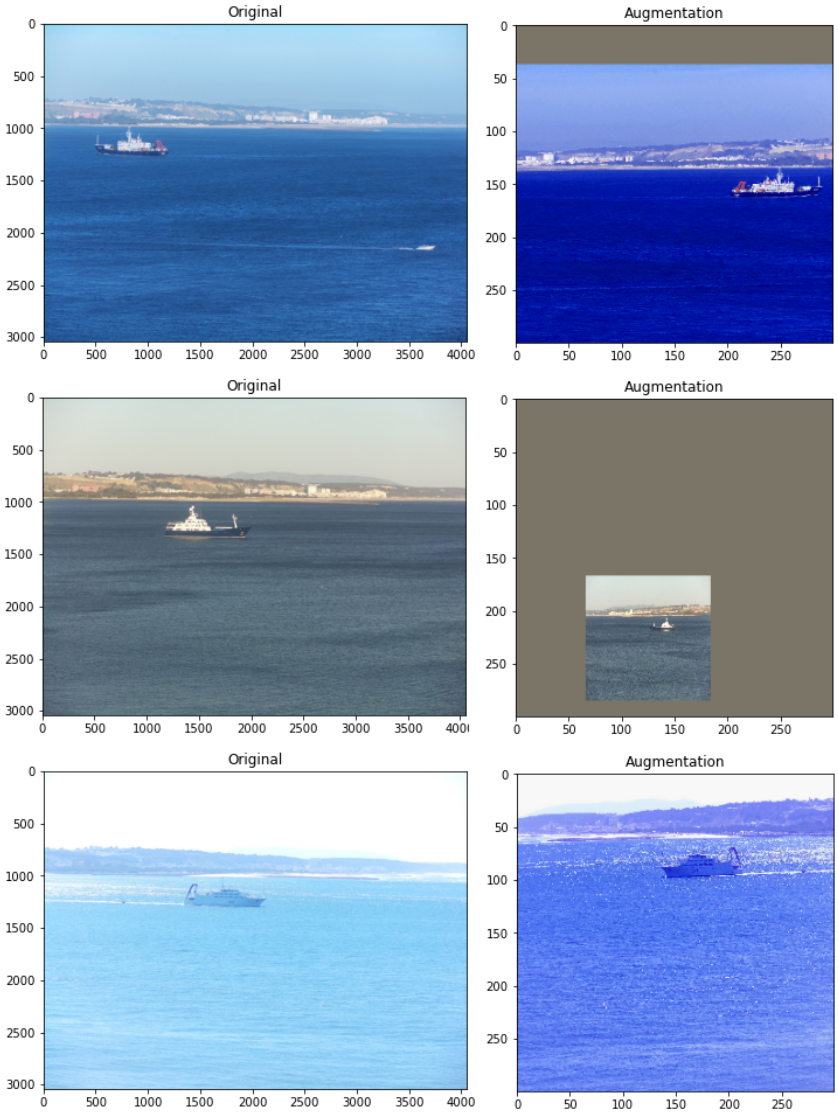


Figura 4.24: Resultados da *augmentation* nas embarcações

Treino

Todas as etapas previamente descritas para o conjunto de dados do VOC foram implementadas de maneira idêntica ao conjunto de dados de embarcações. Durante a primeira época de treino, foi notado um tempo total de aproximadamente 2 horas, o que implicaria um tempo total de treino de 3 dias caso o modelo fosse treinado por 40 épocas. Essa discrepância em relação ao tempo de treino no conjunto VOC, onde cada época levou cerca

de 20 minutos, levanta questões. A diferença de tempo está relacionada com as transformações (*data augmentation*) aplicadas às imagens dos dados, visto que o modelo não foi alterado nesta fase. As imagens neste conjunto de dados são de alta resolução (4056x3040 píxeis), o que significa que as operações de *augmentation* são aplicadas a um grande número de píxeis, tornando o processo de treino consideravelmente mais lento.

A solução adotada foi redimensionar todas as imagens no conjunto de dados, mantendo a proporção original, para um tamanho de 400 píxeis de largura por 300 píxeis de altura em cada imagem. Isso resultou numa redução significativa referente à quantidade de píxeis por imagem, mitigando assim o problema do custo computacional.

Com a redução da resolução das imagens antes de aplicar a técnica de *augmentation*, foi possível reduzir o tempo de treino do modelo para 20 minutos por época. O treino foi conduzido ao longo de 40 épocas.

O comportamento das funções de perda é conforme o esperado, onde os valores de perda de ambos os conjuntos continuam a diminuir progressivamente. Embora haja alguma estagnação nas últimas épocas nos valores de perda de treino e validação, provavelmente não causaria danos em continuar o treino por mais algumas épocas, pelo menos até que a perda de treino alcance ou diminua ligeiramente em relação à perda de validação. No entanto, este modelo não foi submetido a treino adicional por razões que serão explicadas posteriormente.

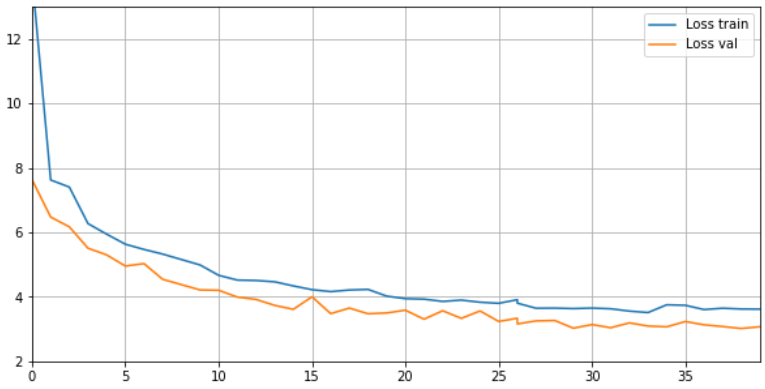


Figura 4.25: Historico de treino do SSD Embarcações

Na figura 4.26 são exibidos os resultados da deteção para dois exemplos dos dados de teste. São apresentadas tanto as caixas delimitadoras reais

quanto as caixas delimitadoras previstas. É evidente uma deteção satisfatória em ambos os casos.

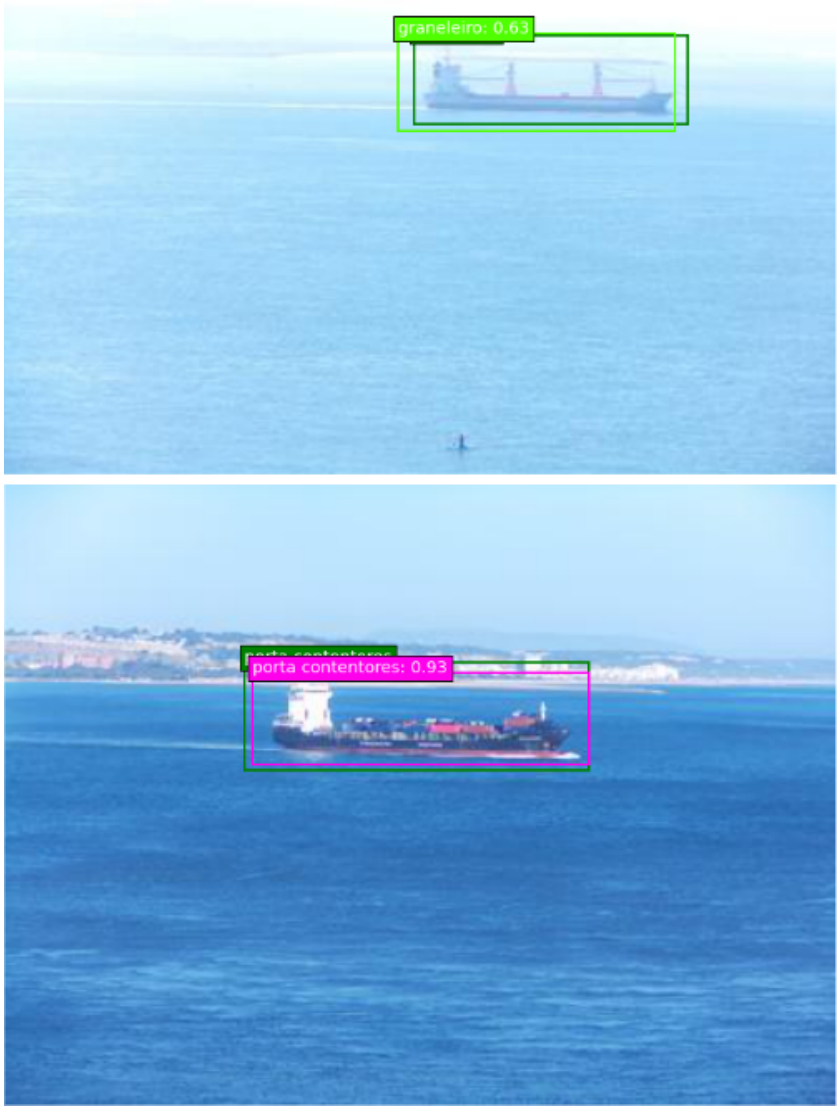


Figura 4.26: Resultados corretos de deteção

A figura 4.27 ilustra exemplos de deteções imprecisas. No primeiro caso, o modelo classificou um objeto como "porta-contentores" em vez de "outro",

devido à semelhança entre os tipos de objetos e à redundância na classe "outro". No segundo caso, apenas dois dos três objetos foram identificados.

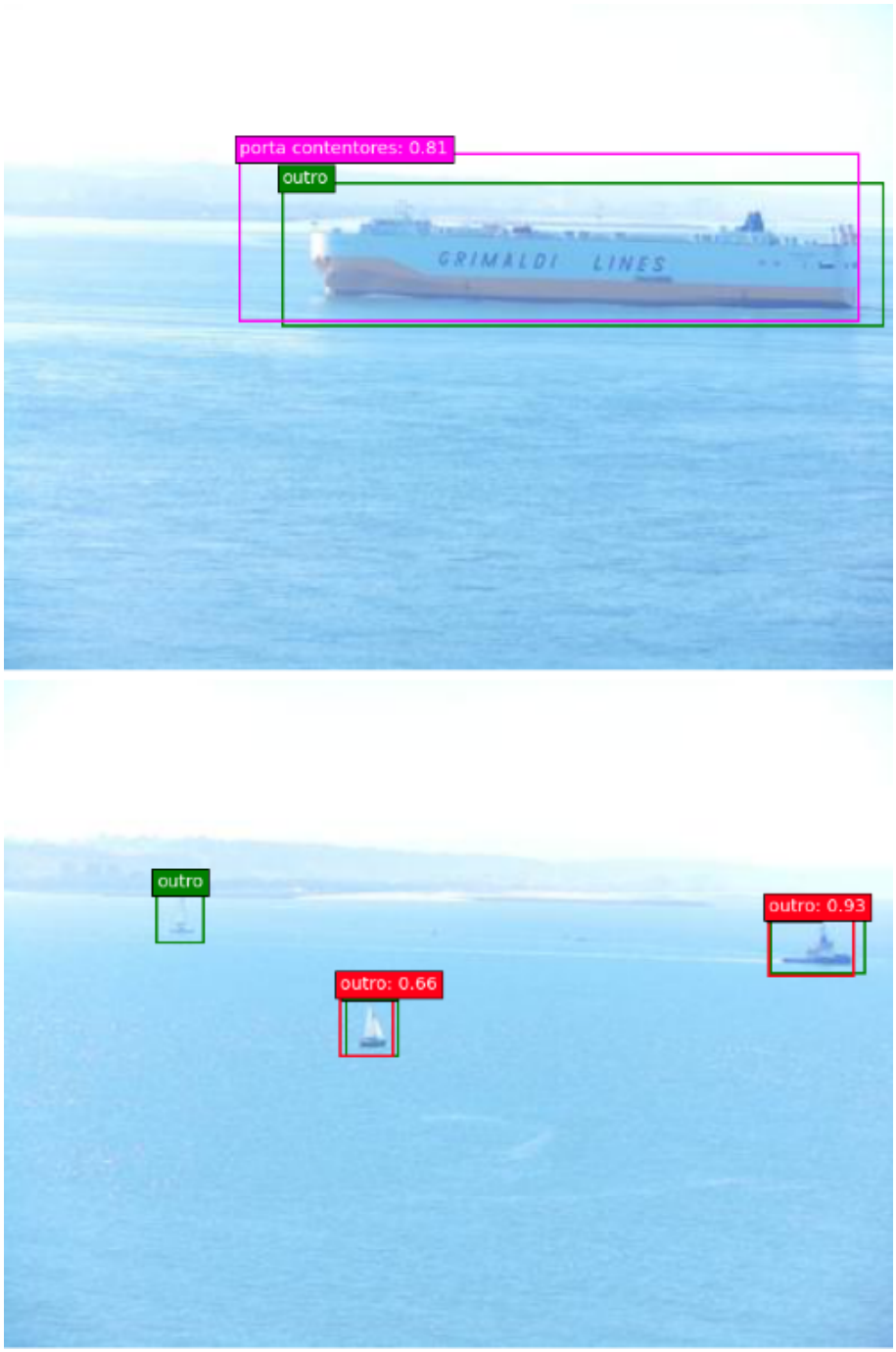


Figura 4.27: Resultados errados de deteção

4.3.3 SSD TransferLearning

Até o momento, foram introduzidos pesos previamente treinados apenas na extração de características base (VGG-16). No entanto, as camadas adicionais de características, bem como as camadas de previsão, estão a ser treinadas do zero. Neste capítulo, será aplicada a técnica de *transfer learning* a todas as camadas, visando aprimorar o modelo. Pois, sempre que possível, é vantajoso transferir pesos previamente treinados em vez de inicializá-los aleatoriamente.

Com isso em mente, optou-se por transferir os pesos do modelo provenientes do treino de um SSD VGG-16 no conjunto de dados COCO. Conforme descrito na teoria em relação à arquitetura das camadas do SSD, as camadas responsáveis pelas previsões têm parâmetros de número de filtros variáveis. Essas variáveis estão relacionadas com o número de âncoras e ao número de classes, o que resulta em camadas com formatos variáveis. Portanto, ao transferir pesos de uma rede para todas as camadas, é essencial garantir conformidade estrutural.

Uma vez que o número de âncoras por camada é o mesmo nas duas redes, as camadas de previsão de localização não apresentam problemas, pois possuem a mesma estrutura. No entanto, as camadas de previsão de confiança dependem do número de classes, o que cria um desafio na transferência de pesos, pois a rede desenvolvida para o conjunto de dados das embarcações contém 7 classes, enquanto aquela que possui pesos treinados no COCO abrange 80 classes. Portanto, é necessário realizar uma sub-amostragem do número de filtros para cada uma das 6 camadas de confiança, ou seja, escolher apenas um subconjunto dos pesos a serem transferidos, a fim de alinhar com a estrutura da rede desenvolvida para embarcações.

Na figura 4.28, [18] é ilustrada a estrutura das camadas de confiança para o COCO e para as embarcações. Por exemplo, na camada conv4_3_mbox_conf, a estrutura é (3,3,512,324), onde os números 3 representam o tamanho do filtro, o 512 é o número de mapas de características que essa camada recebe, e o 324 é igual ao número de classes + 1 multiplicado pelo número de âncoras, ou seja, 81x4. Nessa mesma camadas mas relativa às embarcações, todos os valores são iguais, exceto o último eixo, que agora é 32 (8x4). Portanto, é necessário selecionar apenas 32 pesos dos 324 disponíveis, e o mesmo processo é aplicado aos *bias*. A escolha dos pesos pode ser influenciada pela simila-

ridade entre classes, mas, no caso em que não existem grandes semelhanças entre as classes, optou-se por uma seleção aleatória.

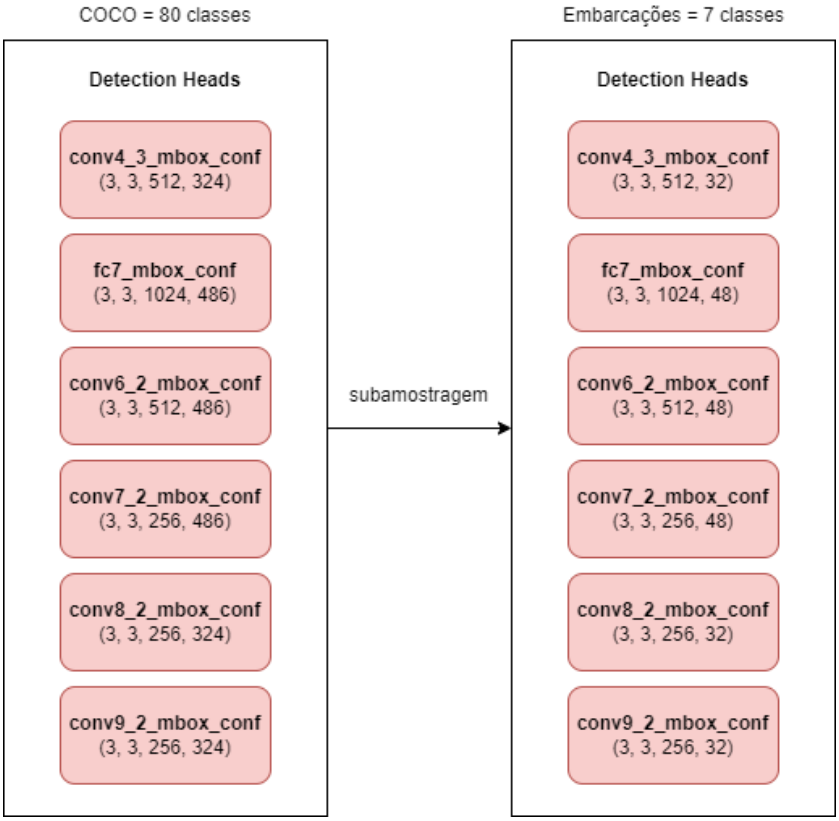


Figura 4.28: Sub-amostragem do número de filtros nas camadas de previsão

Treino

Foram efetuados os mesmos procedimentos descritos nos modelos anteriores. Mais uma vez, o modelo foi submetido a 40 épocas de treino. Os resultados são bastante semelhantes ao caso original, no qual não foi aplicado *transfer learning*. Observa-se uma diminuição da perda em ambos os subconjuntos de dados.

Em relação aos valores da função *loss*, não parece haver uma melhoria significativa. No entanto, os modelos serão comparados com base em métricas de avaliação no próximo capítulo, onde será justificado o uso do *transfer learning*.

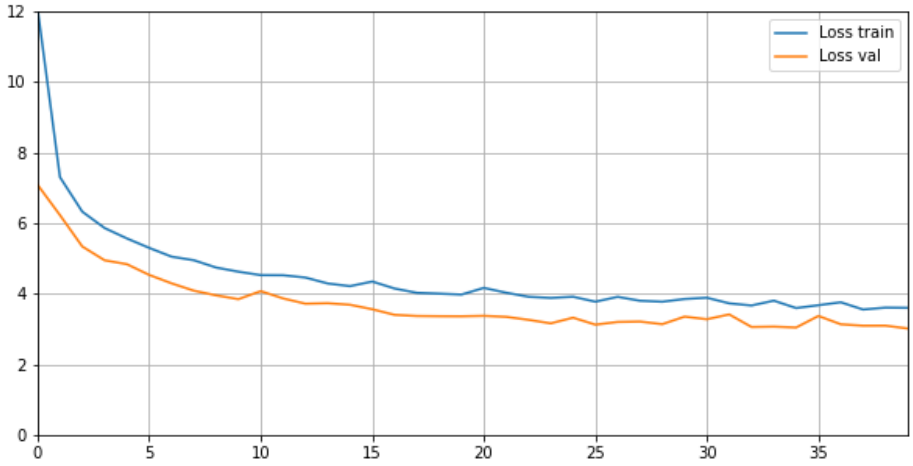


Figura 4.29: Histórico de treino do SSD TransferLearning

4.3.4 SSD MobileNetv2

Nesta fase, o modelo previamente implementado passará por algumas modificações com o objetivo de reduzir o custo computacional, especialmente durante a inferência. Para alcançar esse objetivo, foram realizadas duas alterações significativas.

A primeira modificação envolve a substituição da arquitetura da CNN de base. Agora, é adotado o MobileNetV2, uma arquitetura substancialmente mais leve, composta por aproximadamente 2 milhões de parâmetros. O MobileNetV2 é otimizado para execução eficiente em *hardware* com recursos limitados, ao mesmo tempo mantém um nível notável de precisão. Essa troca de arquitetura permite alcançar resultados de alta qualidade com uma fração dos recursos em comparação com o VGG-16, tornando o MobileNetV2 uma escolha superior.

A segunda alteração realizada envolve a redução do número de camadas de previsão, também conhecidas como "*detection heads*", de 6 para 4 [18]. Essa redução resulta em menos previsões de caixas delimitadoras, o que, por sua vez, leva a um menor custo computacional, principalmente durante a fase de inferência. É importante observar que essa redução no número de camadas de previsão pode impactar ligeiramente o desempenho do processo de deteção, mas, como contrapartida, oferece deteções mais rápidas, tornando especialmente benéfico em cenários onde a velocidade de deteção é crucial.

Em resumo, essas modificações na arquitetura e na estrutura do modelo

podem resultar numa pequena diminuição no desempenho da deteção, mas proporcionam uma notável redução no custo computacional durante a inferência, o que é vantajoso para aplicações que exigem deteções rápidas e eficientes em termos de recursos.

Na figura 4.30, são apresentadas as modificações efetuadas na arquitetura do modelo. Agora, a rede base adota o MobileNetV2, e as quatro camadas de previsão estão conectadas a quatro mapas de características distintos. O primeiro mapa está associado ao bloco 6 do MobileNetV2, que possui uma dimensão de 38x38. Os mapas de características subsequentes são provenientes de camadas extras adicionadas para alcançar as dimensões desejadas: 10x10, 5x5 e 2x2.

Como no SSD original, os mapas de maior dimensão são responsáveis por detetar objetos de menor tamanho, enquanto os de menor dimensão são dedicados à deteção de objetos maiores. Foi cuidadosamente equilibrada a escolha das dimensões dos mapas, de modo a abranger eficazmente os diferentes tamanhos de objetos presentes no conjunto de dados das embarcações.

Essas modificações na arquitetura visam otimizar o desempenho do modelo, permitindo-lhe realizar deteções precisas em objetos de diversas dimensões, o que é essencial para aplicações de deteção de embarcações.

Vale realçar que várias combinações do modelo foram experimentadas antes de se chegar à configuração descrita, abrangendo o número de camadas de previsões, o número de filtros em cada uma dessas camadas e as dimensões dos mapas de características recebidos pelas referidas camadas de previsões.

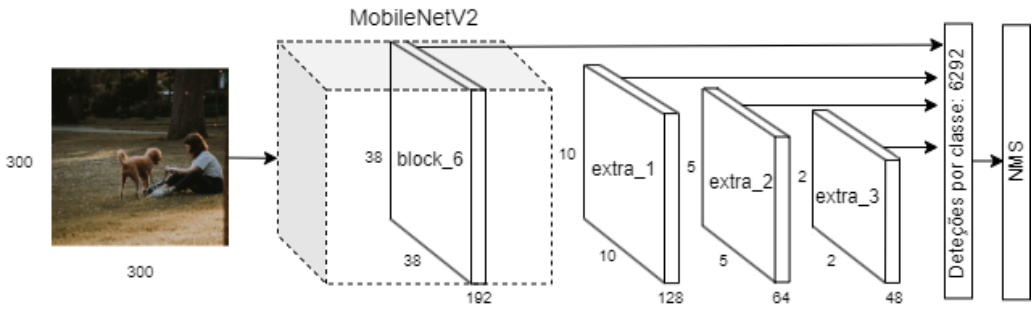


Figura 4.30: Arquitetura SSD MobileV2 com 4 camadas de previsão

Esta versão implementada do SSD MobileNetV2 contém 4 milhões de parâmetros, representando uma redução significativa em comparação com

a versão original, que possuía aproximadamente 24 milhões de parâmetros. Esta redução indica claramente que a nova versão terá um custo computacional mais baixo e é esperada ser uma abordagem mais eficiente para execução no Raspberry Pi 4.

Treino

Conforme mencionado anteriormente, diversas iterações do modelo foram testadas antes de se atingir a versão final. Desta vez o modelo foi submetido a 130 épocas, com o intuito de permitir que a rede fosse capaz de treinar o conjunto de dados para alcançar o seu melhor desempenho possível.

Na figura 4.31, está ilustrada a evolução das funções de perda ao longo das épocas. Com base apenas nas informações do gráfico, sem realizar métricas de avaliação detalhadas, é observado que a época 83 (representada pela linha vertical verde) parece apresentar os melhores valores de pesos. Nessa época, a perda no conjunto de validação aproxima-se da perda no conjunto de treino, e ambos apresentam valores de perda relativamente baixos. No entanto, a partir da época 83, começa a surgir uma disparidade entre as perdas, indicando um possível fenómeno de sobre-aprendizagem.

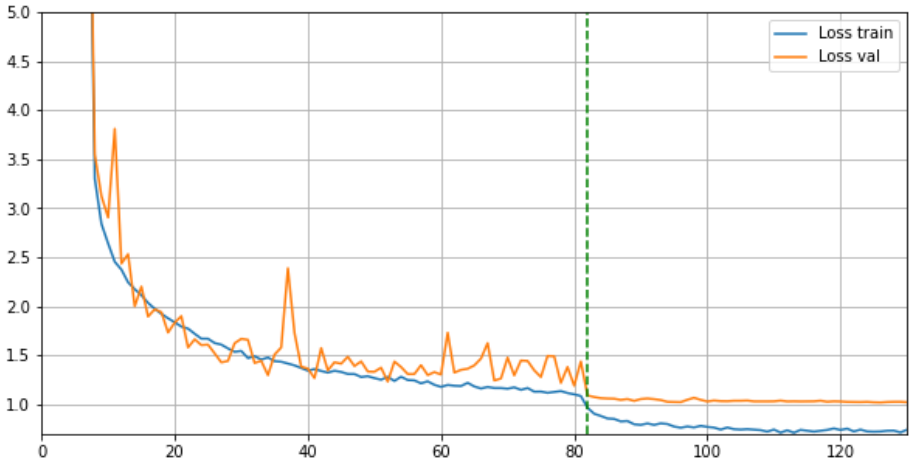


Figura 4.31: Historico de treino MobileNetV2

4.3.5 Deteção em Tempo Real

Após o treino dos modelos de deteção de objetos implementados utilizando o conjunto de dados das embarcações, torna-se essencial, para atingir um dos

principais objetivos deste trabalho, criar um sistema de deteção em tempo real. Nesse sentido, recorre-se a um Raspberry Pi 4, que executa a inferência dos modelos através da receção de *frames* provenientes de uma câmara, estabelecendo, assim, um sistema de deteção em tempo real para deteção automática de embarcações.

Este sistema armazena em memória as imagens das embarcações detetadas, bem como as informações relacionadas às previsões das caixas delimitadoras e as respetivas classes. Essa abordagem possibilita a revisão das deteções posteriormente e a expansão autónoma do conjunto de dados.

A sequência de processos pode ser visualizada na figura 4.32. Inicialmente, é realizada a classificação binária na imagem capturada pela câmara. Somente se a classificação for positiva, o SSD é ativado para realizar a inferência na mesma imagem. Se forem detetados objetos, o sistema armazena a imagem juntamente com as informações das previsões. A razão para não executar apenas o SSD é devido ao seu maior custo computacional em comparação com o classificador binário. Essa abordagem permite reduzir a carga computacional no Raspberry Pi 4, evitando a execução exaustiva do SSD em todas as imagens recebidas. Especificamente, o classificador binário atinge uma taxa de 9 *frames* por segundo, reduzindo para 2 *frames* por segundo durante a execução do detetor de objetos SSD MobileNetV2.

Com o uso do sistema, foi possível armazenar aproximadamente 500 novas imagens de embarcações, devidamente categorizadas com os objetos correspondentes. Este feito representa um marco significativo no trabalho, uma vez que contribui para a expansão do conjunto de dados de embarcações, motivando, assim, o aprimoramento contínuo de sistemas de deteção no âmbito marítimo.

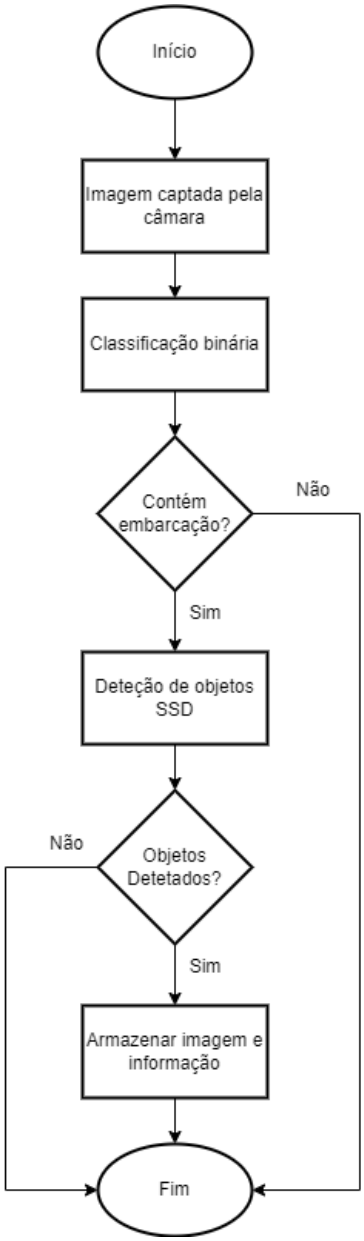


Figura 4.32: Fluxograma do sistema de deteção em tempo real.

Na figura 4.33, é ilustrada a montagem do sistema, com a câmara posicionada numa localização elevada e direcionada para o mar, onde as embarcações transitam. A câmara está conectada a um Raspberry Pi 4, que realiza a inferência das *frames* recebidas. Vale realçar que a localização dessa câmara difere daquela utilizada durante a construção do conjunto de dados.

Essa diferença estratégica permite avaliar com maior rigor a capacidade do modelo em generalizar para situações diversas, como variações na altura da câmara, distância das embarcações, ângulo de visão, entre outros fatores.

Uma vantagem do sistema desenvolvido reside na sua mobilidade, possibilitando o posicionamento ágil em diferentes locais para a recolha de informações sobre as embarcações.

Figura 4.33: Montagem do sistema de deteção.

Capítulo 5

Avaliação

Este capítulo aborda as metodologias utilizadas para a avaliação e teste dos modelos de deteção de objetos previamente desenvolvidos. Aqui, serão apresentadas as técnicas e métricas de avaliação utilizadas para quantificar o desempenho desses modelos, além de serem discutidas as comparações e os resultados obtidos entre eles.

5.1 Classificação binária

A seguir, é exposta a avaliação do modelo binário desenvolvido, o qual será utilizado como uma etapa preliminar de filtragem no sistema final.

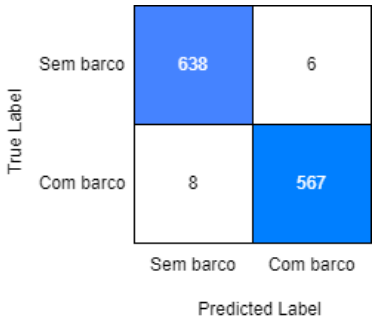


Figura 5.1: Matriz de confusão da classificação binária

Analisando a matriz de confusão relativa à classificação dos dados de teste, é percetível que o modelo apresentou um desempenho adequado ao classificar corretamente 638 casos como negativos e 567 casos como positivos.

No entanto, foi identificada a ocorrência de 6 Falsos Positivos (FP), indicando que houve 6 instâncias classificadas como positivas quando, na realidade, eram negativas. Além disso, foram observados 8 Falsos Negativos (FN), o que significa que houve 8 casos classificados como negativos quando, na verdade, eram positivos. É relevante destacar que existe uma discrepância ligeira entre o número de exemplos positivos e negativos, com um total de 644 exemplos negativos e 575 positivos. Portanto, é compreensível que haja uma discrepância ligeira, embora notável, entre os Falsos Negativos (FN) e os Falsos Positivos (FP), o que é confirmado pela matriz de confusão.

Em seguida, o modelo passará por um processo de calibração, no qual o limite de decisão será ajustado de 0.5 para 0.4. Esse ajuste visa proporcionar ao modelo uma maior sensibilidade na classificação de casos positivos. A justificação para essa abordagem está relacionada com o contexto específico do problema em questão. Na minha perspetiva, dado que se trata de um problema relacionado com o tráfego marítimo, existe uma ligeira vantagem em ter um modelo que identifica mais casos positivos. Isso resultaria numa maior probabilidade de deteção de embarcações, contribuindo, assim, para o aumento da segurança e do controlo. A seguir, será apresentada a matriz de confusão referente à calibração do modelo, sem comprometer o desempenho, existindo agora um aumento de TP:

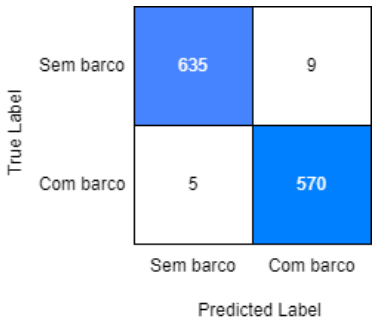


Figura 5.2: Matriz de confusão da classificação binária depois da calibração

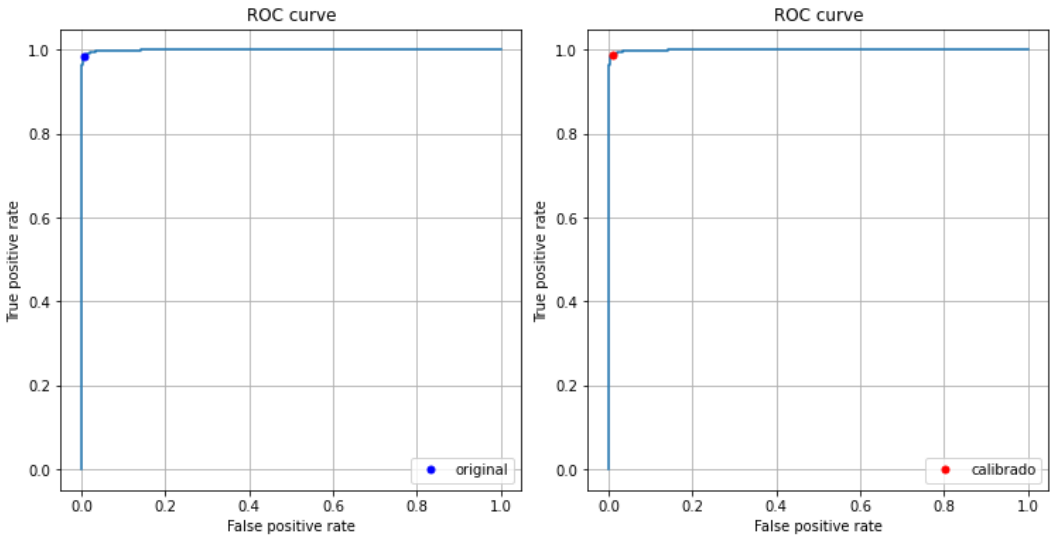


Figura 5.3: Curvas ROC antes e depois da calibração

As curvas ROC na figura 5.3 revelam que a calibração resultou num modelo com mais TP, confirmando o esperado. Além disso, pode-se concluir que o modelo apresenta um desempenho satisfatório, caracterizado por um TP substancial e um FP reduzido.

Conforme mencionado anteriormente, a calibração é motivada pelo contexto do tráfego marítimo. No caso do uso deste modelo para captura de imagens para a expansão do conjunto de dados de embarcações, o limiar de decisão é ajustado para aumentar TN. Isso deve ao objetivo de assegurar que as imagens coletadas englobem, de facto, embarcações.

A *accuracy* é um indicador do desempenho do modelo e é calculada como a razão entre o número de previsões corretas e o total de previsões. Resultando então numa precisão de 98.85%, o que justifica o bom desempenho do modelo.

Como parte da etapa final da análise das classificações do modelo, são apresentados a seguir três exemplos de TP (Verdadeiro Positivo), TN (Verdadeiro Negativo), FP (Falso Positivo) e FN (Falso Negativo).

Na figura 5.4, para as três primeiras imagens, observa-se um valor de saída da rede, proveniente da função sigmoid, próximo de 1, o que indica que o modelo possui alta confiança de que essas imagens pertencem à classe de presença de barcos. Em contrapartida, no caso dos TN (imagem 4-6), o valor de saída é próximo de 0, indicando que o modelo possui alta confiança

de que essas imagens pertencem à classe de ausência de barcos.

Figura 5.4: Exemplos corretamente previstos

Na figura 5.5, encontram-se situações em que a classificação do modelo resultou em erro. No entanto, é de realçar que, na imagem 8, correspondente aos Falsos Positivos (FP), existe um valor de saída da rede consideravelmente elevado, neste caso, o classificador acerta na classificação, uma vez que ocorreu um equívoco na categorização desta imagem. Já nos exemplos de Falsos Negativos (FN), a rede demonstra uma menor confiança em relação à classificação, uma vez que os valores de saída não se aproximam de zero.

Figura 5.5: Exemplos incorretamente previstos

Conclui-se, portanto, que a rede, de forma geral, obteve um desempenho elevado. É relevante observar que o modelo apresenta algumas dificuldades

na deteção de embarcações de menor porte. Tal comportamento é compreensível, uma vez que o conjunto de dados possui uma certa ambiguidade, dado que existem um vestígio de imagens em que embarcações menores são classificadas como negativas, e vice-versa.

5.2 Métricas de avaliação em deteção de objetos

Na avaliação de modelos de deteção de objetos, as metodologias adotadas diferem consideravelmente daquelas utilizadas em problemas de classificação binária ou multi-classe. De seguida, serão apresentadas e discutidas as estratégias e abordagens específicas que compõem o processo de avaliação dos modelos de deteção de objetos, visando garantir uma avaliação sólida e precisa desses modelos.

5.2.1 Curva *Precision* e *Recall*

A curva de *precision-recall* [14] fornece uma representação gráfica que facilita a interpretação visual de como as alterações no valor de limiar de confiança afetam a *precision* e o *recall* do modelo de deteção. Para isso, o primeiro passo é determinar os verdadeiros positivos (TP) e os falsos positivos (FP). Isso é realizado pelo IoU, na qual as caixas delimitadoras previstas pelo modelo são comparadas com as caixas delimitadoras verdadeiras. Caso o valor de IoU seja superior a 0,5, considera-se que essa deteção corresponde a um verdadeiro positivo (TP); caso contrário, se o resultado da operação for inferior a 0,5, a deteção é classificada como um falso positivo (FP).

Com a identificação dos TP e FP, é possível calcular o *recall* e a *precision* utilizando as seguintes fórmulas:

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

A métrica de *precision* avalia a habilidade de um modelo em realizar previsões positivas de forma correta, enquanto a métrica de *recall* avalia

a capacidade do modelo em identificar a totalidade dos exemplos positivos presentes num conjunto de dados.

Para cada classe presente, [10] as previsões são ordenadas pela pontuação de confiança e são calculados os valores de precisão e *recall*. Uma avaliação positiva do modelo, com base na análise dessa curva, ocorre quando a precisão se mantém elevada à medida que o *recall* aumenta. Indicando que mesmo com variações na confiança, ambos os valores permanecem altos.

5.2.2 mAP

Após a construção da curva PR para cada classe, são extraídos valores de precisão com um espaçamento de *recall*. Em seguida, é calculada a *Average Precision* (AP), que consiste na média dos valores de precisão obtidos. É importante observar que o valor de precisão obtido é o resultado de uma interpolação, onde é selecionado o valor mais alto de precisão à direita do ponto de interesse [15].

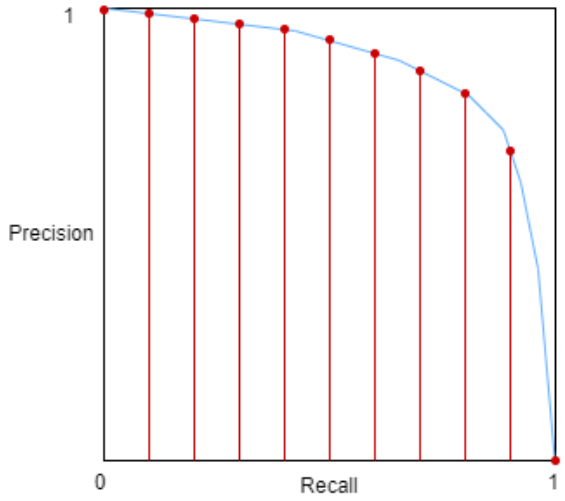


Figura 5.6: Curva PR com 11 pontos de interesse

$$AP = \frac{1}{11} \sum_{R \in \{0, 0.1, \ldots, 1\}} Pinterp(R)$$

Onde

$$Pinterp(R) = max_{\tilde{R} \geq R} P(\tilde{R})$$

A precisão interpolada em um ponto de *recall* (R) é o máximo da precisão obtida para qualquer ponto de recall ($\tilde{R}$) maior ou igual a R ao longo da curva precisão-recall. Significa que a precisão interpolada num ponto de *recall* é o maior valor de precisão alcançado para todos os pontos ao longo da curva de precisão-recall que possuem um *recall* maior ou igual a R.

Após a obtenção dos valores de AP para cada classe, é calculada a média desses valores, resultando no *Mean Average Precision* (mAP):

$$mAP = \frac{1}{N} \sum_{i=1}^{N} APi$$

5.2.3 *Frames* por segundo (FPS)

Uma métrica também de grande importância nos modelos de deteção de objetos é em relação à velocidade de execução, um fator crucial para aplicações em tempo real. Essa métrica [10] pode ser avaliada em termos da quantidade de *frames* processados por segundo (fps) por um determinado modelo, onde um maior número de fps indica um custo computacional inferior para o modelo. É importante destacar que a comparação entre os fps de diferentes modelos é válida apenas quando esses modelos são executados no mesmo sistema computacional.

O número de fps necessário para uma aplicação em tempo real varia significativamente dependendo da natureza do problema em questão. No caso de deteção de embarcações, dado que essas não costumam se deslocar a altas velocidades, não é requisito obter uma taxa de fps elevada.

Para avaliar e comparar as taxas fps dos modelos desenvolvidos, procedeu-se à execução em tempo real desses modelos no sistema constituído pelo Raspberry Pi 4. Foram, então, registados os números de *frames* por segundo para os diferentes modelos durante a execução em inferência em tempo real.

5.3 SSD Embarcações

Foram selecionados os pesos do modelo referentes à última época de treino, correspondente à época 40.

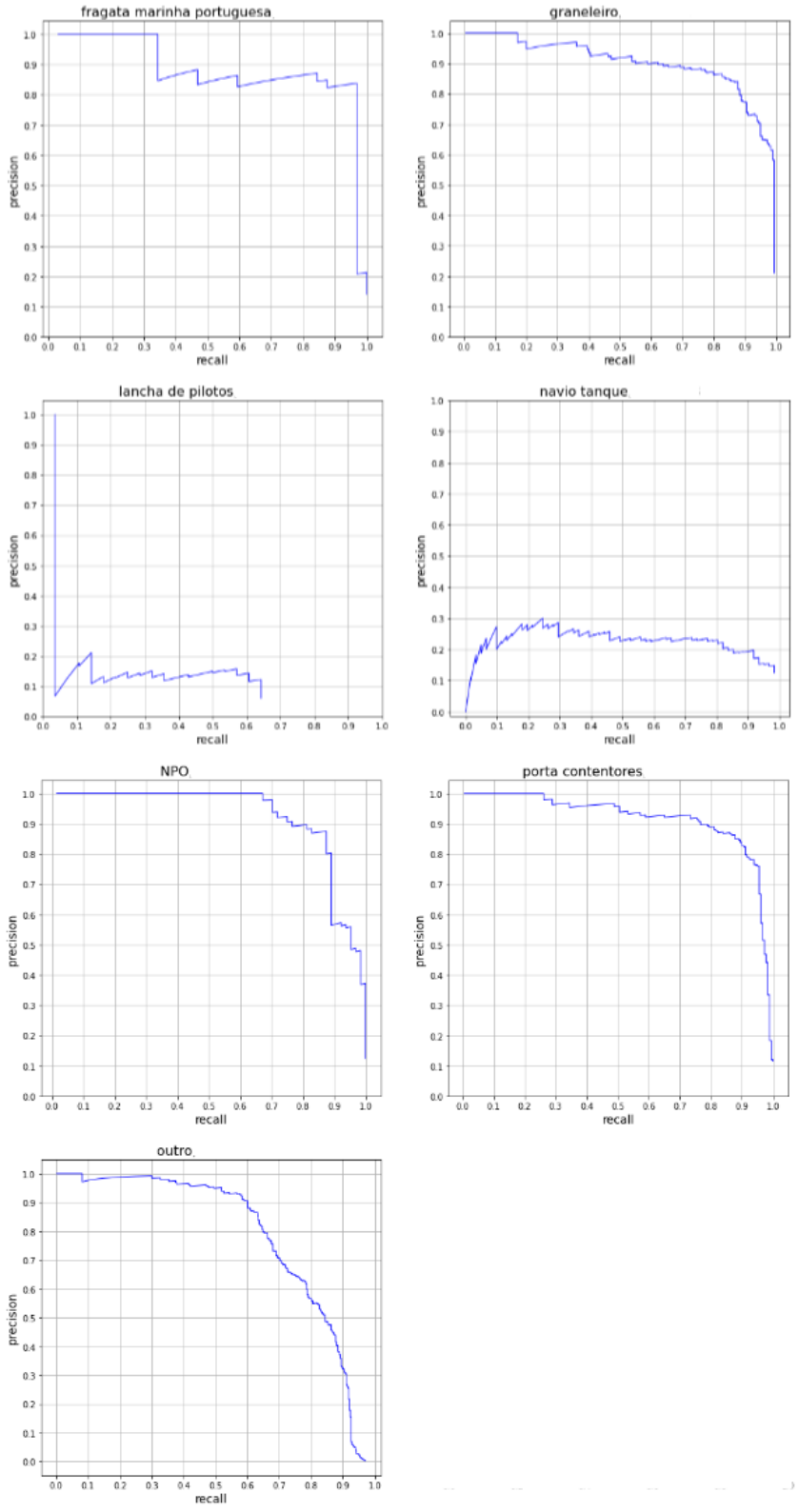


Figura 5.7: Curva PR para cada classe SSD Embarcações

fragata marinha pt	AP	0.856
graneleiro	AP	0.841
lancha de pilotos	AP	0.18
navio tanque	AP	0.233
NPO	AP	0.893
porta contentores	AP	0.873
outro	AP	0.763
	mAP	0.663

Tabela 5.1: mAP SSD Embarcações

Ao analisar as curvas PR e os valores de AP para cada classe, pode-se concluir que o modelo demonstra uma alta precisão na deteção da maioria das classes. No entanto, o desempenho é significativamente menor quando se trata das classes "Lancha de Pilotos" e "Navio Tanque", com pontuações AP de 0,18 e 0,23, respetivamente. Globalmente, o modelo exibe um desempenho razoável, mas há margem para melhorias, especialmente nas duas classes com baixos valores de AP.

Esta constituiu a primeira versão testada em tempo real, durante a qual foram registados apenas 0.3 fps. Este valor não atende aos requisitos de um monitorização em tempo real adequada, mesmo considerando que as embarcações se deslocam a baixas velocidades. Por este motivo, o modelo não foi treinado por mais de 40 épocas, uma vez que o seu custo computacional é excessivamente elevado para a tarefa em questão. Vale realçar que o Raspberry Pi 4, utilizado para inferência em tempo real, possui capacidade computacional limitada, o que contribui para o tempo prolongado de inferência.

5.4 SSD TransferLearning

A adoção da estratégia de *transfer learning* visou melhorar o desempenho do modelo anterior. Semelhante à situação anterior, a escolha dos pesos do modelo recaiu sobre a última época de treino, que correspondendo à época 40.

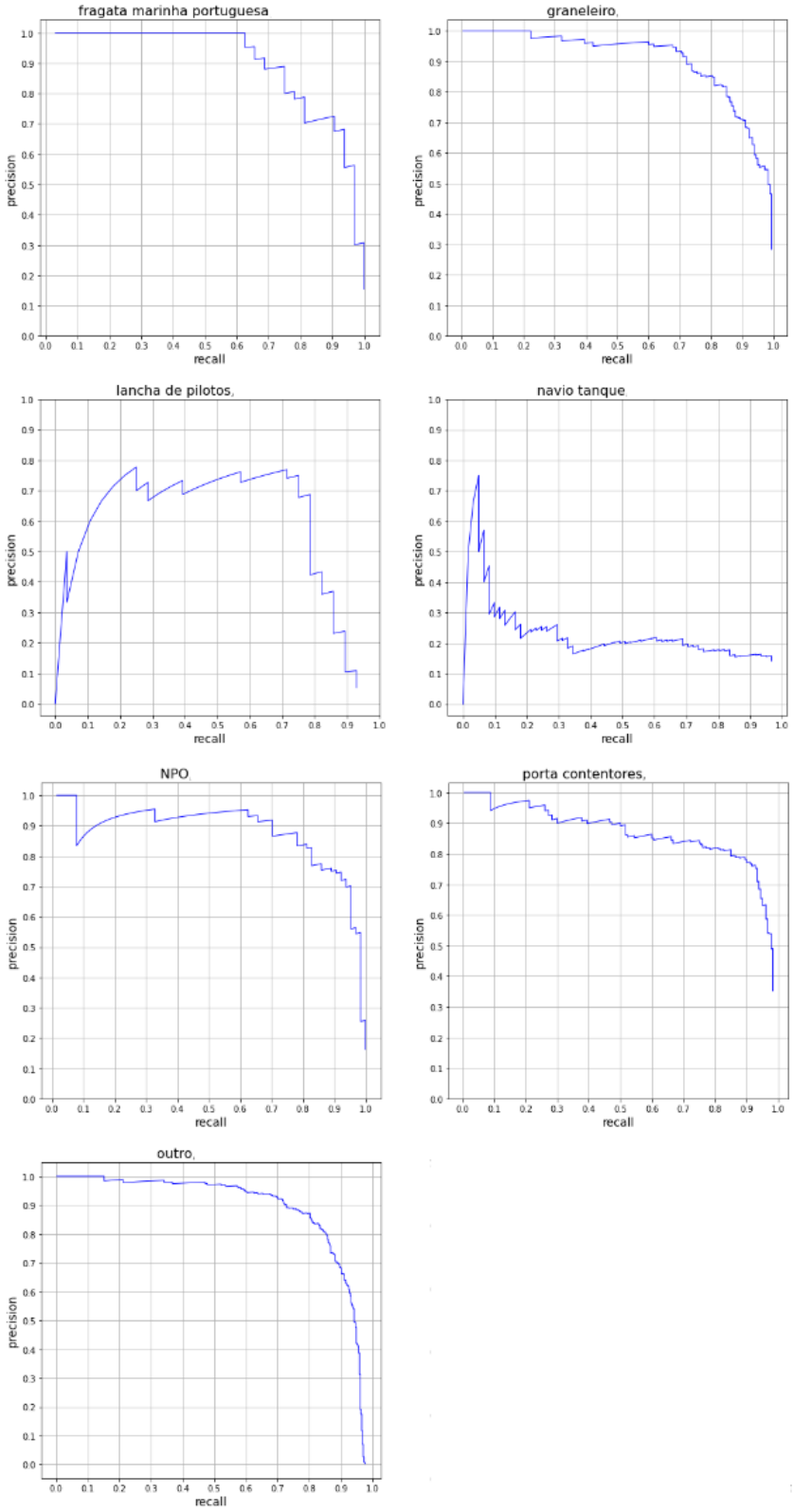


Figura 5.8: Curva PR para cada classe SSD Transfer Learning

fragata marinha pt	AP	0.883
graneleiro	AP	0.851
lancha de pilotos	AP	0.611
navio tanque	AP	0.249
NPO	AP	0.863
porta contentores	AP	0.817
outro	AP	0.85
	mAP	0.732

Tabela 5.2: mAP SSD Transfer Learning

Ao analisar os resultados de AP, é evidente uma melhoria significativa, sobretudo na classe de "lancha de pilotos", na qual se registou um aumento de 0.43. Isso demonstra claramente a eficácia da estratégia de transferência de pesos, que contribuiu para a criação de um modelo mais preciso.

No entanto, é importante notar que a classe "navio tanque" não apresentou uma melhoria tão substancial. É possível que um treino mais extenso seja necessário para aprimorar a deteção.

5.5 SSD MobileNetv2

Neste último modelo desenvolvido, o treino foi estendido consideravelmente em relação aos casos anteriores, de forma a aprimorar o sistema de deteção ao máximo. Conforme mencionado na secção de implementação, a época 83 foi selecionada como uma das iterações de treino que resultará nos melhores resultados, com menor risco de *overfitting*.

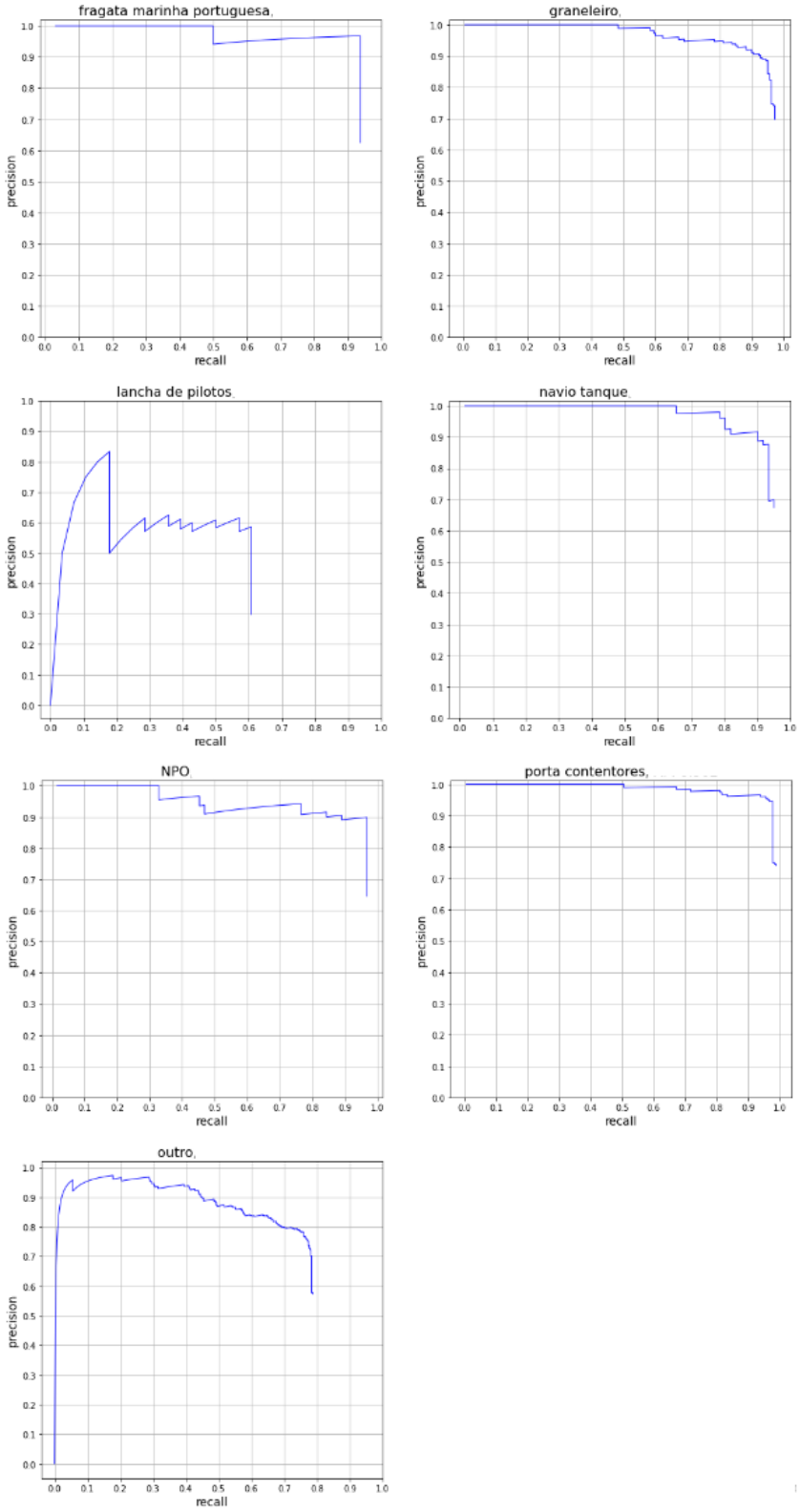


Figura 5.9: Curva PR para cada classe SSD MobileNetV2

fragata marinha pt	AP	0.897
graneleiro	AP	0.889
lancha de pilotos	AP	0.43
navio tanque	AP	0.896
NPO	AP	0.873
porta contentores	AP	0.902
outro	AP	0.665
	mAP	0.793

Tabela 5.3: mAP SSD MobileNetV2

Ao analisar os resultados de avaliação, torna-se evidente que este modelo apresenta o melhor desempenho, apesar de ser consideravelmente mais leve. A principal razão para esses resultados promissores foi um treino mais extenso. É de esperar que o modelo original, com um maior número de camadas de previsão, seria superior em termos de desempenho. A melhoria mais notável ocorreu na classe de "navio tanque", que agora alcança uma pontuação de 0.89 em AP, no entanto existe uma redução na pontuação AP da classe "outro".

Conforme esperado devido à implementação de um modelo mais leve, foi possível alcançar uma taxa de 2 fps na deteção em tempo real. Esse valor é considerado adequado para realizar uma monitorização eficaz do tráfego das embarcações.

Por fim, serão apresentados alguns dos resultados obtidos na deteção em tempo real, utilizando o Raspberry Pi:

Figura 5.10: Exemplo deteção tempo real 1

Figura 5.11: Exemplo deteção tempo real 2

Figura 5.12: Exemplo deteção tempo real 3

Figura 5.13: Exemplo deteção tempo real 4

Capítulo 6

Conclusões e trabalho futuro

Neste trabalho, foram exploradas vários pormenores da deteção de objetos, com um foco particular na aplicação de modelos *Single Shot MultiBox Detector* (SSD) para a deteção de embarcações. Além disso, foi desenvolvida uma versão mais leve do SSD, otimizada para sistemas computacionais limitados, como o Raspberry Pi 4.

Ao longo deste estudo, ficou evidente que a escolha entre precisão e velocidade é uma consideração crítica. A adaptação do modelo para obter tempos de deteção mais rápidos em detrimento de alguma precisão tornou-se essencial para a eficácia do sistema em cenários de tempo real com recursos limitados. Este compromisso entre velocidade e precisão é uma característica comum em aplicações do mundo real, onde a latência desempenha um papel crucial.

É importante destacar que, nesta área de pesquisa em constante evolução, os resultados muitas vezes só podem ser verdadeiramente compreendidos e melhorados por meio de extensos testes. A otimização de modelos e parâmetros requer tempo e esforço consideráveis, mas os resultados alcançados até agora demonstram uma solução viável e promissora para o problema abordado neste trabalho.

Assim, este estudo contribuiu para a compreensão da aplicação do SSD na deteção de embarcações, fornecendo uma base sólida para futuros desenvolvimentos e ajustes, à medida que a busca por sistemas de deteção de objetos mais rápidos e precisos continua.

Trabalho futuro

Este trabalho abre portas para diversas áreas de pesquisa e desenvolvimento que podem melhorar ainda mais o sistema de deteção e classificação automática de embarcações. Algumas das direções para trabalhos futuros incluem:

- **Treino num conjuntos de dados diversificados:** Expandir o treino do modelo com conjuntos de dados de embarcações mais diversificados. Isso possibilitará uma melhor generalização do modelo para lidar com uma variedade maior de embarcações em diferentes condições, fortalecendo assim a capacidade de deteção em cenários do mundo real.

- **Modificações no modelo SSD:** Explorar modificações no modelo SSD, como ajustar o número de camadas de previsões, o tamanho dos *feature maps* ou as especificações das *default boxes*. Isso permitirá uma adaptação mais precisa às características específicas das embarcações em termos de tamanho e forma, melhorando assim a precisão da deteção.

- **Utilização de aceleradores de inferência:** Investigar a integração de aceleradores de inferência, como o Google Coral, no sistema para aumentar significativamente a velocidade de deteção. Estes são projetados para otimizar operações de inferência em redes neuronais, o que pode ser crucial para cenários de tempo real.

- **Exploração de outros detetores:** Experimentar outros detetores populares, como o YOLO (*You Only Look Once*), para o problema em questão e comparar o seu desempenho com o SSD. Essa comparação pode revelar informações interessantes sobre qual detetor é mais adequado neste cenário.

- **Criação da interface de utilizador:** Desenvolver uma interface de utilizador amigável para facilitar a operação e monitorização do sistema em tempo real. Isso tornaria a solução mais acessível e utilizável por operadores e gestores de tráfego marítimo.

- **Armazenamento em *cloud*:** Implementação de um sistema de armazenamento em *cloud*, permitindo que as imagens contendo embarcações

detetadas sejam armazenadas, em vez de serem guardadas em memória local. Permitindo assim uma gestão eficiente de informações no contexto deste sistema.

Para sumarizar, existe um vasto campo de oportunidades para aprimorar e expandir a aplicação de sistemas de deteção e classificação automática de embarcações. As futuras pesquisas e desenvolvimentos podem contribuir significativamente para tornar essa tecnologia mais eficaz e confiável em cenários do mundo real, beneficiando áreas como gestão portuária e controle de tráfego marítimo.

Bibliografia

[1] Farahnakian, F., Heikkonen, J. (2020). Deep Learning Based Multi-Modal Fusion Architectures for Maritime Vessel Detection.

[2] Bogdan Iancu, Valentin Soloviev, Luca Zeliol, Johan Lilius (2021). ABOships—An Inshore and Offshore Maritime Vessel Detection Dataset with Precise Annotations.

[3] Sung-Jun Lee, Myung-Il Roh, Min-Jae Oh (2020). Image-based ship detection using deep learning.

[4] Zhenfeng Shao, Wenjing Wu , Zhongyuan Wang , Wan Du, Chengyuan Li (2018). SeaShips: A Large-Scale Precisely Annotated Dataset for Ship Detection.

[5] Pranshu Sharma (2022). Basic Introduction to Convolutional Neural Network in Deep Learning.

[6] Sumit Saha (2018). A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.

[7] Jun94 (2020). Object detection with Sliding Window and Feature Extraction(HoG).

[8] Fast Object Detection with Fast R-CNN (2018).

[9] Rohith Gandhi (2018). R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms.

[10] Mohamed Elgendy (2020). Deep Learning for Vision Systems.

[11] Zoumana Keita (2020). YOLO Object Detection Explained.

[12] Rohit Kundu (2023). YOLO: Algorithm for Object Detection Explained.

[13] IBM. Neural networks.

[14] Darshankumar Patel (2020). Single Shot Detector for Object Detection using an Ensemble of Deep Learning and Statistical Modelling for Robot Learning Applications

[15] Simen Viken Grini (2019). Object Detection in Maritime Environments

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy,Scott Reed, Cheng-Yang Fu, Alexander C. Berg (2016). SSD: Single Shot MultiBox Detector.

[17] ankur6ue (2018). Data Augmentation in SSD (Single Shot Detector).

[18] Pierluigi Ferrari (2018). SSD Keras: Single-Shot MultiBox Detector implementation in Keras.

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi (2016). You Only Look Once: Unified, Real-Time Object Detection.

[20] J Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database.

[21] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, Andrew Zisserman (2014). The PASCAL Visual Object Classes Challenge.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár (2014) Microsoft COCO: Common Objects in Context.

[23] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun (2016). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.

[24] Ricardo Antonello. Introdução a Visão Computacional com Python e OpenCV.