UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Vitor Hugo Fernandes Oliveira

DETECÇÃO DE OBJETOS ATRAVÉS DE REDES NEURAIS CONVOLUCIONAIS BASEADAS EM REGIÕES

Florianópolis
2024

Vitor Hugo Fernandes Oliveira

DETECÇÃO DE OBJETOS ATRAVÉS DE REDES NEURAIS CONVOLUCIONAIS BASEADAS EM REGIÕES

Trabalho de Conclusão de Curso do Curso de Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal de Santa Catarina para a obtenção do título de bacharel em Engenharia Elétrica.
Orientador: Prof. Joceli Mayer, Ph.D.

Florianópolis
2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Oliveira, Vitor Hugo Fernandes
DETECÇÃO DE OBJETOS ATRAVÉS DE REDES NEURAIS CONVOLUCIONAIS BASEADAS EM REGIÕES / Vitor Hugo Fernandes Oliveira ; orientador, Joceli Mayer, 2024.
58 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Elétrica, Florianópolis, 2024.

Inclui referências.

1. Engenharia Elétrica. 2. RCNN. 3. Detecção de objetos. 4. Region Based CNN. I. Mayer, Joceli. II. Universidade Federal de Santa Catarina. Graduação em Engenharia Elétrica. III. Título.

Vitor Hugo Fernandes Oliveira

DETECÇÃO DE OBJETOS ATRAVÉS DE REDES NEURAIS CONVOLUCIONAIS BASEADAS EM REGIÕES

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "bacharel em Engenharia Elétrica" e aprovado em sua forma final pelo Curso de Graduação em Engenharia Elétrica.

Florianópolis, 04 de dezembro de 2024.

Banca Examinadora:

Documento assinado digitalmente
JOCELI MAYER
Data: 16/12/2024 12:42:26-0300
CPF: ***.833.519-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Joceli Mayer, Ph.D.
Orientador

Documento assinado digitalmente
Bruno Catarino Bispo
Data: 17/12/2024 10:09:49-0300
CPF: ***.973.077-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Bruno C. Bispo, Ph.D.
Avaliador
UFSC

Documento assinado digitalmente
Cesar Ramos Rodrigues
Data: 18/12/2024 07:36:09-0300
CPF: ***.557.990-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Cesar R. Rodrigues, Ph.D.
Avaliador
UFSC

Este trabalho é dedicado à Deus, minha querida família e ao meu futuro.

AGRADECIMENTOS

Gostaria de agradecer primeiramente à Deus pelas suas bênçãos em minha vida e todas as oportunidades que Ele me proporcionou. À minha família por todo o amor e apoio que me concederam e permitiram que tudo isso fosse possível. À minha namorada por demonstrar que a vida deve ser partilhada ao lado de pessoas que ama. Aos meus amigos que me proporcionaram apoio e grandes dias de felicidade nesta jornada universitária. E por fim, gostaria de agradecer ao meu passado e todos os momentos que vivenciei que me tornaram esta pessoa que admiro tanto.

"I've failed over and over again. And that is why I succeed."
Michael Jordan, 1997

RESUMO

Este trabalho visa estudar os conceitos de uma família específica de redes neurais, as RCNN (Region-Based Convolutional Neural Networks). O objetivo é apresentar o contexto teórico dessas redes e os princípios subjacentes que as fundamentam. Foram analisadas as redes RCNN, *Fast* RCNN, *Faster* RCNN e *Mask* RCNN, descrevendo suas arquiteturas, particularidades e aplicações. Além desta análise, o trabalho também inclui uma aplicação prática de uma adaptação da RCNN, abordando os conceitos técnicos relacionados à sua implementação. Essa adaptação permitiu observar empiricamente que, embora os resultados fossem satisfatórios, a evolução desta abordagem era necessária para atender a aplicações mais avançadas de forma mais eficiente, reforçando os aspectos evolutivos discutidos na teoria.

Palavras-chave: RCNN, Detecção de objetos, Region Based CNN

ABSTRACT

This work aims to study the concepts of a specific family of neural networks, the RCNNs (Region-Based Convolutional Neural Networks). The objective is to present the theoretical context of these networks and the underlying principles that support them. The RCNN, *Fast* RCNN, *Faster* RCNN, and *Mask* RCNN networks were analyzed, detailing their architectures, particularities, and applications. In addition to this analysis, the study also includes a practical application of an RCNN adaptation, addressing the technical concepts related to its implementation. This adaptation empirically demonstrated that, while the results were satisfactory, the evolution of this approach was necessary to meet more advanced applications more efficiently, reinforcing the evolutionary aspects discussed in the theory.

Keywords: RCNN, Object Detection, Region Based CNN

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O processamento digital de imagens (PDI) é uma área do conhecimento que aborda a análise e manipulação de imagens digitais através de uma variedade de algoritmos e modelos matemáticos. Segundo (GONZALEZ; WOODS, 2017), autores renomados no campo, o PDI abrange desde operações fundamentais de pré-processamento, como redução de ruído e ajuste de contraste, até técnicas avançadas como segmentação de objetos e reconhecimento semântico. Essas técnicas são cruciais para a extração de informações significativas das imagens e para a automação de tarefas baseadas em dados visuais.

Um avanço recente nos últimos anos tem sido a integração de redes neurais artificiais, especialmente redes convolucionais (CNNs), que revolucionaram a capacidade de processamento e análise de imagens. As CNNs são projetadas para reconhecer padrões complexos diretamente dos pixels de entrada, permitindo tarefas como classificação de imagens, detecção de objetos e segmentação semântica com uma precisão e eficiência sem precedentes. Estas redes abrem novas fronteiras em áreas como diagnóstico médico, monitoramento ambiental e interpretação de dados visuais complexos em diversas aplicações industriais e científicas.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é compreender a evolução das técnicas de uma família de redes neurais baseadas em regiões, bem como avaliar e testar empiricamente essa evolução. O estudo foca na compreensão teórica dos conceitos e técnicas associadas a R-CNN, *Fast* R-CNN, *Faster* R-CNN e *Mask* R-CNN, destacando os aspectos evolutivos presentes em cada um destes modelos. Adicionalmente, o trabalho busca desenvolver uma demonstração da primeira rede desta família, evidenciando de forma prática os principais conceitos que fundamentam este conjunto de abordagens. Especificamente, pretende-se realizar uma adaptação da R-CNN a partir de um subconjunto de dados do COCO *dataset* a fim de demonstrar o percurso de desenvolvimento do reconhecimento e propostas das regiões de interesse através do *Selective Search* e de redes neurais. Esta demonstração visa mostrar que, embora o processo ofereça resultados satisfatórios, ele apresenta ineficiências que demandam uma evolução para aplicações mais eficientes.

2 REFERENCIAL TEÓRICO

Processamento digital de sinais é um dos pilares da tecnologia moderna que permitiu grandes avanços significativos na ciência, na melhoria da qualidade de vida da população e no avanço da sociedade como um todo. Este trabalho traz o foco para uma das áreas contidas no PDS que é o processamento digital de imagens (PDI).

2.1 PROCESSAMENTO DIGITAL DE IMAGENS

O processamento digital de imagens é o uso de algoritmos e modelos matemáticos para processar e analisar imagens digitais contendo como objetivo a melhoria da qualidade das imagens, extração de informações significativas e automatização de tarefas baseadas em imagens (GONZALEZ; WOODS, 2017).

2.2 CLASSIFICAÇÃO, DETECÇÃO E SEGMENTAÇÃO DE IMAGENS

Este trabalho trouxe o foco para o processamento de imagens digitais de alto nível, com ênfase em três técnicas fundamentais: classificação, detecção e segmentação, que são a base da visão computacional moderna.

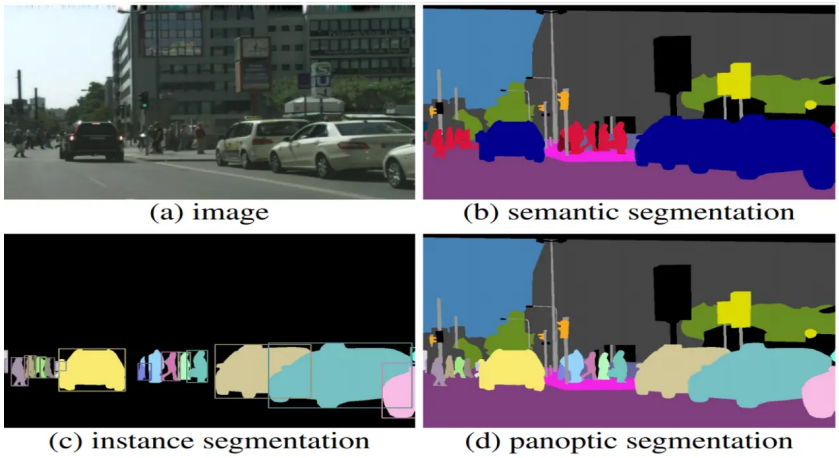
A classificação consiste em criar um algoritmo capaz de atribuir uma única classe ou rótulo a uma imagem, com base em características visuais como cores, texturas, formas e padrões. Na detecção, o objetivo é localizar objetos específicos na imagem. O algoritmo identifica regiões com características semelhantes às do conjunto de treinamento e marca a localização do objeto, rotulando as áreas associadas a ele. Já a segmentação divide a imagem em regiões distintas com base em suas características. Esses algoritmos classificam cada pixel individualmente, utilizando máscaras que delimitam os contornos das áreas de interesse.

Diferentemente da classificação e da detecção, a segmentação se concentra na delimitação precisa dos contornos, atribuindo a cada pixel sua classe correspondente. Enquanto técnicas clássicas de segmentação analisam propriedades como cor e intensidade, modelos de aprendizado profundo usam redes neurais para identificar padrões mais complexos.

Olhando para o contexto da segmentação, ela pode ser dividida em três tipos essencialmente: segmentação semântica, segmentação de instâncias e segmentação panóptica. A Figura 1 mostra a diferença entre estes tipos de segmentação (SULTANA; SUFIAN; DUTTA, 2020).

A segmentação semântica é focada na atribuição de uma única classe ou categoria a cada pixel na imagem. Isso significa que todos os pixels que representam o mesmo objeto ou conceito são agrupados sob uma etiqueta de segmentação comum. A ideia principal é de segregar a imagem entre o que os autores chamam de "*Stuff*" e "*Things*". "*Stuff*" pode ser compreendido como o plano de fundo das imagens ou objetos que não tem formato

Figura 1 – Comparativo segmentações de imagens



Fonte: (SULTANA; SUFIAN; DUTTA, 2020)

específico como céu, rua, paredes, enquanto "*Things*" são objetos contáveis e instâncias de maior interesse como pessoas, carros, objetos em geral (CHUANG; ZHANG; ZHAO, 2023). Por exemplo, em uma cena com carros, árvores e pessoas, a segmentação semântica separaria os pixels correspondentes a cada uma dessas categorias, mas não distinguiria diferentes instâncias de carros, árvores ou pessoas.

Por outro lado, a segmentação de instâncias vai além ao não apenas segmentar a imagem neste contexto de "*Stuff*" e "*Things*", mas também distinguir cada instância individual de objetos dentro da mesma categoria. Isso significa que, em uma imagem com várias pessoas, a segmentação de instâncias identificaria e segmentaria cada pessoa separadamente, atribuindo uma máscara única a cada uma delas. A segmentação de instâncias é desafiadora porque requer a detecção correta de todos os objetos em uma imagem, ao mesmo tempo em que segmenta precisamente cada instância (HE *et al.*, 2017).

A segmentação panóptica combina aspectos da segmentação semântica e de instâncias para compreender a cena de forma completa. Ela busca tanto segmentar a imagem por categoria como também separar as instâncias individuais, proporcionando informações semânticas detalhadas sobre a imagem (CHUANG; ZHANG; ZHAO, 2023).

2.3 REDES NEURAIS PARA SEGMENTAÇÃO DE IMAGENS

Um neurônio em uma rede neural é uma unidade computacional fundamental que replica as características dos neurônios biológicos no cérebro humano. Ele processa informações ao receber entradas, calcula uma saída e a transmite a informação para outros neurônios na rede, contribuindo assim para a realização de tarefas.

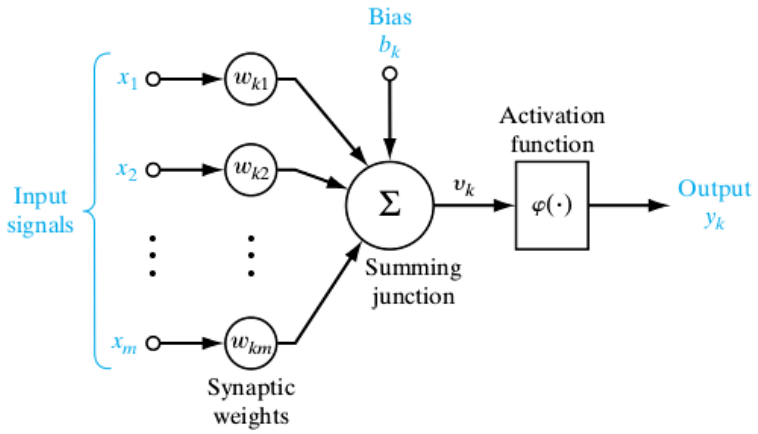
Cada neurônio recebe múltiplas entradas e cada entrada é ponderada por um peso específico, além disso, um neurônio pode ter também um viés (bias), que é um valor

constante adicionado às entradas antes de serem processadas. A operação realizada por um neurônio geralmente segue um processo em que cada entrada é multiplicada pelos pesos correspondentes e os produtos são somados junto com o termo de viés. Esse resultado da soma então é processado por uma função de ativação, que por sua vez determina a saída final do neurônio.

A função de ativação é fundamental para a capacidade do neurônio de aprender e representar relações não lineares entre as entradas e as saídas desejadas. Ela é importante para a capacidade da rede neural de aprender e representar relações complexas nos dados.Exemplos de funções de ativação comuns incluem a função sigmoide, a função ReLU (Rectified Linear Unit) e a função tangente hiperbólica.

A Figura 2 ilustra bem a operação de um neurônio:

Figura 2 – Estrutura de um neurônio artificial



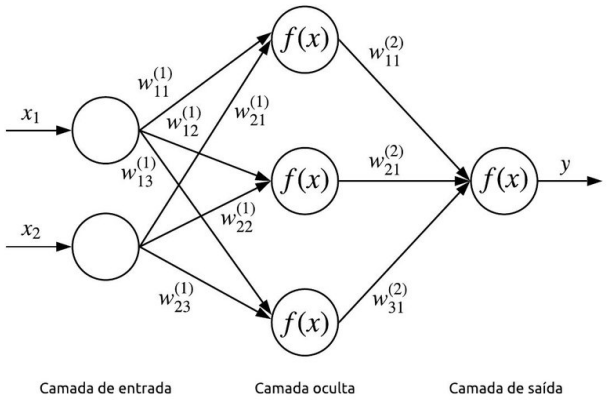
Fonte: (COELHO, 2017)

Uma rede neural é composta por um grande número de neurônios conectados. Esta conexão geralmente é dividida em camadas que são interconectadas entre si (KAUR; KAUR, 2014).

- **Camada de entrada:** Recebe as entradas iniciais para serem processadas pela rede neural.
- **Camadas ocultas:** As camadas ocultas são estágios intermediários entre a entrada e a saída em uma rede neural. Recebem as informações processadas na camada de entrada e são responsáveis por aprender as estruturas complexas nos dados, tornando as redes neurais uma ferramenta poderosa
- **Camada de saída:** Produz a saída final da rede neural após processar as informações geradas pelas camadas ocultas. Dependendo do tipo de tarefa que a rede está

realizando (como classificação, regressão, etc.), a camada de saída pode ter um ou vários neurônios.

Figura 3 – Camadas de uma Rede Neural



Fonte: (LISBOA *et al.*, 2019)

Cada camada na rede "refina"a representação em níveis mais abstratos. Esse tipo de aprendizado em camadas é comumente referido como aprendizado profundo ou deep learning, e essa capacidade tem sido um dos principais impulsionadores por trás do avanço significativo de sistemas de visão computacional e outras aplicações baseadas em redes neurais (GONZALEZ; WOODS, 2017).

No entanto, é importante considerar que os métodos baseados em redes neurais muitas vezes requerem conjuntos de dados grandes e de qualidade, e podem ser computacionalmente intensivos durante o processo de treinamento e inferência, enquanto os métodos clássicos são geralmente mais simples e rápidos.

2.3.1 Extração de *Features*

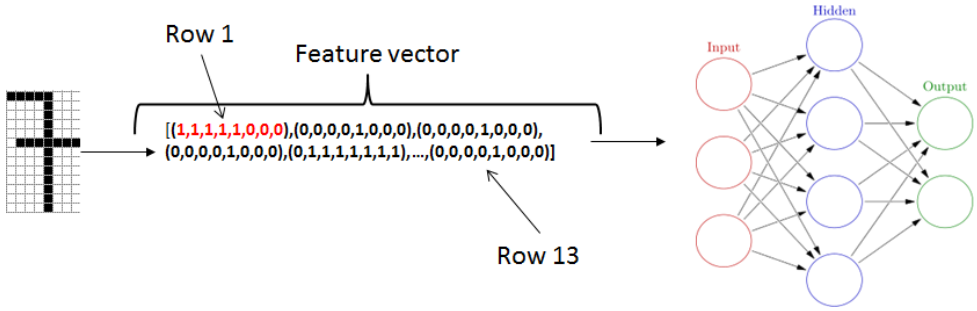
Uma *feature* refere-se a uma característica específica ou atributo identificável em uma imagem em que se tem como objetivo de rotular (GONZALEZ; WOODS, 2017).

A extração de *features* envolve isolar padrões que ajudam a identificar informações relevantes de uma imagem. Essas características podem ser globais, representando o conteúdo da imagem como um todo, ou locais, representando pequenos trechos. Para características globais, um único vetor de *features* é gerado por imagem, permitindo a comparação entre elas. Em contrapartida, as características locais são extraídas de diferentes partes da imagem, gerando múltiplas *features*.

A extração de características locais é dividida em dois passos: detecção e descrição. O detector identifica pontos ou regiões significativas, como cantos e bordas, enquanto a

descrição atribui informações quantitativas ou qualitativas às *features* detectadas. Embora esses dados geralmente não façam sentido visualmente, eles são essenciais para tarefas de alto nível, como reconhecimento de objetos, onde são organizados em vetores para facilitar a análise (HASSABALLAH; AWAD, 2016). A Figura 4 ilustra bem esta característica.

Figura 4 – Vetor de feature

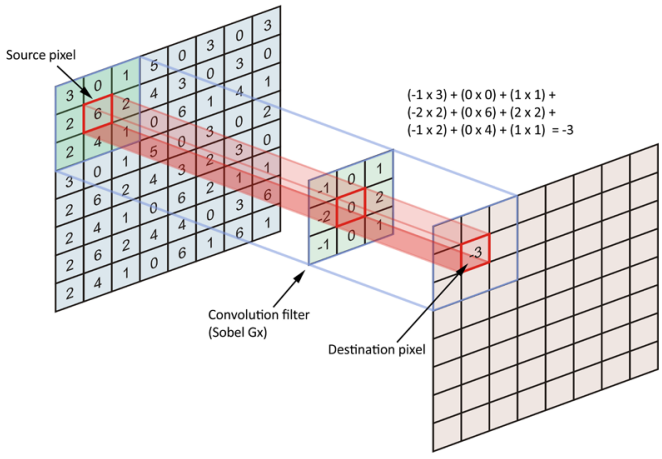


Fonte: (FEATURE..., s.d.)

2.3.2 Filtro Espacial e Convolução

A convolução neste contexto de imagens é uma operação espacial na qual cada pixel na imagem de saída é obtido como a soma ponderada dos pixels de entrada que são vizinhos a ele. A matriz de pesos utilizada nessa operação é conhecida como kernel de convolução, também chamado de filtro (WHAT..., s.d.). A Figura 5 apresenta uma representação de como ocorre este processo.

Figura 5 – Convolução



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

Desta forma, é possível compreender que a convolução calcula uma soma de produtos entre pixels e um conjunto de pesos de kernel. Essa operação é realizada em cada

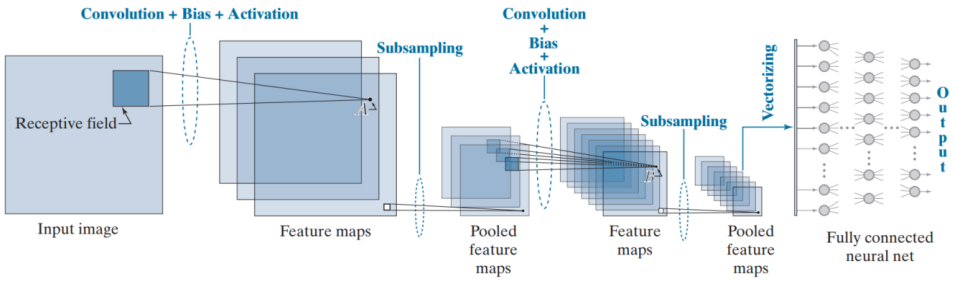
localização espacial na imagem de entrada. O resultado em cada localização (x, y) na entrada é um valor escalar (GONZALEZ; WOODS, 2017).

2.3.3 Redes Neurais Convolucionais - CNN

CNN é um tipo de rede neural especialmente projetada para processar dados em grade, como imagens e vídeos. Ela é chamada de "convolucional"devido à sua capacidade de aplicar operações de convolução, como mostradas anteriormente, ao longo do seu pipeline de execução. Devido a isso, elas são amplamente utilizadas em tarefas de visão computacional, como reconhecimento de padrões, detecção de objetos, segmentação e classificação de imagens, pois a partir da sua capacidade de aprender representações hierárquicas de *features* complexas elas conseguem ótimos resultados com um número de parâmetros menores quando comparados as ANN tradicionais (INDOLIA *et al.*, 2018).

O funcionamento básico de uma CNN envolve a passagem progressiva dos dados de entrada por uma série de camadas: camadas de convolução, camadas de agrupamento(ou pooling) e camadas totalmente conectadas. As duas primeiras realizam o processo de "encoding" das *features* que basicamente é a extração de características das imagens, enquanto a terceira realiza o processo de "decoding", ou seja, mapeia as características extraídas e as classifica para a saída final (YAMASHITA *et al.*, 2018). A Figura 6 representa o processo de execução de uma CNN:

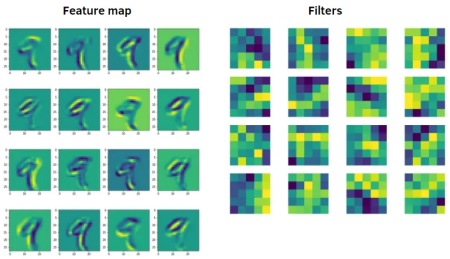
Figura 6 – CNN



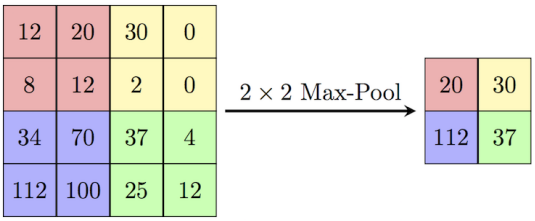
Fonte: (GONZALEZ; WOODS, 2017)

A partir da imagem, podemos entender que, as camadas convolucionais são as primeiras fases da arquitetura de uma CNN. Elas são compostas por uma quantidade de filtros com tamanho n×n que irão extrair mapas de características ao serem convoluídos com as imagens de entrada ou com saídas de outras camadas. A saída da convolução é denominada mapa de características. Esses mapas geralmente representam características gerais extraídas da imagem, como cor, borda, textura e forma, no entanto muitas vezes não apresentam representações visuais que façam sentidos de serem analisadas de forma isolada (SANTOS *et al.*, 2023). A Figura 7 apresenta uma visualização destes mapas de características juntamente com os diferentes tipos de filtros utilizados:

Figura 7 – Mapas e Filtros



Fonte: (ANELLO, 2021)



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

Após as camadas convolucionais, geralmente são empregadas camadas de pooling. Esse tipo de camada reduz a dimensão espacial dos mapas gerados por camadas anteriores, reduzindo o volume de dados e consequentemente o custo computacional. Estas operações utilizam uma janela deslizante de tamanho n×n no mapa de características e para cada passo realizado, o valor máximo (ou médio) daquela janela é retirado e utilizado como saída da camada. Na Figura 7 essa operação é ilustrada (SANTOS *et al.*, 2023).

As camadas totalmente conectadas (fully-connected layers - FCs) geralmente aparecem ao final da arquitetura, após algumas camadas convolucionais. Nesta fase a rede neural já extraiu as características importantes e reduziu sua dimensionalidade, restando agora utilizar desta informação para gerar a saída final do objetivo proposto. A execução desta fase se dá pela operação que transforma a matriz dos mapas de características em uma matriz unidimensional (camada *flatten*) que permite adaptar a informação gerada na camada de pooling para uma uma fully connected network tradicional (SANTOS *et al.*, 2023).

3 METODOLOGIA

3.1 ESTRUTURA DO TRABALHO

Este trabalho é estruturado em cinco etapas bem definidas. Inicialmente, foi realizada uma revisão teórica da literatura para compreender os conceitos fundamentais relacionados ao processamento digital de imagens, redes neurais, detecção e segmentação de imagens. Em seguida, é conduzida uma análise dos trabalhos relacionados, com foco na família de redes neurais baseadas em regiões, para avaliar suas arquiteturas, aspectos técnicos e características evolutivas. Posteriormente, é realizada uma aplicação técnica da R-CNN com *Selective Search*, permitindo uma compreensão empírica dos conceitos teóricos discutidos e uma avaliação do avanço nas técnicas de proposição de regiões. Os resultados obtidos nesta demonstração são analisados para validar os conceitos teóricos investigados. Finalmente, são apresentadas conclusões sobre os resultados alcançados e discussões sobre melhorias na metodologia utilizada, além de sugerir direções para trabalhos futuros nesse campo.

O estudo concentrou-se exclusivamente na aplicação da *R-CNN*, devido à elevada demanda computacional e à complexidade que a inclusão de outras abordagens traria para este trabalho.

Esta seção contém a análise das principais redes baseadas em regiões: R-CNN, *Fast* R-CNN, *Faster* R-CNN e *Mask* R-CNN que são os objetos de estudo deste trabalho. Serão abordadas seus aspectos evolutivos, destacando as mudanças e melhorias ao longo do tempo demonstrando como cada abordagem superou limitações anteriores e contribuiu para avanços na detecção e segmentação de objetos.

3.2 R-CNN

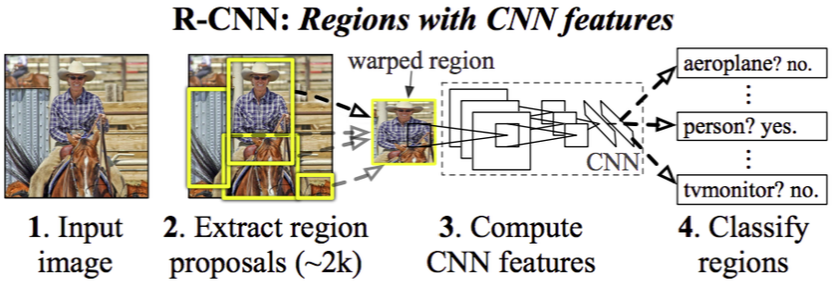
Este artigo explora a integração entre a classificação de imagens e a detecção de objetos, destacando o uso das Redes Neurais Convolucionais (CNNs) para aprimorar significativamente o desempenho na detecção de objetos em comparação com métodos mais simples, como o HOG.

A detecção de objetos envolve a identificação precisa dos mesmos em uma imagem, o que a diferencia da tarefa de classificação. Uma abordagem comum é tratar a localização como um problema de regressão, porém, isso nem sempre se mostra eficaz na prática. Uma alternativa explorada é a criação de um detector de janela deslizante que processa sequencialmente cada área da imagem ao deslizar uma "janela"de pixels pela imagem. Entretanto, um desafio técnico significativo surge em razão do volume de áreas geradas, o que torna computacionalmente exigente realizar a localização precisa dos objetos dentro da imagem.

Para resolver o problema de lidar com um grande número de regiões, Ross Girshick

et al. propuseram um método que utiliza a busca seletiva para extrair apenas 2000 regiões da imagem, chamadas de "propostas de região". Isso permite trabalhar com um conjunto mais gerenciável em vez de tentar classificar um grande número de regiões.(GANDHI, 2018)

Figura 8 – Arquitetura R-CNN



Fonte: (**R-CNN**)

O algoritmo de busca seletiva desempenha um papel fundamental na detecção de objetos, especialmente ao gerar 2000 propostas de região(Item 2 da imagem). Estas propostas, ao serem definidas, são então inseridas em uma rede neural convolucional (CNN), resultando em um vetor de características de 4096 dimensões como saída. A CNN atua como um poderoso extrator de características, cujas features extraídas são usadas por um SVM (*Support Vector Machine*) no processo final de classificação para determinar a presença do objeto dentro da região proposta.

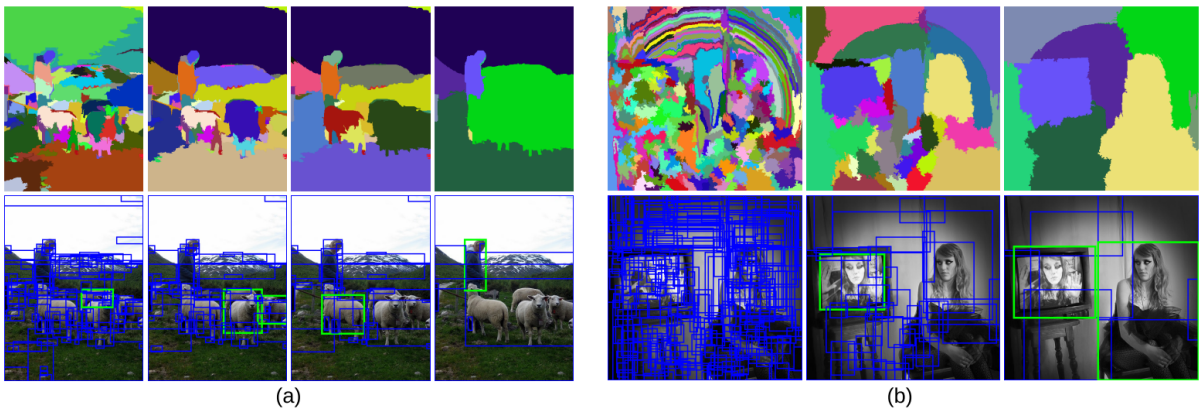
Além da detecção da presença de objetos, o algoritmo também calcula de forma auxiliar quatro valores de offset, essenciais para o ajuste preciso da região proposta. Essa correção é crucial para garantir que a delimitação do objeto dentro da região proposta seja precisa (GANDHI, 2018).

R-CNN demonstra uma média de precisão (mAP) de 53.7% no PASCAL VOC 2010

Embora o R-CNN tenha alcançado alta qualidade nos resultados da época comparado às outras abordagens, ele apresenta algumas desvantagens notáveis. Por exemplo, o treinamento em um pipeline de vários estágios é lento e difícil devido ao fato de cada estágio individual precisar ser treinado individualmente. Além disso, para treinar o classificador SVM e o regressor BBox, são necessários mais recursos computacionais e tempo de processamento. Estas características de sua arquitetura tornam os testes lentos, pois as features da CNN precisam ser extraídas para cada proposta de objeto em cada imagem de teste, sem compartilhamento de computação. Esses problemas com o R-CNN inspiraram o desenvolvimento de outras técnicas, levando ao surgimento de estruturas de detecção aprimoradas, como o *Fast* R-CNN e o *Faster* R-CNN (HAFIZ; BHAT, 2020).

3.2.1 *Selective Search*

Figura 9 – *Selective Search*



Fonte: (UIJLINGS *et al.*, 2013)

O *Selective Search* funciona ao supersegmentar uma imagem usando um algoritmo de superpixel baseado no método de Felzenszwalb. Este método é um algoritmo de segmentação que agrupa pixels em regiões com base na similaridade de cor e textura. Ele é eficiente e hierárquico, permitindo segmentação em diferentes níveis de granularidade. O algoritmo constrói um grafo onde cada pixel é um nó, com arestas que conectam nós adjacentes, pesadas pela diferença de cor, formando assim as regiões (UIJLINGS *et al.*, 2013).

Ao trocar abordagens tradicionais de janelas deslizantes pelo *Selective Search*, as vantagens em eficiência e precisão tornam-se evidentes. Enquanto as janelas deslizantes analisam exaustivamente todas as regiões da imagem, gerando inúmeras caixas delimitadoras e exigindo alto consumo de tempo e processamento, o *Selective Search* otimiza esse processo agrupando regiões semelhantes em cor e textura. Isso reduz significativamente o número de regiões propostas, resultando em menor custo computacional e maior precisão na localização de objetos relevantes.

O algoritmo de *Selective Search* começa com a subsegmentação inicial da imagem, dividindo-a em pequenas regiões. Em seguida, aplica um algoritmo *Greedy* para agrupar recursivamente regiões semelhantes, combinando as duas mais similares em cada iteração, até formar regiões maiores. Por fim, as regiões segmentadas são utilizadas para gerar propostas de localização de objetos, refinando a busca seletiva de maneira eficiente.

As definições do agrupamento baseiam-se em cálculos de similaridade propostos pelo algoritmo, que combinam linearmente as similaridades de cor, textura, tamanho e forma, resultando em uma medida abrangente que orienta o processo de agrupamento das regiões. Neste contexto r_i e r_j representam o par de regiões comparadas para o cálculo da

similaridade s (UIJLINGS *et al.*, 2013).

$$s(r_i, r_j) = s_{\text{cor}}(r_i, r_j) + s_{\text{textura}}(r_i, r_j) + s_{\text{tamanho}}(r_i, r_j) + s_{\text{forma}}(r_i, r_j) \tag{1}$$

- **Similaridade de Cor:** Um histograma de 25 intervalos é computado para cada canal (RGB), resultando em um descritor de 75 dimensões $C_i = \{c_i^1, \cdots, c_i^n\}$. A similaridade é medida pela interseção do histograma, quantificando a sobreposição entre as distribuições de cor de diferentes regiões.

$$s_{cor}(r_i, r_j) = \sum_{k=1}^{n} min(c_i^k, c_j^k) \tag{2}$$

- **Similaridade de Textura:** Derivadas gaussianas são extraídas em 8 orientações para cada canal, gerando histogramas de 10 intervalos e resultando em um descritor de 240 dimensões $T_i = \{t_i^1, \cdots, t_i^n\}$. A similaridade é medida novamente pela interseção do histograma.

$$s_{textura}(r_i, r_j) = \sum_{k=1}^{n} min(t_i^k, t_j^k) \tag{3}$$

- **Similaridade de Tamanho:** O algoritmo prioriza a fusão de regiões menores, evitando que grandes regiões dominem e absorvam as menores, o que é essencial para manter a diversidade de propostas.

$$s_{tamanho}(r_i, r_j) = 1 - \frac{area(r_i) + area(r_j)}{area(im)} \tag{4}$$

- **Similaridade de Forma:** Regiões são consideradas compatíveis se se encaixam, ou seja, podem preencher lacunas nas propostas regionais que é a área ao redor de r_i e r_j definida como BB_{ij}. Regiões que não se tocam não devem ser combinadas, ajudando a manter a integridade das propostas.

$$s_{forma}(r_i, r_j) = 1 - \frac{area(BB_{ij}) - area(r_i) - area(r_j)}{area(im)} \tag{5}$$

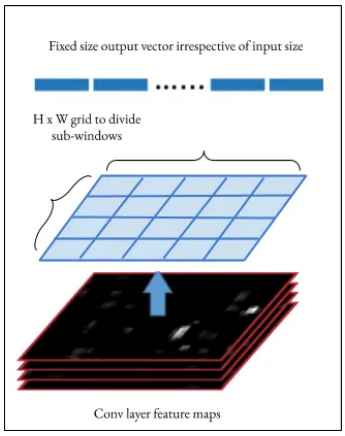
3.3 *FAST* R-CNN

Olhando pela perspectiva de que a R-CNN é computacionalmente ineficiente, o mesmo autor da R-CNN resolveu algumas das desvantagens para construir um algoritmo aprimorado. A abordagem do *Fast* R-CNN é semelhante ao algoritmo anterior, a principal diferença consiste em em deixar de alimentar as propostas de região à CNN e alimentar a imagem de entrada inteira diretamente à CNN para gerar um mapa de características convolucionais (GANDHI, 2018).

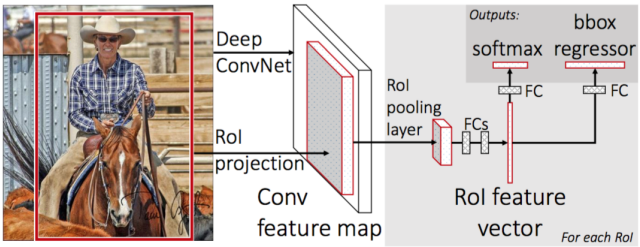
A partir deste mapa, na camada de *RoI Pooling Layer* são identificadas propostas de região e divididas em uma grade fixa para ser processada. Ou seja, considere uma região de interesse definida por suas coordenadas em relação à imagem original. Essa região é dividida em uma grade de células, geralmente de tamanho fixo. Cada célula na grade é mapeada para um valor na saída da camada por meio da operação de pooling, representando uma região da ROI. Isso é feito para todas as células da grade, resultando em uma saída de tamanho fixo independentemente do tamanho da ROI original. A Figura 10 representa bem este processo.

A camada de RoI pooling extrai um vetor de características de comprimento fixo do mapa de características. Cada vetor de características é alimentado em uma sequência de camadas totalmente conectadas (FC) que se ramificam em duas camadas de saída: uma que produz estimativas de probabilidade *softmax* sobre K classes de objetos, além de uma classe *background* geral, e outra camada que gera as coordenadas da *bounding box* para cada uma das classes de objetos (GIRSHICK, 2015). A Figura 10 ilustra esse processo.

Figura 10 – ROI Pooling Layer e Arquitetura *Fast* R-CNN



Fonte: (ANANTH, 2019)



Fonte: (GIRSHICK, 2015)

Como mencionado anteriormente, a razão pela qual o *Fast* R-CNN é mais rápido que o R-CNN é devido ao fato de não ser necessário alimentar 2000 propostas de região na rede neural convolucional para cada imagem. Em vez disso, a operação de convolução é realizada apenas uma vez por imagem e um mapa de características é gerado a partir dela. Isso resulta em uma melhoria significativa nos resultados e na eficiência (GANDHI, 2018).

Na arquitetura VGG16, o *Fast* R-CNN processa imagens de forma significativamente mais rápida do que o R-CNN, alcançando uma velocidade até 146 vezes maior sem o uso do SVD truncado e até 213 vezes mais rápido com essa técnica. Além disso, o tempo de treinamento é drasticamente reduzido, passando de 84 horas para apenas 9,5 horas, o que representa uma diminuição de nove vezes na duração do treinamento.

Em resumo, o *Fast* R-CNN representa um avanço notável em eficiência comparado ao R-CNN. Ao otimizar a alimentação de propostas de região para a CNN, ele acelera

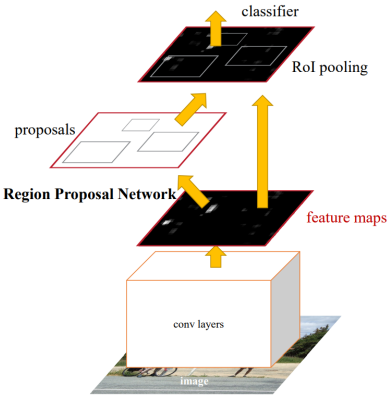
significativamente o treinamento e teste do modelo, reduzindo drasticamente o tempo de processamento. Essa melhoria na velocidade foi crucial para aplicações em tempo real e projetos que dependiam de maior eficiência computacional.

3.4 *FASTER* R-CNN

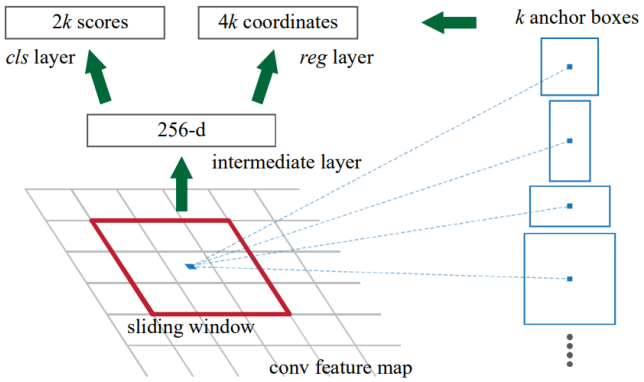
Faster R-CNN surge a partir do contexto evolutivo da R-CNN e *Fast* R-CNN. O diferencial desta abordagem está na introdução da RPN (*Region Proposal Network*) de maneira integrada na arquitetura, permitindo o treinamento de ponta a ponta. Anteriormente as abordagens exigiam uma etapa separada para gerar propostas de região, o que tornava o processo mais lento e computacionalmente custoso.

O autor traz uma analogia interessante com os mecanismos de atenção em redes neurais. Estes mecanismos são componentes que permitem que o modelo "preste atenção"em partes específicas dos dados, com pesos variáveis dependendo da importância dessas partes para a tarefa em questão. Olhando para o caso da *Faster* R-CNN vemos que esta técnica é utilizada para descrever como o RPN direciona a atenção do *Fast* R-CNN para as regiões importantes na imagem, evitando a necessidade de processar toda a imagem simultaneamente. Isso contribui para uma melhoria na eficiência e no desempenho geral do sistema de detecção de objetos. A Figura 11 apresenta como a RPN está integrada na arquitetura da *Faster* R-CNN.

Figura 11 – Arquitetura *Faster* R-CNN e RPN



Fonte: (REN *et al.*, 2015)



Fonte: (REN *et al.*, 2015)

A partir da imagem acima, podemos perceber que *Faster* R-CNN é composto por dois módulos principais. O primeiro módulo é uma rede neural totalmente convolucional (FCN) que gera propostas de região na imagem para possíveis objetos de interesse. O segundo módulo é o detector *Fast* R-CNN, que utiliza as propostas de região geradas pelo RPN para identificar e categorizar objetos dentro dessas áreas. Todo o sistema é integrado em uma única rede neural, permitindo que o RPN e o *Fast* R-CNN trabalhem em conjunto para realizar a detecção de objetos de forma eficiente (REN *et al.*, 2015).

3.4.1 *Region Proposal Network* - RPN

Como explicado anteriormente a RPN é uma rede totalmente convolucional que gera propostas com diferentes escalas e proporções. Ela trabalha no mapa de características de saída retornado da última camada convolucional compartilhada com o *Fast* R-CNN. Uma janela deslizante de tamanho especifico passa pelo mapa de características, e para cada janela, várias propostas de região candidatas são geradas (GAD, 2020). A Figura 11 ilustra bem este processo.

Essas propostas não são as propostas finais, pois serão filtradas com base em sua probabilidade de conter um objeto ou de ser um *background*. Para realizar esta definição elas passam por um classificador e um regressor, introduzindo o conceito de âncoras que representam pontos centrais em uma janela deslizante. O classificador avalia a probabilidade de uma proposta conter o objeto de interesse, enquanto o regressor ajusta suas coordenadas delimitadoras deste objeto.

No artigo, o autor apresenta uma configuração 9 âncoras por pixel, cada uma com uma escala distinta. A presença de âncoras em múltiplas escalas resulta em uma "Pirâmide de âncoras", uma abordagem mais eficiente do que as "Pirâmides de Filtros"usadas em métodos anteriores como o Multi-Box. Essa estratégia não apenas reduz o tempo de processamento, mas também aumenta a robustez do algoritmo contra translações, tornando-o invariante a essas transformações (KARMARKAR, 2018).

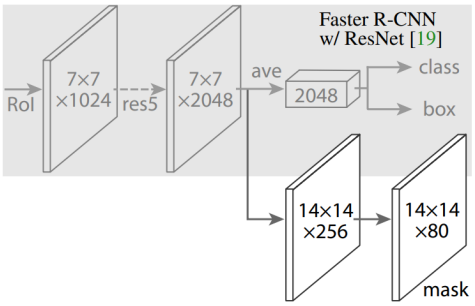
3.5 *MASK* R-CNN

O *Mask* R-CNN é uma extensão do *Faster* R-CNN com um módulo adicional para gerar máscaras de segmentação de alta qualidade para cada imagem. Isso nos permite segmentar pixel a pixel cada objeto e também extrair cada objeto separadamente sem nenhum fundo (o que não é possível com a segmentação semântica)(RESEARCH, 2022). Este modelo adiciona apenas uma pequena sobrecarga ao *Faster* R-CNN, rodando a 5 fps. Além disso, o *Mask* R-CNN também permite generalizar para outras tarefas, por exemplo, permitindo estimar poses humanas no mesmo *framework*.

O *Mask* R-CNN é conceitualmente simples: enquanto o *Faster* R-CNN gera dois resultados para cada objeto candidato, um rótulo de classe e um **bounding-box**, o *Mask* R-CNN vai além ao adicionar um terceiro ramo que produz a máscara do objeto. Além disso, a saída adicional da máscara é distinta das outras saídas, o que faz a extração de um layout espacial muito mais detalhado do objeto. A Figura 12 ilustra esse novo módulo.

Este novo módulo adicional tem o objetivo de gerar uma máscara binária, ou seja, é uma máscara que indica se um determinado pixel faz parte de um objeto ou não. Esta fase (em branco na imagem acima) é apenas uma FCN que tem como entrada um feature map da CNN e como saída uma máscara binária como uma matriz m x m com 1 em todas as localizações onde o pixel pertence ao objeto e 0 em outros lugares.

Figura 12 – Ramo da máscara



Fonte: (HE *et al.*, 2017)

Comparando o *Mask* R-CNN dentro do contexto de detecção de objetos com o estado da arte na época para o *dataset* COCO, vemos que o modelo apresenta um desempenho superior ao ultrapassar as variantes base de modelos anteriores. Utilizando ResNet-101-FPN e, posteriormente, ResNeXt-101-FPN, o *Mask* R-CNN apresentou melhorias significativas, alcançando uma margem de 3.0 pontos de AP sobre as melhores entradas de modelos únicos anteriores.

É importante destacar que, para obter os resultados do *Mask* R-CNN, os autores implementaram algumas modificações no pipeline do *Faster* R-CNN, incluindo a substituição do ROIPool pelo ROIAlign e a utilização do FPN.

3.5.1 RoI Align

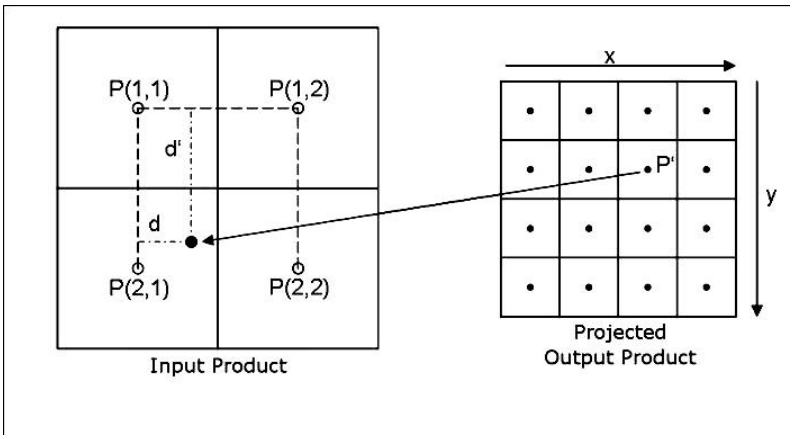
Dentro do contexto da *Mask* R-CNN é predita uma máscara m $\times$ m para cada região de interesse permitindo a preservação do layout espacial do objeto sem reduzi-lo a uma representação de vetor que perde dimensões espaciais. Esse comportamento pixel a pixel requer que as características RoI estejam bem alinhadas para manter a correspondência espacial corretamente. No entanto, o processo executado pelo pooling do RoI está sujeito a quantizações que podem introduzir desalinhamentos entre a RoI e as características extraídas.

Para solucionar esta questão foi proposto uma abordagem de RoI Align. Esta mudança busca evitar qualquer tipo de quantização existente no RoI Pooling. Para isso, é utilizado interpolação bilinear para calcular os valores exatos das características de entrada em quatro locais regularmente amostrados em cada subdivisão da RoI (HE *et al.*, 2017).

A Equação (6) e a Figura 13 ilustram como ocorre esta amostragem e como é realizado o cálculo da interpolação. E que $(y_1, x_1), (y_2, x_2), (y_3, x_3), (y_4, x_4)$ são as coordenadas dos quarto pontos amostrados de cada área e $Q_{11}, Q_{12}, Q_{21}, Q_{22}$ são os valores de cada área.

$$P \approx \frac{y_2 - y}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1}Q_{11} + \frac{x - x_1}{x_2 - x_1}Q_{21}\right) + \frac{y - y_1}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1}Q_{12} + \frac{x - x_1}{x_2 - x_1}Q_{22}\right) \quad (6)$$

Figura 13 – Interpolação Bilinear



Fonte: (RESAMPLING. . . , s.d.)

3.5.2 Feature Pyramid Network - FPN

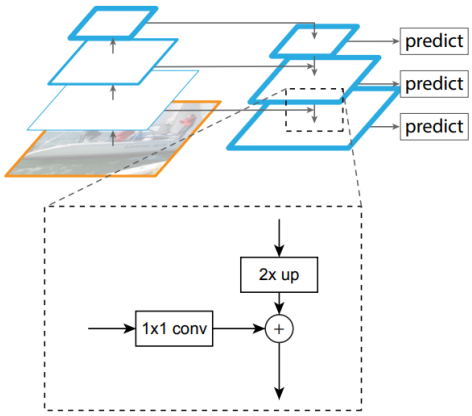
A FPN é um extrator de características projetado especificamente em um formato de pirâmide, buscando um equilíbrio entre precisão e eficiência computacional. Ele gera várias camadas de mapas de características (em múltiplas escalas). Este método recebe uma imagem de tamanho arbitrário como entrada e produz mapas de características de tamanho proporcional em vários níveis independente das arquiteturas utilizadas.

A construção da pirâmide envolve um caminho *bottom-up*, um caminho *top-down* além conexões laterais, como podemos ver na Figura 14. O caminho de baixo para cima é a rede convolucional usual para extração de características. A cada nível da pirâmide a resolução espacial diminui, no entanto o valor semântico de cada camada aumenta devido a detecção de estruturas mais complexas.

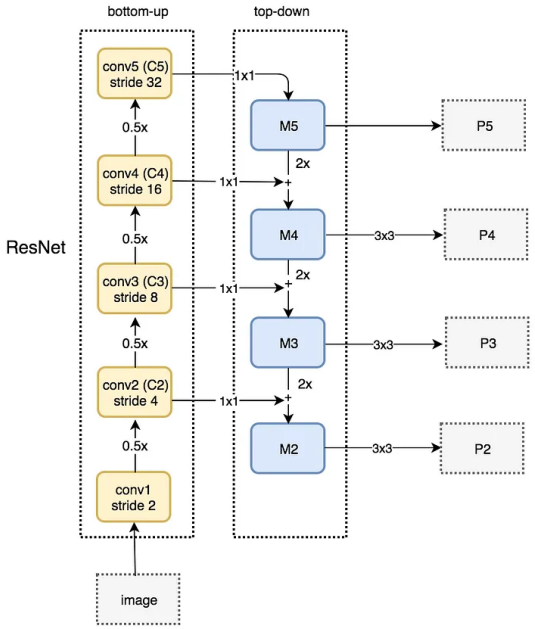
Já o caminho de cima para baixo na FPN simula características de resolução mais altas e semanticamente mais fortes, porém, as localizações dos objetos não são precisas após todo o processo de *downsampling* e *upsampling*. Para solucionar esta questão, são adicionadas conexões laterais entre as camadas reconstruídas e os mapas de características correspondentes para auxiliar o detector a prever as localizações com mais precisão e facilitar o treinamento (LIN *et al.*, 2016). A Figura 14 ilustra bem este processo para uma estrutura com rede da ResNet.

Em resumo, a FPN utiliza o caminho de *bottom-up* para extrair características em cada nível da pirâmide, aumentando o valor semântico conforme avança. O caminho de

Figura 14 – Conexão Lateral e Arquitetura FPN



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



Fonte: (HUI, 2018)

top-down restaura a resolução com informações semânticas ricas. Além disso, as conexões laterais são essenciais para adicionar informações espaciais mais precisas dos objetos de volta, completando assim o processo de detecção e localização de forma mais eficaz e precisa.

4 EXPERIMENTOS E RESULTADOS

4.1 ESTRUTURA DOS EXPERIMENTOS

Os experimentos desenvolvidos neste trabalho visaram implementar e demonstrar os principais conceitos da R-CNN a partir de uma adaptação do seu código, destacando o avanço nas propostas de região. O desenvolvimento foi realizado no *Google Colab Pro*, devido à necessidade de mais memória e GPUs para um processamento eficiente. Utilizou-se uma amostra do *dataset* COCO, focando em uma única classe, com o objetivo de demonstrar a abordagem, sem a intenção de reproduzir os resultados do artigo original.

O trabalho visou concentrar-se apenas na aplicação da *R-CNN* devido a alta exigência computacional necessária e também devido a complexidade que seria adicionada ao trabalho caso o objetivo fosse analisar as outras abordagens.

Para o desenvolvimento do modelo, foram utilizadas principalmente as bibliotecas *TensorFlow* e *Keras*, e sua implementação está disponível no Google Colab através do link: `https://colab.research.google.com/drive/1PXxLaAai5dtAzJCbyUQ_ILKkM32ajfAD#scrollTo=e7H749mWpYYE`. As principais referências para o código desenvolvido se baseiam em (THAKUR, 2019).

4.2 CONJUNTO DE DADOS E PRÉ PROCESSAMENTO

O conjunto de dados utilizado no modelo foi considerado apenas da categoria 4 do COCO *dataset*. Esta categoria é composta apenas com imagens de motocicletas em contextos variados. Podemos perceber algumas características deste subconjunto:

- Todas as imagens são coloridas
- Podem existir um ou mais objetos de interesse em cada imagem
- Podem existir objetos de interesse oblíquos (parcialmente visíveis)
- As imagens possuem contextos variados (nem sempre serão motos em uma estrada, por exemplo)
- São imagens reais e estão sujeitas as variações de luminosidade, contraste, ruídos, etc

Dentro desse contexto, cada imagem possui características distintas que afetam sua qualidade e interpretação, como brilho, contraste e cores.

O brilho se refere à quantidade geral de luz presente na imagem. Imagens com brilho excessivo podem apresentar áreas superexpostas, enquanto aquelas com brilho insuficiente podem ficar escuras demais. Esse fator influencia diretamente a visibilidade dos detalhes e a percepção global da imagem.

Figura 15 – Imagens contextos variados

O contraste, por sua vez, refere-se à diferença entre as áreas claras e escuras. Um alto contraste pode tornar as imagens mais vivas e definidas, mas também pode ocultar detalhes em sombras ou áreas muito brilhantes. Já um baixo contraste tende a gerar uma aparência mais plana, com detalhes menos perceptíveis.

Por fim, as cores variam em termos de saturação, matiz e intensidade. Em imagens coloridas, é fundamental equilibrar essas variações para garantir uma representação precisa e visualmente agradável.

A Figura 16 apresenta a análise do histograma de brilho e a análise da curva normal do contraste das imagens.

Figura 16 – Análise do contexto de brilho e contraste das imagens

Imagem - 578962

Imagem - 150769

Podemos observar que, no contexto dessas duas imagens, a primeira aparenta ser

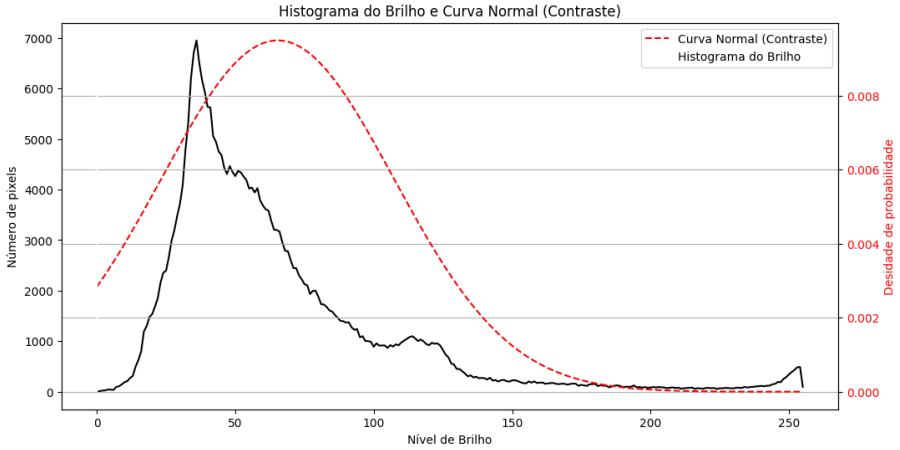


Figura 17 – Análise imagem - 578962

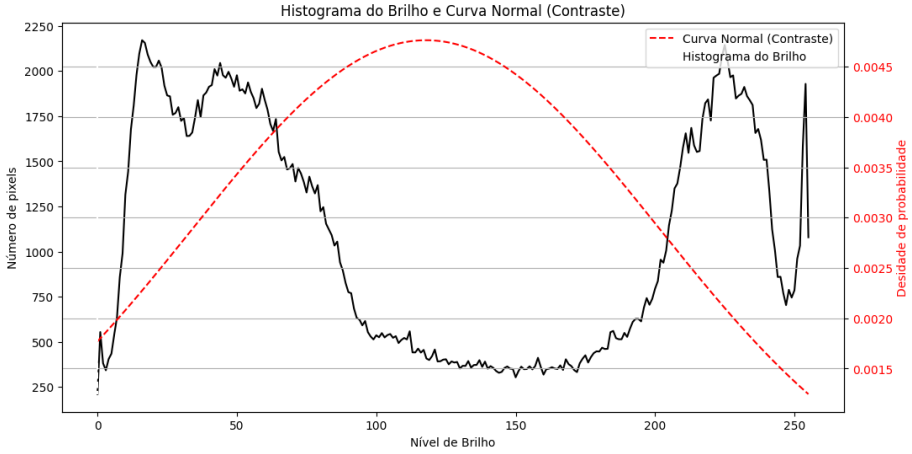


Figura 18 – Análise imagem - 150769

visualmente mais "escura", o que podemos confirmar analisando a intensidade dos pixels presentes na imagem. O gráfico mostra valores mais baixos para cada pixel, indicando um nível de brilho menor.

Por outro lado, a segunda imagem exibe uma distribuição mais equilibrada entre pixels de baixa e alta intensidade, mas com poucos pixels de intensidade média. Isso sugere a presença de regiões muito claras e muito escuras na mesma imagem. Essa variação é visível na imagem, onde uma parte está na sombra e outra está iluminada pelo sol.

Outra análise que também é apresentada na Figura 19 são os histogramas para cada canal de cor, indicando uma possível predominância na imagem.

Podemos perceber pelos histogramas que na imagem em específico há um a concentração de pixels do canal 'azul' com maiores intensidades e uma concentração de pixels do canal 'vermelho' com baixas intensidades. Isso reflete em uma imagem com a predominância da cor azul, o que é nitidamente visível devido ao contexto de iluminação do local onde foi capturada a imagem.

Figura 19 – Análise dos canais de cor da imagem

Figura 20 – Imagem Original

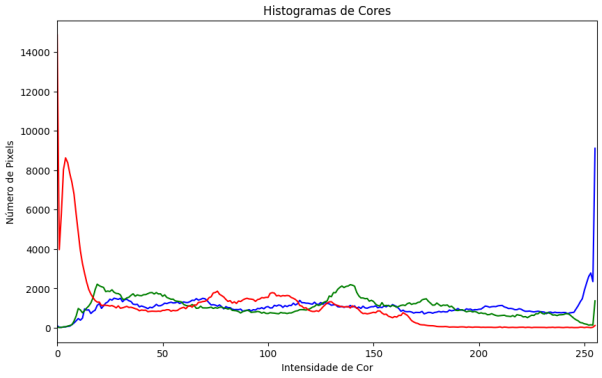


Figura 21 – Análise histograma de cores

4.2.1 Correções nas imagens

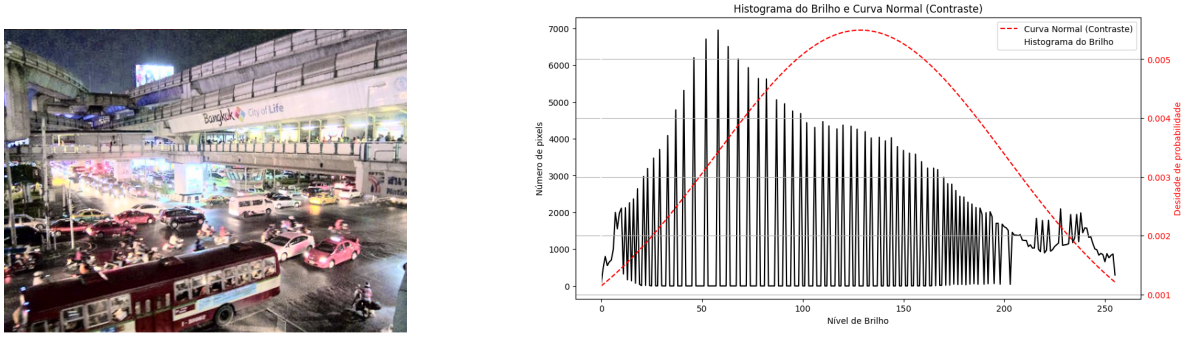
O *Selective Search* utiliza características das imagens para propor regiões, tornando crucial sua otimização para aumentar a relevância dessas sugestões. O algoritmo avalia componentes como cor, agrupando áreas com tonalidades semelhantes, e textura, identificando padrões repetitivos para diferenciar regiões. Também considera o tamanho, combinando pequenas e grandes áreas em propostas mais significativas, além de analisar a forma para agrupar regiões com contornos semelhantes.

Para melhorar a eficácia do *Selective Search*, podem ser adotadas técnicas de pré-processamento, como equalização de histograma e filtragem, que ressaltam características importantes e aprimoram a segmentação. Nesse contexto, foram desenvolvidas funções específicas para apoiar esse processo de otimização.

- **Equalização de histograma brilho canal Y (imagem YUV):** A função realiza a equalização de histograma especificamente no canal de luminosidade de uma imagem colorida. Primeiramente inicia convertendo a imagem colorida do espaço BGR para YUV, separando o brilho (canal Y) das informações de cor (canais U e V). Em seguida, aplica equalização de histograma ao canal Y para melhorar o contraste. Por fim, a imagem é convertida novamente para o espaço BGR. O benefício desta função é que ela melhora o contraste e a visibilidade dos detalhes na imagem, focando na luminosidade sem alterar as informações de cor, resultando em uma imagem com um brilho mais equilibrado e detalhes mais claros.

- **Equalização de histograma dos canais de cor** A função realiza a equalização de histograma em cada canal de cor individualmente para corrigir desbalanceamentos de cor em uma imagem. A função separa a imagem em canais de cor, equaliza o histograma de cada um e os reúne em uma única imagem. O benefício é que ela pode corrigir desbalanceamentos de cor e melhorar o contraste geral da imagem ao

Figura 22 – Equalização de histogramas do canal Y

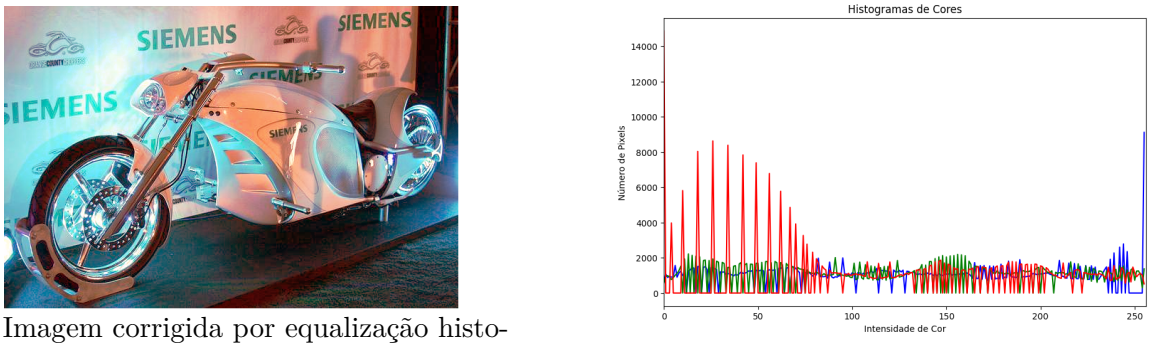


(a) Imagem equalizada

(b) Histograma equalizado

tratar cada canal de cor de forma independente, resultando em uma imagem mais equilibrada e visualmente mais agradável.

Figura 23 – Correção dos canais de cor da imagem



(a) Imagem corrigida por equalização histograma de cada canal de cor

(b) Análise histograma imagem corrigida

- **Filtro bilateral para reduzir ruido** A função aplica um filtro bilateral para suavizar a imagem enquanto preserva as bordas e os detalhes importantes. O filtro é ajustado por três parâmetros: d, que define o diâmetro dos pixels vizinhos usados para o cálculo, influenciando a área de aplicação do filtro; *sigma_color*, que controla a suavização com base na diferença de cor entre pixels, determinando quão suavemente as cores próximas são misturadas; e *sigma_space*, que define a suavização com base na distância espacial entre pixels, afetando a quantidade de influência dos pixels vizinhos. O resultado é uma imagem suavizada onde o ruído é reduzido, mas as bordas e detalhes importantes são mantidos.

4.3 R-CNN

4.3.1 Preparo Dados Treinamento

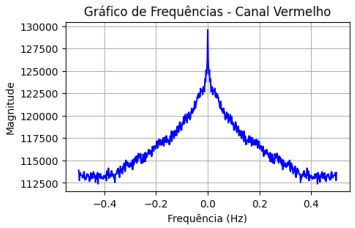
Na R-CNN foi utilizado o mecanismo de *Selective Search* para propor as regiões de interesse. A implementação deste algoritmo é bastante direta com o uso da biblioteca

Figura 24 – Imagem com filtro bilateral

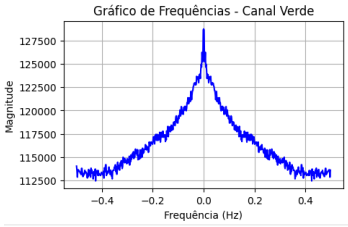
(a) Imagem original

(b) Imagem suavizada

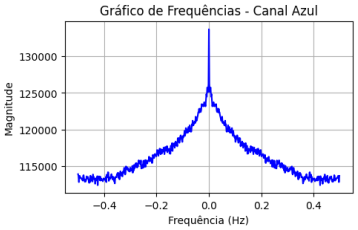
Figura 25 – Frequências imagem original



Canal Vermelho

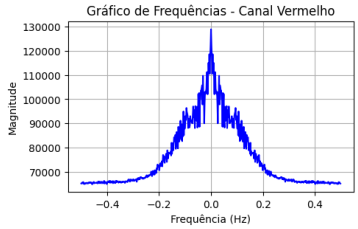


Canal Verde

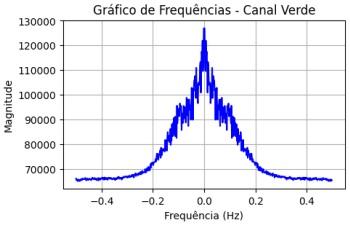


Canal Azul

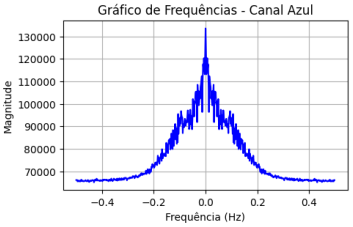
Figura 26 – Frequências imagem suavizada



Canal Vermelho



Canal Verde



Canal Azul

OpenCV, conforme ilustrado no código a seguir:

```
ss = cv2.ximgproc.segmentation.createSelectiveSearchSegmentation()
ss.setBaseImage(image_path)
ss.switchToSelectiveSearchFast()
rects = ss.process()
```

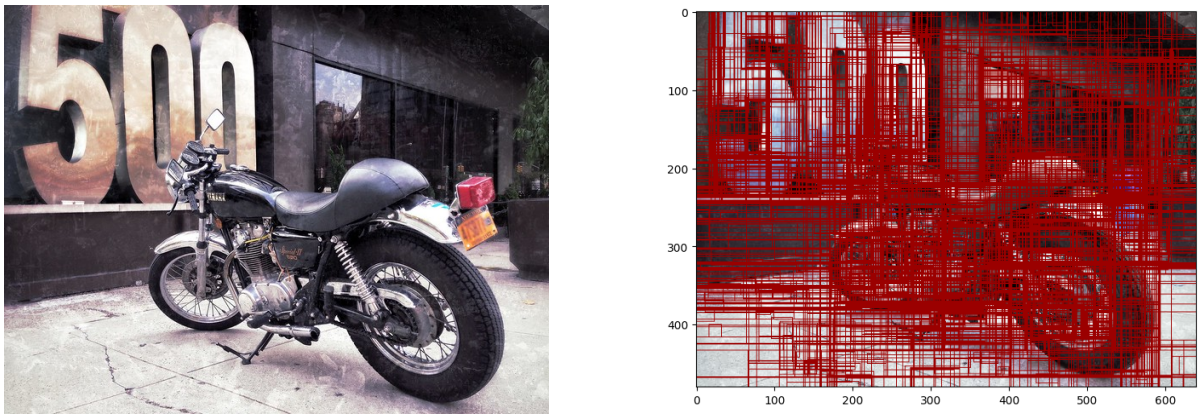
Este código inicializa o módulo de segmentação do *Selective Search*, define a imagem base e ativa o modo rápido com *switchToSelectiveSearchFast()* para gerar propostas de regiões de interesse de forma eficiente. Alternativamente, *switchToSelectiveSearchQuality()* oferece resultados de maior qualidade, mas com maior tempo de processamento.

O *Selective Search* funciona segmentando a imagem em regiões menores baseadas

em similaridades principalmente de cor, textura, tamanho e forma. Em seguida, essas regiões são agrupadas de forma hierárquica para formar as propostas. No modo rápido, o algoritmo realiza menos iterações de agrupamento e utiliza parâmetros ajustados para acelerar esse processo, resultando em um conjunto de propostas de regiões que ainda são eficazes para detecção de objetos, mas geradas em um tempo significativamente menor.

Finalmente após este processo, as regiões de interesse são geradas e armazenadas na variável *rects*. A Figura 27 apresenta a visualização das regiões propostas pelo algoritmo. É perceptível a grande quantidade de regiões para uma única imagem.

Figura 27 – Regiões Propostas *Selective Search*



(a) Imagem original

(b) Imagem com regiões propostas

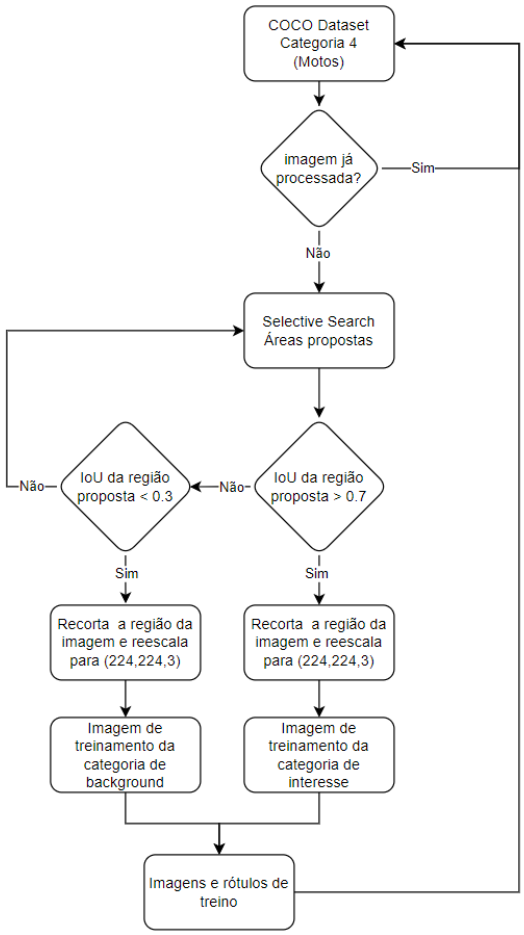
A partir deste ponto, os dados de treinamento são organizados classificando as áreas de interesse preditas, ROIs com IoU superior a 0.7 são exemplos positivos, enquanto aqueles com IoU de 0.3 ou menos são exemplos negativos. Este critério foi ajustado em relação ao artigo original, que usava um limiar de 0.5 para positivos. Selecionamos 5 regiões positivas e 5 negativas, cada uma redimensionada para 224x224 pixels, diferente das dimensões originais de 32x32 e 96x96 pixels. Este balanceamento e padronização são cruciais para a eficácia do treinamento do modelo, melhorando sua precisão e capacidade de generalização.

Uma observação interessante neste experimento é que o algoritmo frequentemente propõe regiões com dimensões inadequadas, como ROIs com altura ou largura igual a zero ou valores muito pequenos. Essas características inviabilizam essas regiões, tornando-as inutilizáveis no contexto do conjunto de dados utilizado. Portanto, tais ROIs devem ser desconsideradas, independentemente do resultado do cálculo do IoU, pois não conseguem representar nenhuma classe, seja de fundo ou de algum objeto.

4.3.2 Arquitetura Rede Neural

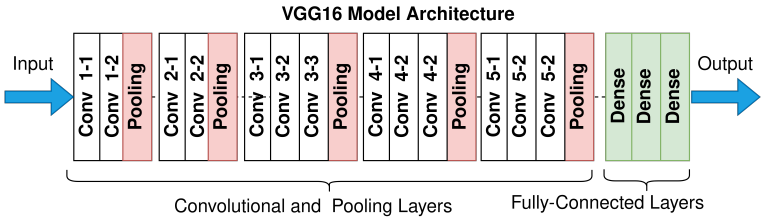
O treinamento do modelo foi realizado com base em uma rede neural pré existente chamada VGG16. Sua simplicidade e eficiência em tarefas de reconhecimento de imagens

Figura 28 – Diagrama preparação regiões propostas para treinamento



a tornaram amplamente utilizada. Com 16 camadas treináveis, sendo 13 convolucionais e 3 totalmente conectadas, a VGG16 usa filtros pequenos de 3x3 e aumenta a profundidade da rede adicionando mais camadas, o que permite capturar detalhes mais refinados das imagens. Cada conjunto de camadas convolucionais é seguido por uma camada de *Max Pooling*, que reduz a dimensionalidade, mantendo as características essenciais. Nas últimas camadas, a rede é totalmente conectada e utiliza uma camada *softmax* para a classificação final. A VGG16 é amplamente empregada em *transfer learning* por sua alta precisão, mas seu grande número de parâmetros a torna uma arquitetura pesada em termos de memória e processamento.

Figura 29 – Arquitetura VGG16



4.3.2.1 Treinamento do Modelo

Diversas abordagens foram adotadas ao longo do desenvolvimento, incluindo técnicas de *transfer learning*, treinamentos completos da rede, testes incluindo ou não implementação de *batch normalization* e *dropout*, entre outros. Após realizar uma série de testes em diferentes variações de modelos, foi possível identificar uma estratégia que apresentou os melhores resultados.

O primeiro passo realizado foi carregar a VGG16 sem as camadas *fully connected* originais, removendo a parte superior da rede (com o parâmetro *include_top=False*). Isso manteve apenas as camadas responsáveis pela extração de características das imagens, porém, sem usar pesos pré-treinados, o que significa que essas camadas foram treinadas do zero.

Em seguida uma camada *Flatten*, transforma a saída em um vetor unidimensional. Isso possibilita a passagem dos dados para as novas camadas densas de 64 neurônios que foram adicionadas com a ativação *ReLU*. Foram incluídas também camadas de *batch normalization* para normalizar as ativações e *dropout* com uma taxa de 20%, que desativa aleatoriamente uma fração dos neurônios durante o treinamento, ajudando a prevenir o *overfitting*.

O único neurônio da camada final é utilizado com a função de ativação *sigmoid* para realizar a classificação da região proposta pelo algoritmo do *Selective Search*, representando a probabilidade da região conter o objeto de interesse ou ser apenas o *background*.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 VGG16 (camadas anteriores)

 flatten (Flatten)            (None, 25088)             0

 dense (Dense)                (None, 64)                1,605,696

 batch_normalization (BatchNorm) (None, 64)             256

 dropout (Dropout)            (None, 64)                0

 dense_1 (Dense)              (None, 64)                4,160

 batch_normalization_1 (BatchNorm) (None, 64)           256

 dropout_1 (Dropout)          (None, 64)                0

 dense_2 (Dense)              (None, 1)                 65
=================================================================
```

O modelo foi compilado utilizando o otimizador Adam, com uma taxa de aprendizado de 1×10^{-8}, permitindo um ajuste gradual dos pesos durante o treinamento. A função de perda escolhida foi *binary_crossentropy* e a métrica usada para avaliar o desempenho do modelo foi a acurácia.

A escolha de 64 neurônios para a camada densa representa um equilíbrio entre capacidade e eficiência, pois um número moderado de neurônios ajuda a capturar padrões complexos sem tornar o modelo excessivamente grande ou suscetível ao *overfitting*. A função de ativação *ReLU* foi escolhida para ser utilizada nessa camada devido às sua característica de introduzir a não-linearidade no modelo, permitindo que ele aprenda representações complexas e interações não-lineares entre as características, além de ser computacionalmente eficiente e ajudar a evitar o problema do *vanishing-gradient*, o que contribui para um treinamento mais eficiente.

A motivação para usar o otimizador Adam é que ele ajusta automaticamente a taxa de aprendizado para cada parâmetro com base em momentos de primeiro e segundo ordens, combinando as vantagens dos métodos de gradiente estocástico com momento e adaptação da taxa de aprendizado. Isso permite uma convergência mais rápida e estável, evita problemas de ajuste inadequado da taxa de aprendizado, e é computacionalmente eficiente

4.4 RESULTADOS

Nesta seção, são apresentados os resultados do fluxo adaptado da R-CNN, que abrange todas as etapas do processo: desde o pré-processamento das imagens, passando pela preparação dos dados de treinamento com o uso da técnica *Selective Search* para definir possíveis regiões de interesse, até o treinamento do modelo.

É importante ressaltar que o objetivo deste trabalho não é apresentar os resultados mais precisos ou replicar o artigo original, mas sim demonstrar a execução do modo como as regiões são propostas e avaliar se elas realmente contêm objetos de interesse.

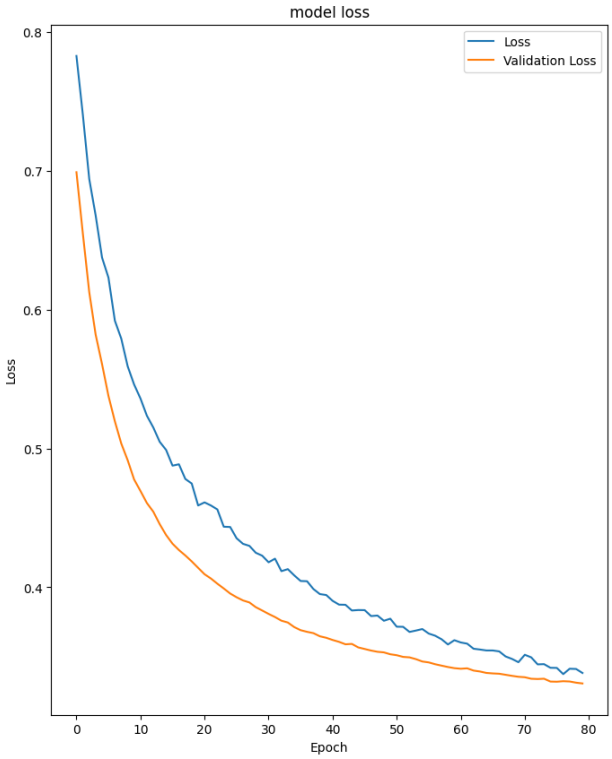
4.4.1 Resultado do treinamento do modelo

O total de parâmetros treináveis do modelo foi de 16,324,865 (62.27 MB) e realizamos seu treinamento por 80 épocas com *batch size* de 64 e *validation split* de 0.2. O tempo de execução de cada época era de aproximadamente 268 segundos.

Os resultados da última época de treinamento indicam um desempenho do modelo adaptado da R-CNN com acurácia de 86,24% no conjunto de treinamento e 87,33% no conjunto de validação demonstram que o modelo foi capaz de identificar corretamente as regiões que continham objetos de interesse em uma alta porcentagem dos casos considerando um referencial que não exige altas precisões.

Em relação às perdas, o valor de 0,3363 para o conjunto de treinamento e 0,3307

Figura 30 – Treinamento do modelo



Fonte: Autor

para o conjunto de validação na última época mostram que o modelo conseguiu minimizar a função de erro ao longo do treinamento. A pequena diferença entre as perdas dos dois conjuntos é outro indicativo de que o modelo está bem balanceado, com boa capacidade de generalização. Esse comportamento positivo sugere que as etapas de pré-processamento, definição de regiões de interesse e a estrutura do modelo R-CNN estão adequadamente configuradas para a tarefa proposta.

É possível observar na Figura 30 que as perdas do conjunto de treinamento são maiores do que as perdas do conjunto de validação, isso pode ocorrer devido a uma regularização mais intensa durante o treinamento. Quando técnicas de regularização, como *dropout* (que é o caso deste experimento), L2 ou *early stopping* são aplicadas, o modelo é incentivado a não se ajustar excessivamente aos dados de treinamento. A regularização pode resultar em um aumento das perdas no conjunto de treinamento, enquanto melhora a capacidade de generalização no conjunto de validação. Assim, a regularização pode fazer com que as perdas de validação sejam menores.

4.4.2 Amostras de Regiões de Interesse

Nesta subseção, serão apresentadas amostras das imagens processadas pelo modelo, destacando exemplos de boas e más regiões de interesse propostas durante o processo

de detecção. As imagens selecionadas ilustram casos em que a técnica *Selective Search* e o modelo treinado foram eficazes em identificar corretamente as regiões que realmente contêm objetos de interesse, bem como situações em que o método falhou, gerando regiões irrelevantes ou imprecisas. Essas amostras são fundamentais para analisar o comportamento do modelo em diferentes cenários e identificar possíveis melhorias no processo de detecção e na definição das regiões de interesse.

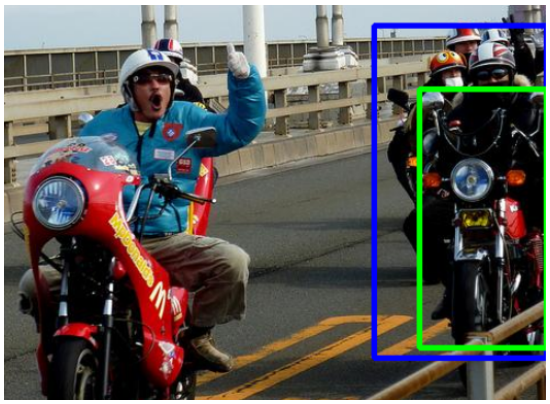
4.4.2.1 Amostras com bons resultados

Estas são amostras de imagens que conseguiram cumprir com o objetivo proposto pela adaptação da R-CNN. As regiões propostas pelo *Selective Search* foram realmente dentro do contexto da objeto de interesse e o modelo conseguiu identificar que esta região realmente continha este objeto.

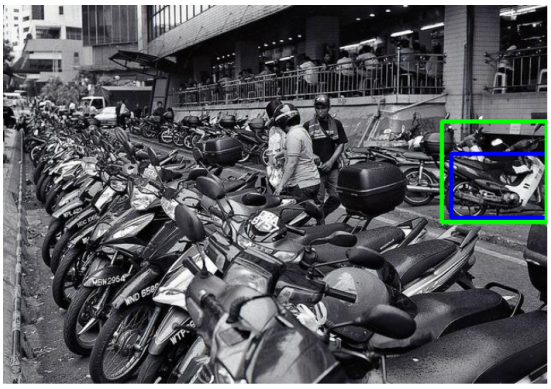
Figura 31 – Bons Resultados

(a) id: 102411 - IoU 0.68 - Resultado 0.96

(b) id: 226417 - IoU 0.57 - Resultado 0.90



(a) id: 143556 - IoU 0.57 - Resultado 0.97



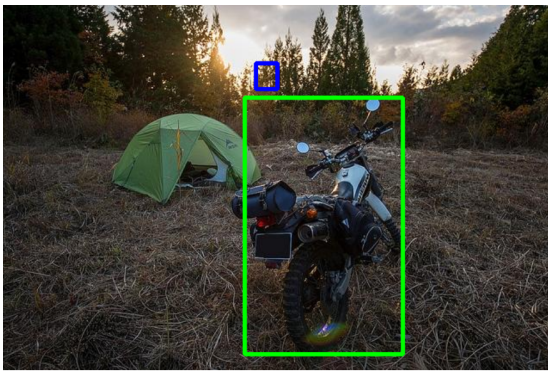
(b) id: 165681 - IoU 0.85 - Resultado 0.99

4.4.2.2 Amostras fora do esperado

Estas são amostras de imagens que apresentaram outras particularidades ou não conseguiram cumprir com o objetivo proposto. Os resultados apresentados nesta seção são análises de possíveis casos de futuras melhorias que podem ser feitas no projeto.

Figura 33 – Resultados fora do esperado

(a) id: 226417 - Resultado 0.96



(b) id: 48924 - Resultado: 0.93

Os casos da Figura 33 demonstram dois exemplos distintos de resultados não esperados.

Na primeira e na segunda imagem, podemos observar que houve uma identificação incorreta do objeto de interesse. A região em azul, proposta pelo *Selective Search*, foi selecionada com base em distinções de cor e forma, conforme esperado para o algoritmo. No entanto, o erro ocorre porque o modelo treinado identificou essa região como semelhante ao objeto de interesse em análise, neste caso, motocicletas.

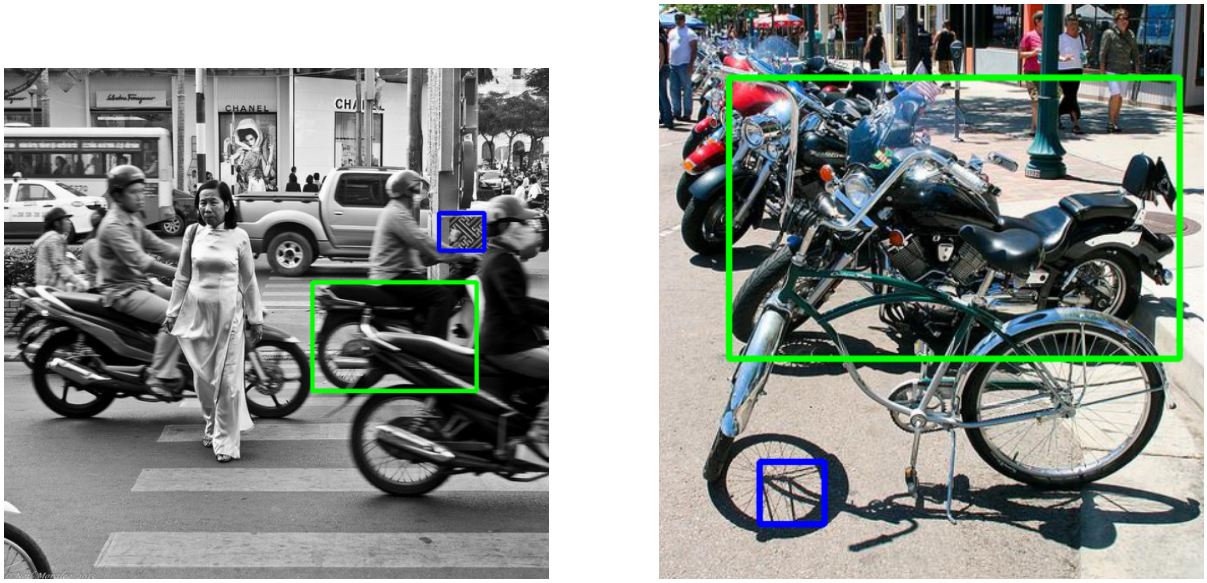
A identificação incorreta pode ter ocorrido devido à semelhança de características que o modelo identificou como relevantes entre a região proposta e as motocicletas, levando o modelo a cometer um falso positivo. Levando em consideração os resultados obtidos corretamente na maioria dos casos este aparenta ser o motivo mais plausível.

Outras possibilidades também envolvem questões como: dados de treinamento insuficientes ou desequilibrados, fazendo com que o modelo não diferencie corretamente entre o objeto de interesse e outros elementos. A generalização excessiva do modelo ou propostas de regiões inadequadas pelo *Selective Search* podem ter contribuído para o erro. Ou então, um possível *overfitting* pode estar fazendo com que o modelo memorize padrões específicos em vez de generalizar bem para novos exemplos. No entanto, estes motivos aparentam ser menos prováveis.

4.4.2.3 Amostras com particularidades

É importante observar que dentro dos conjuntos de amostras que ficaram fora dos resultados esperados, ou seja, com IoU iguais a 0, existem alguns casos particulares.

Figura 34 – Resultados fora do esperado com particularidade



(a) id: 181542 - Resultado 0.94

(b) id: 29634 - Resultado: 0.96

A primeira imagem da Figura 34 mostra uma região proposta que, à primeira vista, parece estar fora da *bounding box* original do conjunto de dados. No entanto, ao observar com mais atenção, percebe-se que a região proposta inclui parte do retrovisor da motocicleta. Uma boa hipótese é que o modelo identificou essa área como relevante devido às características aprendidas durante o treinamento, que associam retrovisores ao objeto de interesse. Neste caso, a predição de que a região contém o objeto de interesse estaria correta, mesmo que o rótulo original não capture completamente essa informação.

Na segunda imagem da Figura 34, observamos outra particularidade interessante. A imagem contém uma bicicleta, que é visualmente semelhante às motocicletas que é o objeto de interesse. Embora sejam diferentes, compartilham características que podem confundir o modelo.

Outro ponto relevante nesta imagem é a sombra projetada da bicicleta no chão. Embora não seja um objeto real, sua forma reflete características visuais do objeto em questão. O fato de o modelo ter identificado parte da sombra de um objeto semelhante ao objeto de interesse sugere que algumas características aprendidas durante o treinamento são compartilhadas entre ambos.

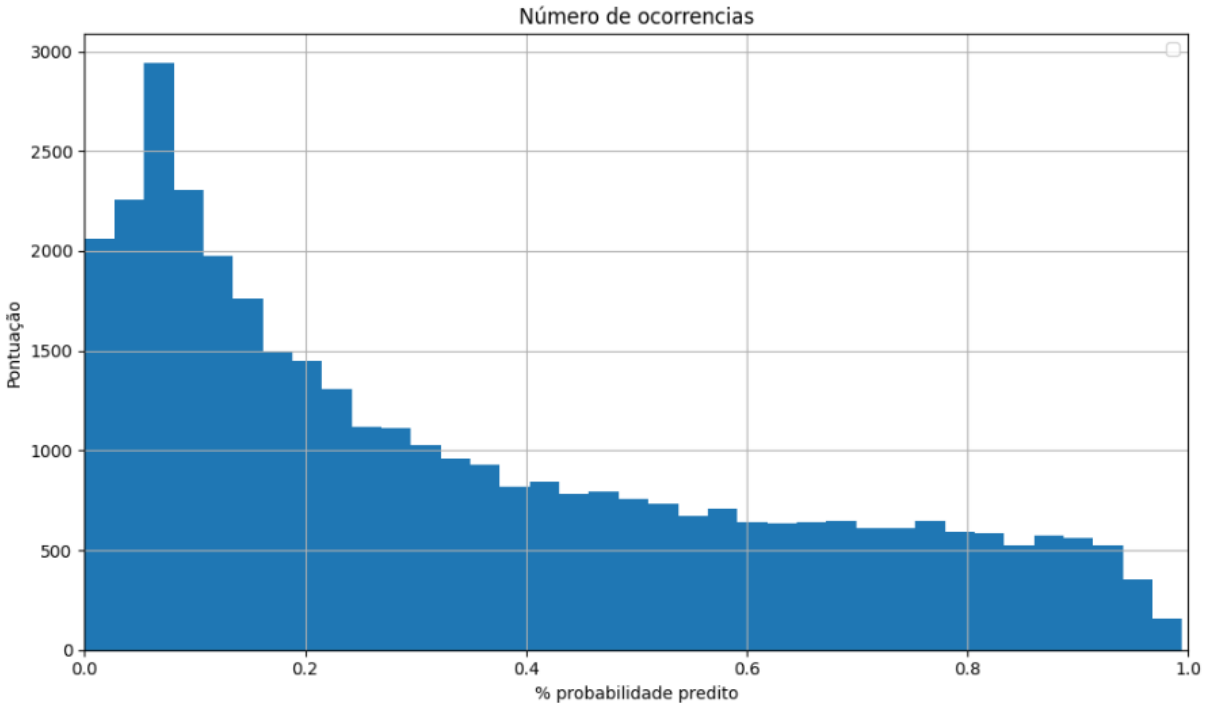
4.4.3 Análise resultados preditos

Na fase de testes, foram analisadas 371 imagens distintas. Para cada imagem, o algoritmo *Selective Search* gerou 100 regiões propostas, que foram avaliadas pelo modelo treinado. O objetivo era identificar se cada região continha ou não um objeto de interesse.

4.4.3.1 Histograma de distribuição dos resultados

Neste item, avaliamos a distribuição da probabilidade prevista pelo modelo para cada região conter um objeto de interesse. Observa-se que a maioria das imagens apresenta valores próximos a zero ou valores baixos de predição. Numericamente, cerca de 75% dos casos preditos estão abaixo de 55% (0,55), conforme ilustrado no histograma apresentado na Figura 35.

Figura 35 – Histograma distribuição de resultados



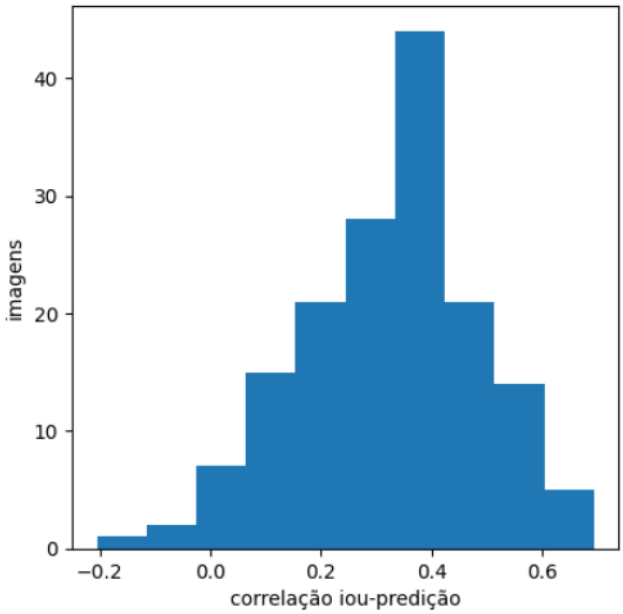
Este resultado, que revela a presença de muitas regiões com baixa probabilidade de conter o objeto de interesse, destaca uma limitação importante do *Selective Search*: sua dificuldade em generalizar para objetos específicos em imagens mais complexas. O método baseia-se em características visuais e padrões simples da imagem, como cor, textura e proximidade, o que o torna menos eficaz em capturar padrões mais sofisticados que redes neurais são capazes de reconhecer. Consequentemente, o *Selective Search* precisa propor um grande número de regiões para que algumas sejam úteis, o que o torna uma estratégia pouco eficiente para tarefas que exigem maior precisão na detecção de objetos.

4.4.3.2 Histograma de distribuição das correlações

Foi realizada a análise da hipótese de que, dentro das características específicas do modelo, o IoU das regiões propostas pelo algoritmo não teria uma correlação significativa com os valores de confiança preditos para cada região.

A hipótese se baseia na premissa de que não há uma relação direta entre o percentual predito e o IoU da região proposta, pois o modelo foi projetado para identificar se uma região contém ou não o objeto de interesse. Assim, uma região que captura apenas uma parte do objeto pode ter um alto percentual predito, indicando que o modelo reconheceu a presença do objeto, apesar de possuir um baixo IoU.

Figura 36 – Histograma distribuição de correlações



A Figura 36 apresenta os resultados que corroboram com a hipótese levantada: a maior parte das imagens apresenta correlações entre 0.3 e 0.5 entre os $IoUs$ das regiões propostas e seus respectivos valores preditos. Embora exista alguma relação entre essas variáveis — já que, para que o modelo identifique que uma região contém o objeto de interesse, é necessário que o IoU seja maior que zero —, a correlação não é necessariamente forte. Esse comportamento reforça a ideia de que a confiança das predições do modelo não não está diretamente associado ao IoU.

Pode-se explicar este fenômeno a partir dos quatro cenários de classificação das regiões propostas:

- **IoU alto e previsão alta:** Resultados verdadeiros positivos, onde o modelo indica com certeza que as boas regiões propostas contêm um objeto de interesse.
- **IoU alto e previsão baixa:** Resultados falsos negativos, onde a região proposta contém o objeto, mas o modelo não o identifica, resultando em erro. Este cenário indica o resultado mais incorreto do modelo.
- **IoU baixo e previsão alta:** Resultados falsos positivos, onde o modelo identifica um objeto na região, mesmo que não esteja completamente representado. Esse cenário não é necessariamente errado, pois é possível identificar um objeto a partir de uma parte dele.

- **IoU baixo e previsão baixa:** Resultados verdadeiros negativos, onde o modelo corretamente indica que uma região sem o objeto de interesse não possui relevância.

Neste contexto, os Falsos Positivos não podem ser interpretados como em outras classificações binárias, pois aqui podem ser interpretados como resultados satisfatórios. Além disso, como a proposição de regiões é feita pelo algoritmo do *Selective Search* e não pela rede neural, o formato das regiões não está diretamente relacionado aos resultados do modelo, mas sim à forma como o algoritmo identifica regiões relevantes para avaliação.

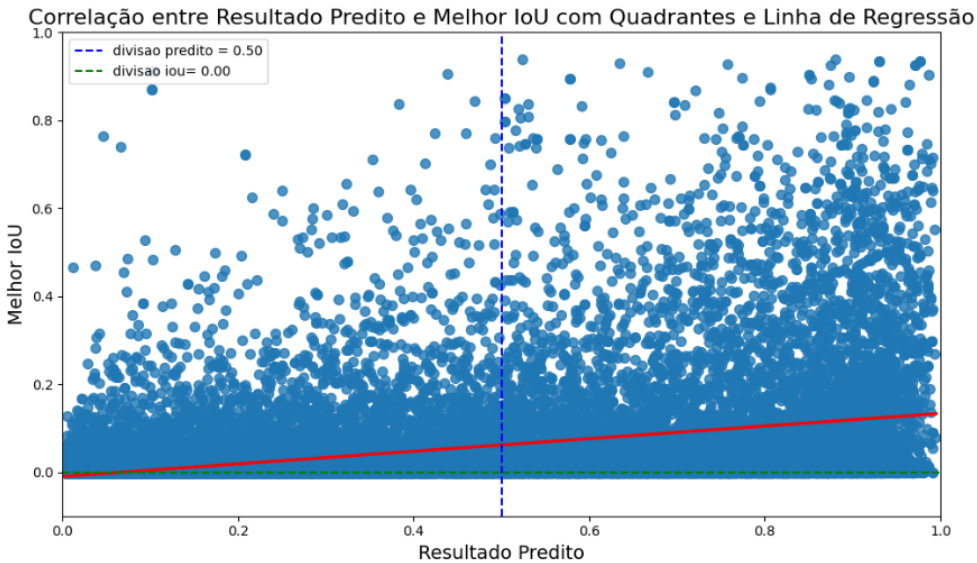
4.4.3.3 Dispersão *IoU* x Previsão

A partir da hipótese e das considerações apresentadas anteriormente, é possível visualizar a relação entre o IoU das regiões propostas e os valores de confiança preditos por meio de um gráfico de dispersão na Figura 37.

Este é o primeiro cenário de dois que serão realizados para avaliar os testes relacioados à análise da dispersão de IoU por percentuais de previsão. Para a classificação correta das informações as informações caracterizadas como positivas são referentes às regiões que contem totalmente ou parcialmente o objeto de interesse nas regiões analisadas.

De forma geral podemos definir que nestes experimentos os verdadeiros Positivos (TP) são caracterizados por um IoU elevado e uma previsão alta, indicando uma detecção precisa do modelo. Falsos Negativos (FN) ocorrem quando o IoU é alto, mas a previsão é baixa, refletindo uma falha na identificação do objeto. Falsos Positivos (FP) apresentam um IoU baixo associado a uma previsão alta, o que pode ser aceitável para objetos parcialmente detectados, embora introduzam ambiguidade com os Verdadeiros Positivos. Verdadeiros Negativos (TN) possuem IoU e previsão baixos, confirmando corretamente a ausência do objeto na região avaliada.

Figura 37 – Gráfico dispersão IoU x resultados



A visualização foi dividida em quatro regiões com limiares de 0.5 para valores preditos e 0.0 para o IoU, permitindo uma análise mais clara dos cenários. O limiar de 0.5 foi escolhido porque, acima desse valor, a região proposta tem maior probabilidade de conter o objeto de interesse do que de ser fundo. O limiar de 0.0 para o IoU reflete o objetivo do modelo de identificar a presença ou ausência do objeto na região proposta, independentemente de o objeto estar totalmente ou parcialmente presente. Assim, é possível avaliar numericamente os resultados e as métricas associadas à distribuição dos dados em cada quadrante, oferecendo uma visão quantitativa do desempenho do modelo em diferentes cenários como mostra a Tabela 2.

Tabela 1 – Métricas de Avaliação do Modelo

Métrica	Valor
Verdadeiro Positivo	7278
Falso Negativo	11469
Verdadeiro Negativo	14885
Falso Positivo	3468

F1 Score	0.494
Precisão	0.677
recall	0.388

- **Precisão:**

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{7}$$

A precisão foi calculada em 0.677, o que significa que aproximadamente 67,7% das previsões positivas do modelo estão corretas. Essa taxa de acerto indica que, quando o modelo afirma que encontrou um objeto, é mais provável que essa afirmação seja verdadeira.

- ***recall*:**

$$recall = \frac{TP}{TP + FN} \tag{8}$$

O *recall* do modelo foi medido em 0.388, indicando que apenas cerca de 38,8% dos objetos presentes foram corretamente identificados. No entanto, a interpretação desta métrica ficou comprometida devido a uma variação nos parâmetos do cálculo. Como discutido anteriormente, a interpretação dos falsos negativos deve ser considerada com cautela, especialmente em casos onde o IoU é baixo. Nesses casos, tanto um IoU baixo quanto uma predição de confiança baixa podem, de fato, representar verdadeiros negativos, pois a região proposta pode não conter o objeto de interesse de forma significativa.

- ***F1 Score*:**

$$F1 \text{ Score} = 2 \cdot \frac{\text{Precisão} \cdot \textit{recall}}{\text{Precisão} + \textit{recall}} \tag{9}$$

Como o *F1 Score* depende do *recall* para ser calculado o valor de 0.494 é o resultado obtido considerando esta variação dos parâmetros de cálculo do recall.

Uma outra abordagem para analisar esses resultados é considerar os casos em que o IoU não é muito baixo. Essa filtragem permite focar em regiões que abrangem uma parte mais substancial do objeto de interesse, eliminando situações onde a interpretação pode ser ambígua. A Figura 38 apresenta novamente o gráfico de dispersão, mas agora filtrado apenas regiões que possuem mais de 30% de IoU. O limiar de IoU também foi reajustado para 0.5 nesta avaliação.

Figura 38 – Gráfico dispersão IoU x resultados - filtrado

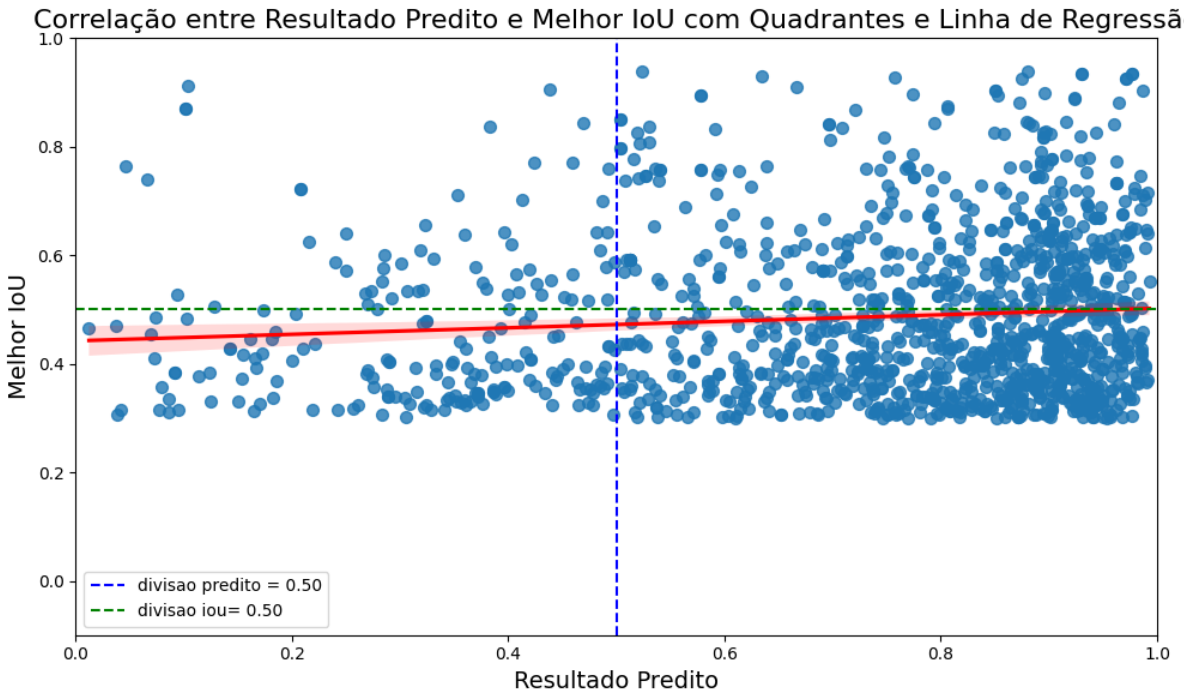


Tabela 2 – Métricas de Avaliação do Modelo

Métrica	Valor
Verdadeiro Positivo	442
Falso Negativo	61
Verdadeiro Negativo	139
Falso Positivo	659

F1 Score	0.551
Precisão	0.401
recall	0.879

É perceptível que houve uma variação significativa nas métricas analisadas. O *recall*, que anteriormente era de 38,8%, agora apresenta um valor de 87,9%, indicando

uma mudança expressiva na capacidade do modelo de identificar objetos presentes nas imagens.

Por outro lado, a precisão de 0.401 neste novo cenário não deve ser comparada diretamente com a precisão do cenário anterior, pois a alteração do limiar de *IoU* comprometeu os casos de análise, resultando em uma divisão distinta entre verdadeiros positivos e falsos positivos. Essa modificação na definição de limites afeta a interpretação das predições do modelo e pode levar a conclusões imprecisas sobre seu desempenho.

Da mesma forma, o *F1 Score*, que é uma métrica que combina precisão e *recall*, também foi afetado por essas mudanças. O valor de 0.551 reflete a realidade do desempenho do modelo apenas para este segundo caso em que a precisão foi comprometida.

De modo geral podemos compreender que o modelo é capaz de classificar corretamente grande parte das amostras realizadas, como apresentado pela métrica de acurária de 86% comentada anteriomente. No entanto a influência da ambiguidade entre os falsos positivos e os verdadeiros positivos influenciou as métricas de precisão, recall e F1-score que ficaram comprometidas em alguns dos cenários de teste.

4.4.4 Discussões dos Resultados

Após a avaliação dos resultados obtidos, podemos afirmar que o modelo desenvolvido apresenta um desempenho satisfatório dentro do contexto proposto. Essa conclusão é respaldada por métricas relevantes, como o *F1 Score*, *recall*, precisão, que indicam uma boa capacidade de identificação de objetos de interesse. No entanto, reconhecemos que há oportunidades significativas para aprimorar a eficiência precisão das predições a fim de aumentar a robustez do modelo.

Um ponto importante observado nos experimentos foi a constatação de que o percentual de IoU não está diretamente associado à probabilidade predita pelo modelo. Essa característica foi específica da adaptação realizada, uma vez que o objetivo se concentrou exclusivamente em identificar se a região proposta continha ou não o objeto de interesse, ou parte dele.

Essa observação exigiu uma análise mais cautelosa dos resultados. Ao examinar o gráfico de IoU em relação à probabilidade predita, notou-se certa ambiguidade, conforme apresentado anteriormente. Casos com baixo IoU e alta probabilidade podem indicar regiões pequenas dentro do objeto de interesse que refletem *features* específicas, mas que não abrangem o objeto em sua totalidade.

Essa característica é inerente à arquitetura proposta, pois o *Selective Search*, nesta adaptação, foi configurado para capturar objetos de tamanhos variados. Consequentemente, como a necessidade de propor regiões grandes e pequenas visava detectar todos os objetos presentes na imagem, a geração de regiões pequenas resultou em casos em que a região proposta representava apenas partes menores do verdadeiro objeto de interesse.

Apesar dessa particularidade da adaptação da *R-CNN*, a análise dos resultados

filtrados indicou que o conjunto formado pelo Selective Search e pela rede neural treinada conseguiu identificar corretamente a presença do objeto de interesse, ou parte dele, na maioria dos casos testados.

Apesar de este trabalho não focar em replicar os resultados do artigo original, é importante reconhecer que também existem outras abordagens com a possibilidade de levar a detecções mais precisas em trabalhos futuros. Neste contexto, aumentar o volume de dados de treinamento e garantir anotações precisas é essencial para melhorar a robustez do modelo. Além disso, a inserção de parâmetros de ajuste (*offsets*) podem ser adicionados como saídas da rede neural a fim de ajustar melhor as regiões propostas aos objetos. Por fim, uso de técnicas como *Non-Maximum Suppression* (NMS) ajudaria a evitar sobreposições, melhorando a qualidade das detecções. Esses pontos, embora fora do escopo atual, podem contribuir para obter resultados mais fieis aos que foram apresentados em (GIRSHICK *et al.*, 2013).

5 CONCLUSÃO

5.1 CONSIDERAÇÕES FINAIS

Ao longo deste trabalho, foi possível observar as análises teóricas nas redes neurais baseadas em regiões. Além dessa compreensão teórica, o estudo concentrou-se em uma análise prática da adaptação da R-CNN, demonstrando que, mesmo sendo uma adaptação, os resultados obtidos corroboram com as características teóricas discutidas previamente.

De acordo com os artigos analisados, os autores destacam a evolução técnica voltada para o aprimoramento da eficiência computacional dessas arquiteturas, uma vez que a R-CNN original apresentava um alto custo computacional e tempo de treinamento elevado. Na aplicação prática deste trabalho, observou-se a mesma limitação, confirmando a necessidade de otimizações apontada na literatura.

Embora a ineficiência computacional tenha sido um desafio, os resultados experimentais demonstraram que o modelo desenvolvido foi capaz de atingir um desempenho satisfatório quanto comparado com o artigo original. A combinação do Selective Search com a rede neural mostrou-se eficiente na identificação correta da presença do objeto nas regiões de interesse, em grande parte das amostras testadas.

Conclui-se, portanto, que o objetivo do trabalho foi atingido ao demonstrar que, embora a R-CNN apresente resultados satisfatórios, ainda há margem significativa para melhorias. Tanto a arquitetura quanto a qualidade dos resultados podem ser aprimoradas, reforçando a importância das evoluções contínuas observadas nas versões subsequentes das redes neurais baseadas em regiões .

REFERÊNCIAS

ANANTH, Shilpa. **Fast R-CNN for object detection**. en. [*S.l.*: *s.n.*], ago. 2019. Disponível em: https://towardsdatascience.com/fast-r-cnn-for-object-detection-a-technical-summary-a0ff94faa022.

ANELLO, Eugenia. **Visualizing the feature maps and filters by convolutional neural networks**. en. [*S.l.*: *s.n.*], jun. 2021. Disponível em: https://medium.com/dataseries/visualizing-the-feature-maps-and-filters-by-convolutional-neural-networks-e1462340518e.

CHUANG, Yuelong; ZHANG, Shiqing; ZHAO, Xiaoming. Deep learning-based panoptic segmentation: Recent advances and perspectives. en. **IET image processing**, v. 17, n. 10, p. 2807–2828, 2023. ISSN 1751-9659. DOI: 10.1049/ipr2.12853. Disponível em: http://dx.doi.org/10.1049/ipr2.12853.

COELHO, Mateus. **Fundamentos de Redes Neurais - Laboratório iMobilis**. pt. [*S.l.*]: Laboratório iMobilis, jun. 2017. Disponível em: https://www2.decom.ufop.br/imobilis/fundamentos-de-redes-neurais/.

FEATURE Vector Representation Neural Networks. en. [*S.l.*: *s.n.*]. Disponível em: https://stackoverflow.com/questions/27227450/feature-vector-representation-neural-networks.

GAD, Ahmed Fawzy. **Faster R-CNN explained for object detection tasks**. en. [*S.l.*: *s.n.*], nov. 2020. Disponível em: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/.

GANDHI, Rohith. **R-CNN, fast R-CNN, faster R-CNN, YOLO — object detection algorithms**. en. [*S.l.*: *s.n.*], jul. 2018. Disponível em: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

GIRSHICK, Ross. Fast R-CNN, 2015. DOI: 10.48550/ARXIV.1504.08083. Disponível em: http://dx.doi.org/10.48550/ARXIV.1504.08083.

GIRSHICK, Ross *et al.* Rich feature hierarchies for accurate object detection and semantic segmentation, 2013. DOI: 10.48550/ARXIV.1311.2524. Disponível em: http://dx.doi.org/10.48550/ARXIV.1311.2524.

GONZALEZ, Rafael C.; WOODS, Richard E. **Digital Image Processing, Global Edition**. 4. ed. London, England: Pearson Education, 2017. ISBN 9781292223049.

HAFIZ, Abdul Mueed; BHAT, Ghulam Mohiuddin. A Survey on Instance Segmentation: State of the art, 2020. DOI: 10.48550/ARXIV.2007.00047. Disponível em: http://dx.doi.org/10.48550/ARXIV.2007.00047.

HASSABALLAH, Mahmoud; AWAD, Ali Ismail. Detection and Description of Image Features: An Introduction. *In:* [*S.l.*: *s.n.*], fev. 2016. v. 630, p. 1–8. ISBN 978-3-319-28852-9. DOI: 10.1007/978-3-319-28854-3_1.

HE, Kaiming *et al.* Mask R-CNN, 2017. DOI: 10.48550/ARXIV.1703.06870. Disponível em: http://dx.doi.org/10.48550/ARXIV.1703.06870.

HUI, Jonathan. **Understanding Feature Pyramid Networks for object detection (FPN)**. en. [*S.l.*: *s.n.*], mar. 2018. Disponível em: https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c.

INDOLIA, Sakshi *et al.* Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach. **Procedia Computer Science**, v. 132, p. 679–688, 2018. International Conference on Computational Intelligence and Data Science. ISSN 1877-0509. DOI: https://doi.org/10.1016/j.procs.2018.05.069. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050918308019.

KARMARKAR, Tanay. **Region proposal network (RPN) — backbone of faster R-CNN**. en. [*S.l.*: *s.n.*], ago. 2018. Disponível em: https://medium.com/@codeplumber/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9.

KAUR, Dilpreet; KAUR, Yadwinder. Various image segmentation techniques: a review. **International Journal of Computer Science and Mobile Computing**, v. 3, n. 5, p. 809–814, 2014.

LIN, Tsung-Yi *et al.* Feature pyramid networks for object detection, 2016. DOI: 10.48550/ARXIV.1612.03144. Disponível em: http://dx.doi.org/10.48550/ARXIV.1612.03144.

LISBOA, Adriano *et al.* Interação com a sociedade por meio de sistema de visão computacional para monitoramento ambiental de linhas de transmissão, nov. 2019.

REN, Shaoqing *et al.* Faster R-CNN: Towards real-time object detection with region proposal networks, 2015. DOI: 10.48550/ARXIV.1506.01497. Disponível em: http://dx.doi.org/10.48550/ARXIV.1506.01497.

RESAMPLING Methods. [*S.l.*: *s.n.*]. Disponível em: https://seadas.gsfc.nasa.gov/help-9.0.0/general/overview/ResamplingMethods.html.

RESEARCH, Fractal AI. **Mask R-CNN unmasked - Fractal AI research**. en. [*S.l.*: *s.n.*], mai. 2022. Disponível em: https://medium.com/@fractal.ai/mask-r-cnn-unmasked-28419817a990.

SANTOS, Elineide *et al.* Explicando as decisões com IAs: Demonstrando sua aplicação em imagens médicas. *In:* [*S.l.*: *s.n.*], jun. 2023. P. 92–133. ISBN 9788576695462. DOI: 10.5753/sbc.12346.2.3.

SULTANA, Farhana; SUFIAN, Abu; DUTTA, Paramartha. Evolution of image segmentation using deep convolutional neural network: A survey. [*S.l.*], 2020. Disponível em: http://arxiv.org/abs/2001.04074.

THAKUR, Rohit. **Step-by-step R-CNN implementation from scratch in python**. en. [*S.l.*: *s.n.*], out. 2019. https://towardsdatascience.com/step-by-step-r-cnn-implementation-from-scratch-in-python-e97101ccde55. Accessed: 2024-10-16.

UIJLINGS, Jasper *et al.* Selective Search for Object Recognition. **International Journal of Computer Vision**, v. 104, p. 154–171, set. 2013. DOI: 10.1007/s11263-013-0620-5.

WHAT Is Image Filtering In The Spatial Domain. en. [*S.l.*: *s.n.*]. Disponível em: https://www.mathworks.com/help/images/what-is-image-filtering-in-the-spatial-domain.html.

YAMASHITA, Rikiya *et al.* Convolutional neural networks: an overview and application in radiology. en. **Insights into imaging**, v. 9, n. 4, p. 611–629, 2018. ISSN 1869-4101. DOI: 10.1007/s13244-018-0639-9. Disponível em: http://dx.doi.org/10.1007/s13244-018-0639-9.