Pontifícia Universidade Católica
do Rio de Janeiro

Claudio Vieira Escudero

Seleção de retângulos envolventes em arquiteturas para detecção de objetos

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio .

Orientador: Prof. Marcus Vinicius Soledade Poggi de Aragão

Rio de Janeiro
Abril de 2021

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Claudio Vieira Escudero

Seleção de retângulos envolventes em arquiteturas para detecção de objetos

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio . Aprovada pela Comissão Examinadora abaixo.

Prof. Marcus Vinicius Soledade Poggi de Aragão
Orientador
Departamento de Informática – PUC-Rio

Prof. Helio Cortes Vieira Lopes
Departamento de Informática – PUC-Rio

Prof. Marco Antonio Casanova
Departamento de Informática – PUC-Rio

Dr. Luiz José Schirmer Silva
Departamento de Informática – PUC-Rio

Rio de Janeiro, 16 de Abril de 2021

Claudio Vieira Escudero

Graduou-se em Sistemas de Informação na PUC-Rio (Pontifício Universidade Católica do Rio de Janeiro) em 2009.

Ficha Catalográfica

Escudero Vieira,Claudio

Seleção de retângulos envolventes em arquiteturas para detecção de objetos / Claudio Vieira Escudero; orientador: Marcus Vinicius Soledade Poggi de Aragão. – Rio de janeiro: PUC-Rio , Departamento de Informática, 2021.

v., 58 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui bibliografia

1. Informática – Teses. 2. Visão Computacional;. 3. Rede Neural Convolucional;. 4. Non-Maximum Suppression;. 5. Soft-NMS;. 6. Non-Maximum Weighted;. 7. Weighted Boxes Fusion;. 8. Retângulos Envolventes;. 9. Tempo Real.. I. Soledade Poggi de Aragão, Marcus Vinicius. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Ao meu orientador Profº. Drº. Marcus Poggi que sempre foi atencioso e acessível.

A minha mãe pelo incentivo e apoio em todos momentos que precisei.

À minha esposa, Flávia Escudero, pelo apoio, pelas valiosas contribuições e pela paciência em sempre ler o que escrevia.

A Deus, por esta incrível oportunidade.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

Resumo

Escudero Vieira,Claudio; Soledade Poggi de Aragão, Marcus Vinicius. **Seleção de retângulos envolventes em arquiteturas para detecção de objetos**. Rio de Janeiro, 2021. 58p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação estuda métodos e algoritmos para critérios de seleções dos retângulos envolventes focando em arquiteturas de detecção de objetos baseada redes neurais convolucionais para tempo real, que processam mais de 30fps, que também possibilitam a expansão para outras arquiteturas. O objetivo desta dissertação é melhorar as métricas Recall e Precision, proporcionando mais assertividade nos resultados destas arquiteturas sem a necessidade de recriá-las ou retreiná-las, diminuindo, assim, os recursos para manutenções. As arquiteturas que trabalham em tempo real normalmente não apresentam melhores resultados, pois são desenvolvidas visando a redução do tempo de execução. Para resolver estes problemas, serão testados outros métodos de critérios de seleção de retângulos envolventes em estado da arte, são eles: Non-maximum Suppression (NMS), Soft-NMS, Non-Maximum Weighted (NMW) e Weighted Boxes Fusion (WBF). Os resultados obtidos foram comparados aos originais das arquiteturas, utilizando as métricas mAP, Recall e Precision. Através desta comparação foi possível comprovar que os novos critérios apresentaram bons resultados. O tempo de execução dos novos critérios também foi analisado com execuções de imagens em lotes, contornando alguns overheads dos critérios mais pesados. As arquiteturas utilizadas como base nos experimentos foram baseadas nos sistemas YOLOv3-Tiny e YOLOv4-Tiny, utilizando o dataset QMUL-OpenLogo público e especializado em logotipos e baseado em fotos reais.

Palavras-chave

Visão Computacional; Rede Neural Convolucional; Non-Maximum Suppression; Soft-NMS; Non-Maximum Weighted; Weighted Boxes Fusion; Retângulos Envolventes; Tempo Real.

Abstract

Escudero Vieira,Claudio; Soledade Poggi de Aragão, Marcus Vinicius (Advisor). **Bounding boxes selection in object detection architectures**. Rio de Janeiro, 2021. 58p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This dissertation studies methods and algorithms for bounding box selection criteria focusing on object detection architectures based on convolutional neural networks for real-time, processing over 30fps, which also allow expansion to other architectures. The goal of this study is to improve the Recall and Precision metrics, providing more assertiveness in the results of these architectures without the need to recreate or retrain them, thus reducing the resources for maintenance. Architectures that work in real-time usually do not present good results, because they are developed aiming to reduce execution time. To solve these problems, other state-of-the-art bounding box selection criteria methods will be tested: Non-maximum Suppression (NMS), Soft-NMS, Non-Maximum Weighted (NMW) and Weighted Boxes Fusion (WBF). The results obtained were compared to the original architectures, using the mAP, Recall and Precision metrics. Through this comparison it was possible to prove that the new criteria presented satisfactory results. The execution time of the new criteria was also analyzed with batch image executions, bypassing some overheads of the heavier criteria. The architectures used as basis for the experiments were based on the YOLOv3-Tiny and YOLOv4-Tiny systems, using the public dataset QMUL-OpenLogo specialized in logos and based on real photos.

Keywords

Computer Vision; Convolutional Neural Network; Non-Maximum Suppression; Soft-NMS; Non-Maximum Weighted; Weighted Boxes Fusion; Bounding Boxes; Real-Time.

Sumário

Lista de figuras

Lista de tabelas

Lista de algoritmos

Lista de Códigos

Lista de símbolos

SDD – Single Shot MultiBox

YOLO – You Only Look Once

AP – Average Precision mAP – Mean Average Precision

IoU – intersection over union

CDIoU – Control Distance IoU

GIoU – Generalized Intersection over Union

NN – Neural Network

CNN – Convolutional Neural Network

NMS – Non-maximum Suppression

NMW – Non-Maximum Weighted

WBF – Weighted Boxes Fusion

FP – False Positive

TP – True Positive

FN – Fale Negative

FP – False Positive

COCO – Common Objects in Context

FPS – Frames Per Seconds

RGB – Representa as cores Red, Green e Blue

Se tu o desejas, podes voar, só tens de confiar muito em ti.

Steve Jobs

1
Introdução

O mundo está cada vez mais globalizado e as estratégias de competição do mercado se apresentam como ferramentas fundamentais para conquistar visibilidade. Com o aumento da qualidade da internet, o consumidor está visualizando cada vez mais conteúdos de imagens e vídeos. Com isso, a gestão online das marcas torna-se mais complexa, dificultando, assim, o monitoramento da concorrência e até mesmo a detecção de violação de direitos autorais.

A proposta deste trabalho é desenvolver uma arquitetura de *deep learning* focada no reconhecimento de logotipos de marcas em vídeos ou imagens, processada em tempo real, utilizando técnicas de redes neurais convolucionais e seleção de retângulos envolventes, que também possibilita uma expansão para outras arquiteturas de detecção de objetos e outras classes de objetos, não necessariamente logotipos. Por exemplo, os experimentos se basearam em logotipos porque o nível de detalhe da classe não é tão complexo quanto o nível de detalhamento de uma classe animal, já que um animal pode estar em diversas posições e também pode estar realizando inúmeras ações. A classe logotipo é estática, uma vez que, geralmente, só existe um único formato por marca..

As técnicas de *deep learning* têm sido capazes de resolver muitos problemas de detecção de objetos utilizando determinadas redes neurais performáticas e, deste modo, otimizam as execuções tempo real.

Dessa maneira, é possível detectar, de uma forma automática e no momento exato, se uma marca ou produto aparecem em uma imagem ou vídeo. De tal modo, é possível responder a algumas perguntas, como por exemplo:

- Quanto tempo a minha marca ficou visível?
- Estão violando os meus direitos autorais?
- A minha marca aparece em qual momento do vídeo?
- O concorrente aparece junto com a minha marca?

1.1 Motivação

É sempre um grande desafio processar arquiteturas de redes convolucionais em tempo real. Dentro dessa ótica, as arquiteturas exigem processamentos robustos, o que dificulta a execução de arquiteturas leves com grandes performances e altas acurácias. Entretanto, a diferença do resultado ainda é grande quando comparada às arquiteturas mais pesadas que não são apropriadas para processamentos tempo real. Tomando por base uma análise dessas constatações, torna-se importante desenvolver um aprimoramento confiável de detecção e classificação de imagens.

A grande motivação é poder trazer melhores resultados nas arquiteturas atuais no mercado sem precisar realizar retreino, utilizando apenas o refinamento das informações geradas por elas.

Apesar de alguns estudos terem sido encontrados na literatura sobre detecções de logotipos, pois é foco do experimento, não foi encontrado nenhum modelo de redes neurais já treinadas e com seus pesos disponibilizados para a realização de possíveis *benchmark*. Com isso, um *benchmark* foi gerado nesta dissertação utilizando arquiteturas originais e realizando treinamentos com um *dataset* público de logotipos. Este novo *benchmark* possibilitará a comparação das modificações propostas.

1.2 Definição do problema

As arquiteturas de redes neurais convolucionais de processamento de detecção de objetos em tempo real normalmente possuem uma baixa acurácia comparadas às arquiteturas que não são adequadas à característica de processamento em tempo real. Dessa forma, confrontando ambos os resultados, constata-se que as arquiteturas tempo real, na maioria vezes, apresentam resultados insatisfatórios.

Na tentativa de solucionar os problemas de acurácia identificados nos processamentos tempo real, muitos usuários optam por realizar retreinos ou, até mesmo, pesquisar novas arquiteturas mais modernas. Entretanto, este caminho pode ser muito oneroso, já que grandes processamentos podem elevar muito o custo de implementação.

Em visto disso, alguns ajustes na etapa de pós-processamento podem contribuir para resolver este problema, uma vez que o refinamento das combinações das detecções de objetos possibilita a melhora da acurácia dos resultados. Dessa forma, é possível proporcionar ganhos sem a necessidade de retreino ou do desenvolvimento de outras arquiteturas.

A fim de ilustrar a comparação dos resultados das referidas as arquiteturas, a tabela 1.1 traz uma listagem dos modelos de detecção de objetos mais utilizados na atualidade. Nela observa-se o contraste entre os valores de mAP (Mean Average Precision) (seção 2.4.3) das arquiteturas que podem ou não serem aplicadas em tempo real. Os modelos que estão em negrito são aplicados em tempo real, tendo em vista que neste trabalho para as arquiteturas serem consideradas tempo real, precisam processar mais de 30fps.

Modelo	Imagem (px)	FPS	mAP	GPU
DETR-DC5-R101	-	10	44.9%	-
SSD512	512	19	79.5%	Titan X
DSSD321	321	9.5	78.6%	Titan X
Faster RCNN-DC5	-	16	39.0%	-
SAPD+ ResNet-101-DCN	-	9.1	46.0%	
RPDet+ ResNet-101-DCN		8	45.0%	
EfficientDet-D1	640	74	40.5%	Tesla V100
EfficientDet-D0	512	98	34.6%	Tesla V100
YOLOv4	416	96	41.2%	Titan X
YOLOv4-tiny	416	165	40.2%	Titan RTX
YOLOv3	320	45	28.2%	Titan X
YOLOv3-tiny	416	200	33.1%	Titan X

Tabela 1.1: Arquiteturas de detecção de objetos em redes neurais convolucionais

Referências da tabela 1.1: (Shashi Pal et al., 2017), (Nassif et al., 2019), (Bochkovskiy et al., 2020), (Tan et al., 2020) e (Han et al., 2019)

1.3 Objetivos

O presente trabalho tem como objetivo central aprimorar os resultados das redes neurais convolucionais de detecções dos objetos, que foram construídas com o intuito de serem utilizadas em tempo real.

Um dos pontos essenciais é aperfeiçoar o pós-processamento das arquiteturas de redes convolucionais, através das definições de mudanças nos critérios das seleções dos retângulos envolventes, pois estas arquiteturas normalmente apresentam baixas acurácias por se tratarem de redes que foram construídas pensando na redução do tempo de processamento e na otimização das execuções. Com isso, elas não conseguem mapear muitos detalhes no treinamento dos modelos e, assim, geram resultados equivocados. Dessa forma, foram mapeados alguns critérios para as seleções de retângulos envolventes em estados da arte que prometem trazer bons resultados e que serão utilizados no trabalho. Além disso, também serão utilizados testes com diferentes quantidades de

imagens em lote no processamento das detecções, ao invés de processar uma imagem por vez. Dessa forma, será possível mostrar um ganho nas execuções em tempo real.

É importante levar em consideração o tempo de execução, porque, eventualmente, pode ocorrer um gargalo no tempo de latência nos sistemas tempo real e, se o tempo de processamento for alto, o tempo de espera pode tornar-se impraticável. O ideal é possuir uma baixa latência a fim de viabilizar a execução.

1.4 Contribuições

Esta dissertação propõe alternativas para o melhoramento da precisão dos modelos de redes neurais convolucionais de detecções de objetos existentes e já treinados, modificando apenas o pós-processamento na seleção dos retângulos envolventes, sem a necessidade de realizar retreino. Com isso, ao utilizar apenas o refinamento proposto, os resultados apresentaram melhoras quando comparados aos gerados na arquitetura original.

Também será definido um método completo para processamentos em lotes para detecção de objetos, melhorando a performance de processamento.

1.5 Trabalhos relacionados

O YOLO (Redmon et al., 2016) foi uma das primeiras redes neurais de detecção de objetos super-rápidas capaz de processar em tempo real. Como o YOLO não possui uma característica de "*region proposal*", há a necessidade de executar a arquitetura em diversas regiões da imagem para localizar os objetos, realizando uma passagem na rede apenas uma única vez. O YOLO foi desenvolvido para ser processado uma única vez, sendo o pioneiro neste modo de localizar os objetos.

O trabalho "*Real-Time Single-Shot Brand Logo Recognition*" (Bombonato et al., 2017) foi desenvolvido baseado na estrutura *Single Shot MultiBox* (SDD) (Liu et al., 2016) com o objetivo de detectar logotipos para serem processados em tempo real. Isto contribuiu para que neste trabalho fosse comprovado que o uso da Rede Neural Convolucional é, de fato, adequado para realizar este tipo de tarefa.

Um trabalho importante que está relacionado a esta dissertação é o "*Open Logo Detection Challenge*" (Hang Su et al., 2018), pois ele foi construiu um *dataset* baseado em outros, contendo milhares de imagens e centenas

de logotipos já pré-classificados. Este *dataset* já foi utilizado na arquitetura *Faster-RCNN* obtendo um mAP com performance de 48.3%

1.6 Estrutura da dissertação

Esta dissertação discute arquiteturas e técnicas no estado da arte, com o intuito de apresentar melhores resultados processados em tempo real. Ademais, esta dissertação está dividida em 5 (cinco) capítulos, a fim de aprimorar os resultados descritos nos experimentos da solução:

Capítulo 2 – Conceitos básicos: Este capítulo tem a finalidade de apresentar alguns conceitos e as principais métricas para o entendimento geral dos experimentos.

Capítulo 3 – Arquitetura de detecção de objetos: Neste capítulo será apresentada a arquitetura que foi desenvolvida para ser utilizada nos experimentos.

Capítulo 4 – Resultados: Neste capítulo são aplicadas novas técnicas, com suas implementações, nas arquiteturas em estado da arte, com o objetivo de serem utilizadas no melhoramento dos resultados.

Capítulo 5 – Conclusão: Neste capítulo são apresentadas conclusões do comportamento das técnicas aplicadas.

2
Conceitos Básicos

2.1
Deep Learning

Deep learning é o estado da arte dentre os métodos de aprendizagem de máquinas. Embora os conceitos básicos desta técnica tenham sido desenvolvidos no ano 1971 pelo matemático Alexey Ivakhnenko (Ivakhnenko, 1971), considerado "o pai da *Deep learning*", ao longo dos últimos 20 anos, tais conceitos foram altamente aprimorados, proporcionando, assim, meios para o desenvolvimento da inteligência artificial.

As arquiteturas de redes neurais profundas, redes neurais recorrentes e redes neurais convolucionais são utilizadas na área de visão computacional, reconhecimento de fala (Nassif et al., 2019), processamento de linguagem natural (Junyi Chai & Anming Li, 2019) e tradução automática (Shashi Pal et al., 2017). É importante ressaltar que, em alguns casos, estas técnicas se assemelham ou, até mesmo, superam a performance humana (Cirean et al., 2012).

Cabe destacar a importância de determinadas redes neurais convolucionais, pois apresentam um alto desempenho no reconhecimento de imagens e processamento de vídeos e por isso são utilizadas, até mesmo, em carros autônomos (Grigorescu et al., 2020).

A primeira arquitetura de reconhecimento de imagem foi a LeNet (Lecun et al., 1998) desenvolvida por Yann LeCun. Entretanto, a mesma não era muito profunda por questões de capacidade de processamento da época, mas proporcionou a elaboração de diversos conceitos utilizados até hoje. No ano de 2012, foi desenvolvida a arquitetura AlexNet (Krizhevsky et al., 2012) que ganhou visibilidade porque obteve grandes resultados classificando imagens na competição Large Scale Visual Recognition Challenge (ILSVRC) (ILSVRC, 2012).

2.2
Detecção de objetos

Usando arquiteturas de *Deep Learning* podemos construir redes para executar tarefas com objetivo identificar e localizar padrões em uma imagem, como, por exemplo, localizar pessoas, rostos, armas de fogo, animais, etc.

Essas técnicas são muito utilizadas em carros autônomos, já que para um carro andar sozinho, ele precisa identificar pessoas, carros, sinais de trânsito, entre outros obstáculos. Além disso, tais técnicas também são utilizadas em outras diversas áreas de visões computacionais.

As redes mais conhecidas são:

- R-CNN (Girshick et al., 2016), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2017)
- SSD (Liu et al., 2016)
- RetinaNet (Lin et al., 2020)
- YOLO v1 (Redmon et al., 2016), YOLO 9000 (Redmon & Farhadi, 2017), YOLO v3 (Redmon & Farhadi, 2018), YOLO v4 (Bochkovskiy et al., 2020)
- EfficientDet (Tan et al., 2020)

2.3 Latência e tempo real

Latência é o tempo de atraso entre a entrada e saída de uma transformação ou deslocamento dos dados. A latência de vídeo é a transferência de um *stream* para um telespectador no momento da solicitação, normalmente medido em milissegundos.

Os principais fatores que provocam a latência são atrasos na propagação na rede de comunicação, o tempo dos *pipelines* de decodificação/codificação de vídeo, o tempo da rede de entrega de conteúdo (CDN) e, em alguns outros casos, as limitações de infraestrutura.

As transferências que possuem atrasos relativamente baixos são conhecidas como redes de baixa latência, enquanto as que possuem retardos significativos são conhecidas como redes de alta latência (WEBSITE AWS, 2021). Quando se trata de tempo real, existem faixas de tempo para alta e baixa latência, seguem as faixas na tabela 2.1.

	Alta (segundos)	Baixa (segundos)
Latência reduzida	18	6
Baixa latência	6	2
Latência ultra-baixa	2	0,2

Tabela 2.1: Níveis de latência de vídeo

2.4 Principais métricas

2.4.1 Intersect over Union (IoU)

A principal métrica para determinar se um conjunto de retângulo envolvente pertence ao mesmo objeto é o *Intersect over Union* (IoU) (Wajih et al., 2020), frequentemente referenciado como métrica de coeficiente de Jaccard, que consiste em medir a sobreposição de duas regiões de interesse.

O cálculo do IoU é formulado pela equação 2-1

$$\text{IoU} = \frac{area(b_1 \bigcap b_2)}{area(b_1 \bigcup b_2)} \tag{2-1}$$

Onde b_1 e b_2 representam 2 (dois) retângulos envolventes. O valor da razão entre as duas áreas é comparado com um valor limite. O valor IoU limite é uma constante que determina se os 2 (dois) retângulos referem-se ao mesmo objeto.

$$\text{IoU} < \text{IoU}_{limite} \tag{2-2}$$

A figura 2.1 contém exemplos de 2 (dois) dos retângulos envolventes com diferentes situações de sobreposição.

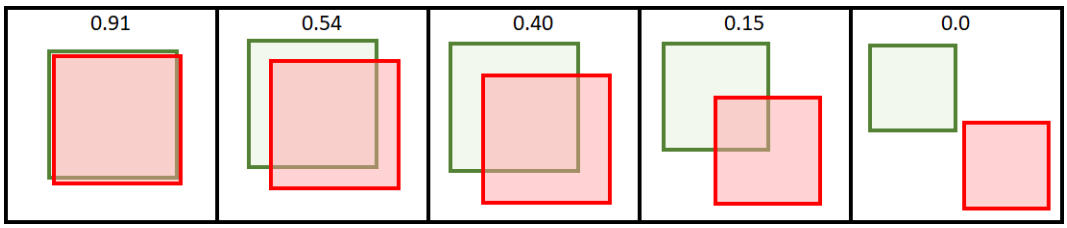

Figura 2.1: Valores do IoU em diferentes sobreposições

Observando o exemplo 2.1, a interpretação do IoU é bastante intuitiva. O valor do IoU pode corresponder ao intervalo entre 0 até 1, sendo que o valor máximo é igual a 1 (um), significando que o retângulo detectou exatamente a mesma posição que o retângulo original. O valor mínimo é igual a 0 (zero), significando que os retângulos envolventes não se sobrepõem.

2.4.2 Matriz de confusão

Matriz de confusão é uma tabela que apresenta o desempenho de um algoritmo de classificação com uma visão geral das previsões de teste dos

modelos em relação aos valores verdadeiros. As principais métricas que fazem uso da matriz são Acurácia, *Precision* e *Recall*.

Acurácia é uma métrica que faz a medição baseada na performance geral do modelo, mostrando quantas detecções foram corretas dentre todas as detecções.

Precision é uma métrica que faz a medição baseada nos FP (*False Positive*) que são considerados prejudiciais, e, em contrapartida, os FN (*False Negative*) não são relevantes para a métrica.

Recall é uma métrica que faz a medição baseada nos FN (*False Negative*) que são considerados prejudiciais, e, em contrapartida, os FP (*False Positive*) não são relevantes.

A matriz é uma tabela com formato de 2x2 e contém quatro combinações nos resultados das previsões (linhas) e valores verdadeiros (colunas). As combinações são: *True Positive* (TP), *False Positive* (FP), *False Negative* (FN) e *True Negative* (TN).

É importante destacar as definições das combinações da matriz de confusão no conceito de detecção de objetos:

- *True Positive* (TP): É uma detecção correta do objeto da mesma classe. Ocorre quando o retângulo envolvente da previsão e do gabarito estão com o IoU $\geq$ IoU$_{limite}$. Segue o exemplo na figura 2.2
- *False Positive* (FP): É uma detecção incorreta do objeto da mesma classe. Ocorre quando o retângulo envolvente da previsão e do gabarito estão com o IoU $>$ IoU$_{limite}$. Segue o exemplo na figura 2.3
- *False Negative* (FN): Ocorre quando o modelo não detectou o objeto. Segue o exemplo na figura 2.4
- *True Negative* (TN): Não se aplica. Ocorre quando uma previsão correta é feita de onde não existe o real. Sendo assim, todos os TN serão classificados como FP. Este valor não é usado nas métricas.

Figura 2.2: Exemplo de *True Positive*

Figura 2.3: Exemplo de *False Positive*

Figura 2.4: Exemplo de *False Negative*

2.4.3 Mean average precision

A métrica *Mean Average Precision* (mAP) é utilizada nas medições para modelos de detecção de objetos. Para o entendimento do mAP é necessário compreender melhor outras duas métricas, *Precision-Recall Curve* e *Average precision*

2.4.3.1 Precision vs Recall curve

A *Precision vs Recall curve* é uma boa maneira de avaliar o desempenho de um modelo de detector de objetos à medida que a confiabilidade é alterada de uma forma decrescente traçando uma curva para cada classe de objeto. Um modelo utilizado por uma determinada classe é considerado bom se seu *precision* se mantiver alto (o mais próximo de um) enquanto o *recall* aumenta, o que significa que se houver uma variação no limite da confiabilidade, os valores do *precision* e *recall* ainda permanecerão altos.

2.4.3.2 Average Precision

Average precision (AP) é uma métrica que calcula o desempenho dos modelos de detecção de objetos. Uma prática muito comum do cálculo do desempenho é calcular a área sob a curva do *Precision x Recall*. As curvas do AP apresentam um comportamento que aumenta os valores do *recall* à medida que os valores de FN diminuem. Entretanto, *precision* desce quando os valores de FP aumentam, mas sobe quando o valor de TP aumenta.

Hoje existem duas formas de calcular a área sob a curva do *Precision x Recall*. A primeira utiliza uma interpolação de apenas 11 pontos igualmente espaçados, como segue na equação 2-3, e outra forma é criar uma interpolação com todos os pontos de detecção, como segue na equação 2-4.

$$AP = \frac{1}{11} \times \sum_{re} max(precision(re))))) \tag{2-3}$$

$$AP = \sum_{1}^{r=0}((r_{n+1} - r_n) \times max(precision(r_{n+1}))) \tag{2-4}$$

2.4.3.3 Definição de mAP

Na métrica *mean Average Precision* (mAP) é calculada a média de todos os resultados do AP para cada classe do modelo de detecção de objetos. Segue a equação 2-5:

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2-5}$$

3
Arquitetura de detecção de objetos

Este capítulo vai detalhar o uso dos métodos propostos (NMW e WBF) na etapa do pós-processamento nos modelos de detecção de objetos. Com o intuito de buscar alternativas mais eficazes para o melhorar as seleções dos retângulos envolventes, foram utilizadas modificações dos sistemas (YOLOv3-Tiny e YOLOv4-Tiny) e também um *dataset* de logotipos (QMUL-OpenLogo) que será descrito mais à frente. Os resultados dos experimentos do capítulo 4 possibilitarão a comparação dos resultados utilizando diversas configurações no pós-processamento. Embora a arquitetura tenha sido desenhada para a detecção das classes logotipos em tempo real, a proposta pode seguir com qualquer outro tipo de arquiteturas e para outras classes de objetos.

Na seção 3.8 será demonstrada a proposta da arquitetura completa com processamento em lotes, aprimorando e otimizando os resultados.

3.1
Visão geral

A arquitetura de detecção de objetos foi desenvolvida com o objetivo de classificar e permitir a localização dos objetos. Por conta disso, serão apresentadas comparações e propostas de refinamento com intuito de aprimorar o resultado das seleções dos retângulos envolventes.

A figura 3.1 mostra as três principais das etapas que serão abordadas nesta dissertação, incluindo a etapa de pós-processamento dos resultados, que é a responsável pela seleção dos retângulos.

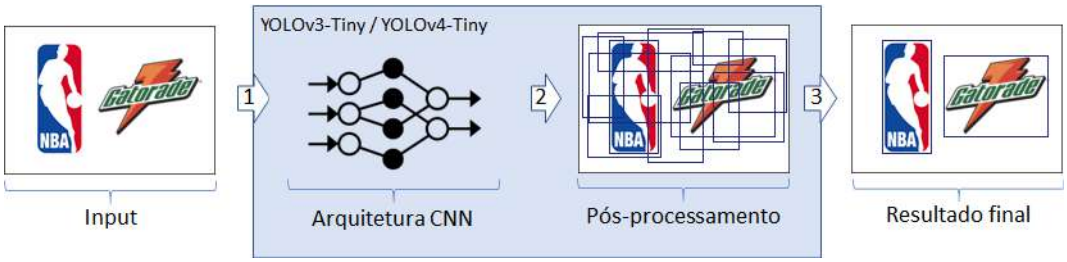

Figura 3.1: Etapas da arquitetura

Etapa 1: A rede neural convolucional (CNN) recebe uma imagem ou um frame de um vídeo para realizar a detecção. Essa imagem se converter em uma matriz com os valores RGB. Esta matriz será utilizada no YOLO.

Etapa 2: A rede neural convolucional realiza as detecções com centenas de possíveis resultados. Esses resultados possuem 3 principais informações:

- Rótulo: Identificação da classificação do objeto;
- Coordenadas: Os dois pontos com os pares dos eixos X e Y;
- Confiabilidade: Um número entre 0 a 1 indicando a probabilidade que essa localização está correta.

Etapa 3: O pós-processamento realiza uma etapa importante, que é a seleção dos retângulos envolventes que foram retornados pela rede neural convolucional. O critério das seleções utilizado é o non-maximum weighted (NMS) como são definidos na arquitetura original.

Dessa forma, o objetivo da dissertação é propor outras seleções para os retângulos envolventes que pertencem a etapa de pós-processamento. Os métodos propostos serão: Non-Maximum Weighted (NMW) e Weighted Boxes Fusion (WBF).

Além dos métodos propostos, nas próximas seções irão abordar os métodos NMS e Soft-NMS para possíveis revisões da literatura e para um entendimento mais detalhado. Em vista disso, os resultados são comparações dos métodos NMW e WBF com o método NMS.

3.2 Non-maximum Suppression (NMS)

Nas seleções dos retângulo envolvente, o *non-maximal suppression* (Bodla et al., 2017) é utilizado no YOLO, pois ele seleciona o candidato com maior confiabilidade para um determinado objeto. Este algoritmo elimina todos os vizinhos mais próximos. Os modelos que utilizam técnicas de *Single Shot* normalmente utilizam técnicas de NMS, sendo os atuais YOLO (Redmon & Farhadi, 2018) (Bochkovskiy et al., 2020) e SSD (Liu et al., 2016).

Segue a função do NMS no algoritmo 1.

Algoritmo 1: NMS

```
1  INPUT: B = {b_1, ..., b_n}, S = {s_1, ..., s_N}, N_t
2         B é uma lista inicial de retângulos envolventes detectados
3         S é uma lista de confiabilidade de B
4         N_t é o limite do IoU
5  D ← {}
6  enquanto B ≠ vazio faça
7      m ← argmax(S)
8      M ← b_m
9      D ← D ⋂ M
10     B ← B − M
11     para b_i in B faça
12         se IoU(M, b_i) ≥ N_t então
13             B ← B − b_i
14             S ← S − s_i
15         fim
16     fim
17 fim
18 retorna D, S
```

A função do NMS possui 3 (três) parâmetros de entrada. O primeiro é o conjunto B, que contém todos os candidatos a retângulos envolventes. O segundo é o conjunto S, que contém um conjunto de confiabilidade de cada candidato do conjunto B. Por último, o conjunto N_t, que é o limite do IoU. Com isso, a função retorna um novo conjunto de retângulos D (linha 5) selecionado. Mas para que isso aconteça, é necessário fazer uma série de iterações (linha 6) no conjunto B com todos os candidatos e para cada iteração é selecionado o candidato com maior confiabilidade (linha 8). Assim, esse candidato é adicionado no conjunto D de retorno e removido do conjunto de iterações (linha 9). As linhas 11 a 13 são responsáveis por remover os candidatos que se sobrepõem comparando com IoU superior ao N_t, que basicamente pertencem ao mesmo objeto e possuem um valor menor de confiabilidade. Caso os candidatos possuam valores do IoU inferiores ao N_t na sobreposição, eles entrarão num outro momento da iteração da linha 6.

A figura 3.2 possui um exemplo com duas etapas ilustrativas do comportamento do critério do NMS:

Etapa 1: Realiza uma iteração de forma decrescente de confiabilidade de cada retângulo envolvente, eliminando os candidatos que se sobrepõem ao candidato com maior confiabilidade de acordo com o limite do IoU.

Etapa 2: Seleciona apenas os retângulos com maior confiabilidade de cada objeto.

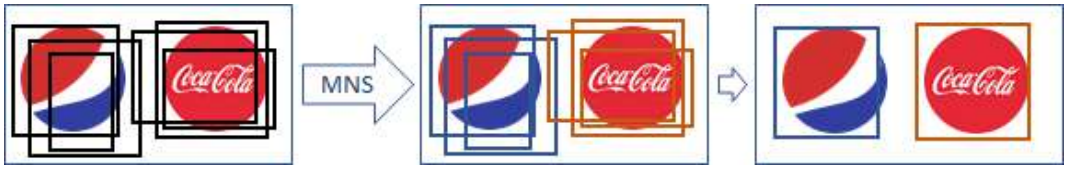

Figura 3.2: Duas etapas finais do NMS

Com os critérios acima, adicionando um limite do IoU alto, é possível criar um grande número de FP (*False Positive*), pois este processo irá gerar mais de um resultado para o mesmo objeto.

De acordo com o algoritmo 1, a complexidade do NMS é de $O(n^2)$, pois a quantidade de n é a quantidade de candidatos no conjunto B.

3.3 Soft-NMS

O algoritmo Soft-NMS (Bodla et al., 2017) é similar ao NMS, mas ao invés de remover os candidatos mais próximos, ele adiciona um peso entre 0 e 1 na confiabilidade dos candidatos, proporcional ao valor do IoU, ocasionando uma redução. Segue a função do Soft-NMS no algoritmo 2:

Algoritmo 2: Soft-NMS

```
1  INPUT: B = {b_1, ..., b_n}, S = {s_1, ..., s_N}, N_t
2         B é uma lista inicial de retângulos envolventes detectados
3         S é uma lista de confiabilidade de B
4         N_t é o limite do IoU
5  D ← {}
6  enquanto B ≠ vazio faça
7      m ← argmax(S)
8      M ← b_m
9      D ← D ⋂ M
10     B ← B − M
11     para b_i in B faça
12         s_i ← s_i f(IoU(M, b_i))
13     fim
14 fim
15 retorna D, S
```

As linhas de 5 a 11 são similares ao NMS, a diferença entre os algoritmos está presente na linha 12, que reduz a confiabilidade dos demais candidatos.

Existem 2 (dois) métodos de redução: linear ou gaussiana.

A equação 3-1 apresenta o método de redução linear:

$$s_i = \begin{cases} s_i & \text{IoU}(M, b_i) < N_t, \\ s_i(1 - \text{IoU}(M, b_i)) & \text{IoU}(M, b_i) \geq N_t \end{cases} \tag{3-1}$$

A função da equação 3-1 adiciona um peso na confiabilidade dos candidatos com limites superiores a N_t com a sobreposição do IoU, fazendo com que os candidatos sobrepostos sofram reduções. Sendo assim, os candidatos não sobrepostos não irão sofrer com o peso na confiabilidade.

A equação 3-2 apresenta o método de redução gaussiana:

$$s_i = s_i e^{-\frac{\text{IoU}(M,b_i)^2}{\sigma}}, \forall b_i \notin D \tag{3-2}$$

A função do equação 3-2 é aplicada para cada iteração, reduzindo a confiabilidade dos candidatos restantes baseada no IoU sem filtro de sobreposição com o limite do N_t.

O exemplo da Figura 3.2 do NMS é similar ao Soft-NMS, a diferença é que o Soft-NMS não elimina os candidatos sobrepostos, ele apenas atribui um peso e, consequentemente, reduz a confiabilidade. Este processo faz com que o resultado do mAP (*mean Average Precision*) seja um pouco ganancioso, pois os retângulos envolventes mais próximos provocam uma redução na confiabilidade, dando uma baixa preferência às detecções FP (*False Positive*) sem eliminá-los. Com isso, o valor do mAP aumenta dando a sensação de que o resultado é mais satisfatório comparando com o NMS.

De acordo com o algoritmo 2, a complexidade do Soft-NMS é de $O(n^2)$, sendo a mesma do NMS, pois a quantidade de n é a quantidade de candidatos no conjunto B.

3.4 Non-maximum weighted (NMW)

O método *Non-maximum weighted* (NMW) (Ning et al., 2017) foi desenvolvido para a modelo SSD substituindo o NMS. Com base nesta referência, a mesma abordagem desenvolvida no SSD também será utilizada nos testes desta dissertação.

O NMW utiliza média ponderada com o valor do IoU. Ao contrário dos métodos citados nas seções 3.2 e 3.3, que simplesmente eliminam os outros candidatos, o NMW utiliza todos os candidatos previstos para o mesmo objeto,

visando melhorar expressivamente a qualidade dos retângulos envolventes. Segue o método NMW no algoritmo 3:

Algoritmo 3: NMW

```
1  INPUT: B = {b_1, ..., b_n}, S = {s_1, ..., s_N}, N_t
2          B é uma lista inicial de retângulos envolventes detectados
3          S é uma lista de confiabilidade de B
4          N_t é o limite do IoU
5  D ← {}
6  enquanto B ≠ vazio faça
7      m ← argmax(S)
8      M ← b_m
9      BW_sum ← 0
10     w_sum ← 0
11     para b_i in B faça
12         se IoU(M, b_i) ≥ N_t então
13             w ← s_i × IoU(M, B_i)
14             BW_sum ← BW_sum + (w × b_i)
15             w_sum ← w_sum + w
16             B ← B − b_i
17             S ← S − s_i
18         fim
19     fim
20     M_aux ← BW_sum / w_sum
21     D ← D ⋂ M_aux
22 fim
23 retorna D, S
```

O algoritmo 3 é similar ao NMS, a diferença está dentro da condição da linha 12, que gera 2 (dois) somatórios que serão utilizados na linha 20 que, consequentemente, cria um novo retângulo envolvente que é adicionado no conjunto de retorno na linha 21.

Também é possível formular o método em equações. Supondo que todos os candidatos pertencem ao mesmo objeto, as equações 3-3 e 3-4 do método NMW indicarão uma determinada seleção de retângulo envolvente:

$$box = \frac{\sum_{i=1}^{n} w_i \times B_i}{\sum_{i=1}^{n} w_i} \tag{3-3}$$

$$w_i = C_i \times \mathrm{IoU}(B_i, B_{argmax(C_i)}) \tag{3-4}$$

Na equação 3-3, n é a quantidade de candidatos, já a equação 3-4 calcula o valor do IoU do candidato do somatório por meio da sobreposição do candidato de maior confiabilidade. O valor de w é o peso para cada candidato.

Segundo o artigo (Chengcheng Ning et al., 2017) do SSD com o *dataset* OOD, o método NMW obteve ganhos de 0.11% a 1.21% comparando com NMS, conforme na tabela 3.1.

Method	mAP	person	car	tree	stone	stair
I-SSD300	79.5	78.4	81.1	67.9	82.6	87.7
I-SSD300+NMW	79.9	78.7	81.4	68.0	83.6	87.8

Tabela 3.1: Performance no OOD com NMW

A imagem 3.3 apresenta um exemplo ilustrativo com as coordenadas do retângulo envolvente da tabela 3.2.

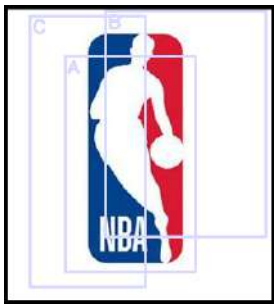

Figura 3.3: Exemplo de *input* no NMW

#	Confiabilidade	X1	Y1	X2	Y2	IoU
A	0.95	60	50	190	265	1.0
B	0.80	100	5	260	230	0.34
C	0.35	25	10	140	280	0.41

Tabela 3.2: Valores das coordenadas dos retângulos envolventes do *input*

A equação 3-5 demonstra o cálculo da criação de um novo retângulo envolvente, resultando numa nova coordenada com os valores [X1=64, Y1=36, X2=198, X2=259]:

$$
\begin{aligned}
BB_1[X_1] &= \frac{0.95 \times 1.0 \times 60 + 0.80 \times 0.34 \times 100 + 0.35 \times 0.41 \times 25}{0.95 \times 1.0 + 0.80 \times 0.34 + 0.35 \times 0.41} &= 64 \\
BB_1[Y_1] &= \frac{0.95 \times 1.0 \times 50 + 0.80 \times 0.34 \times 5 + 0.35 \times 0.41 \times 10}{0.95 \times 1.0 + 0.80 \times 0.34 + 0.35 \times 0.41} &= 36 \\
BB_1[X_2] &= \frac{0.95 \times 1.0 \times 190 + 0.80 \times 0.34 \times 260 + 0.35 \times 0.41 \times 140}{0.95 \times 1.0 + 0.80 \times 0.34 + 0.35 \times 0.41} &= 198 \\
BB_1[Y_2] &= \frac{0.95 \times 1.0 \times 265 + 0.80 \times 0.34 \times 230 + 0.35 \times 0.41 \times 280}{0.95 \times 1.0 + 0.80 \times 0.34 + 0.35 \times 0.41} &= 259
\end{aligned}
\tag{3-5}
$$

A imagem 3.4 mostra o resultado com o novo retângulo envolvente:

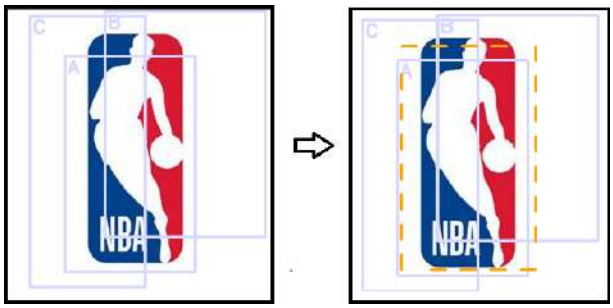

Figura 3.4: Exemplo do resultado do NMW

De acordo com a função NMW, é necessário percorrer todos os candidatos e localizar todos os candidatos que pertencem ao mesmo objeto. Com isso, é constata-se que a complexidade do NMW é de $O(n^2)$.

3.5 Weighted boxes fusion (WBF)

O método *Weighted boxes fusion* (WBF) (Solovyev et al., 2019) foi desenvolvido para o uso de combinações de diferentes resultados de modelos, propondo um ajuste das coordenadas e da confiabilidade dos resultados gerados. Nesta dissertação será utilizada um combinador de retângulos envolventes sobrepostos no mesmo objeto para a seleção do resultado da arquitetura. Dessa forma, haverá uma ponderação e um refinamento do resultado.

O funcionamento do WBF é similar ao NMW, mas ao invés de ajustar apenas as coordenadas, também ajusta o valor da confiabilidade. Segue o método WBF no algoritmo 4:

Algoritmo 4: WBF

1 **INPUT:** $B = \{b_1, ..., b_n\}, S = \{s_1, ..., s_N\}, N_t$
2 B é uma lista inicial de retângulos envolventes detectados
3 S é uma lista de confiabilidade de B
4 N_t é o limite do IoU
5 $D \leftarrow \{\}$
6 $S_r \leftarrow \{\}$
7 **enquanto** $B \neq vazio$ **faça**
8 $m \leftarrow argmax(S)$
9 $M \leftarrow b_m$
10 $BS_{sum} \leftarrow 0$
11 $s_{sum} \leftarrow 0$
12 $n \leftarrow 0$
13 **para** b_i *in* B **faça**
14 **se** $IoU(M, b_i) \geq N_t$ **então**
15 $n \leftarrow n + 1$
16 $BS_{sum} \leftarrow BS_{sum} + (s_i \times b_i)$
17 $s_{sum} \leftarrow s_{sum} + s_i$
18 $B \leftarrow B - b_i$
19 $S \leftarrow S - s_i$
20 **fim**
21 **fim**
22 $M_{aux} \leftarrow BS_{sum}/s_{sum}$
23 $S_{aux} \leftarrow s_{sum}/n$
24 $D \leftarrow D \bigcap M_{aux}$
25 $S_r \leftarrow S_r \bigcap S_{aux}$
26 **fim**
27 **retorna** D, S_r

O algoritmo 4 é similar ao NMS, a diferença está dentro da condição da linha 14, que gera 2 (dois) somatórios que serão utilizados nas linhas 16 e 17 e, por fim, cria um retângulo envolvente e sua confiabilidade que serão retornados.

Supondo que todos os candidatos pertencem ao mesmo objeto, a equação 3-6 do método WBF indicará uma determinada seleção de retângulo envolvente:

A implementação do WBF segue a equação 3-6:

$$box = \frac{\sum_{i=1}^{n} C_i \times B_i}{\sum_{i=1}^{n} C_i}$$
$$C = \frac{\sum_{i=1}^{n} C_i}{n} \tag{3-6}$$

Apesar das semelhanças entre os métodos WBF e NMW, existem algumas diferenças significativas, são elas:

- WBF modifica o valor de confiabilidade;
- WBF não utiliza o IoU para pesar os retângulos envolventes;
- NMW utiliza somente o maior valor de confiabilidade entre os candidatos para o objeto, enquanto o WBF utiliza todos os valores de confiabilidade.

Na figura 3.5 segue um exemplo dos resultados obtidos entre NMS/Soft-NMS vs WBF, contendo um conjunto de previsões (em verde) e uma detecção verdadeira (em vermelho).

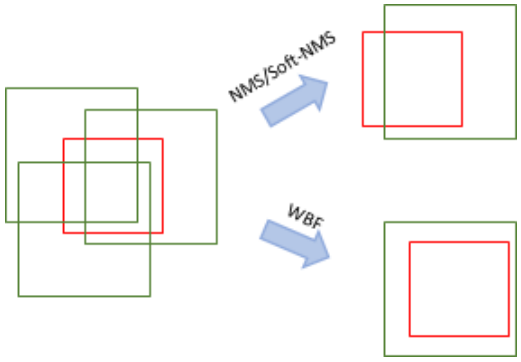

Figura 3.5: Diferença ilustrativa do NMS e WPF

Seguindo o mesmo exemplo da seção NMW, a equação 3-7 apresenta o cálculo da criação de um novo retângulo envolvente utilizando o método WBF, resultando numa nova coordenada com os valores [X1=64, Y1=36, X2=198, X2=259]:

$$BB[X_1] = \frac{0.95 \times 1.0 \times 60 + 0.80 \times 0.34 \times 100 + 0.35 \times 0.41 \times 25}{0.95 \times 1.0 + 0.80 \times 0.34 + 0.35 \times 0.41} = 64$$
$$BB[Y_1] = \frac{0.95 \times 1.0 \times 50 + 0.80 \times 0.34 \times 5 + 0.35 \times 0.41 \times 10}{0.95 \times 1.0 + 0.80 \times 0.34 + 0.35 \times 0.41} = 36$$
$$BB[X_2] = \frac{0.95 \times 1.0 \times 190 + 0.80 \times 0.34 \times 260 + 0.35 \times 0.41 \times 140}{0.95 \times 1.0 + 0.80 \times 0.34 + 0.35 \times 0.41} = 198$$
$$BB[Y_2] = \frac{0.95 \times 1.0 \times 265 + 0.80 \times 0.34 \times 230 + 0.35 \times 0.41 \times 280}{0.95 \times 1.0 + 0.80 \times 0.34 + 0.35 \times 0.41} = 259 \tag{3-7}$$

A imagem 3.6 mostra o resultado com o novo retângulo envolvente:

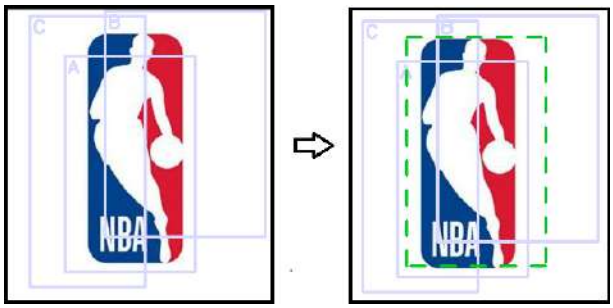

Figura 3.6: Exemplo do resultado do WBF

A função WBF possui a complexidade de $O(n^2)$, considerando n o tamanho dos candidatos, já que para localizar sobreposições é necessário percorrer todos os candidatos e localizar todos os que estão sobrepostos.

3.6 Comparação dos métodos

Nesta seção serão apresentadas etapas de uma comparação de todos os critérios de uma forma ilustrativa.

Etapa 1: A imagem 3.7 é utilizada como input no modelo para identificar os logotipos.

Figura 3.7: Exemplo de input

Etapa 2: O modelo retorna uma lista de retângulos envolventes com as suas respectivas confiabilidades na tabela 3.3 e na imagem 3.8 com os valores já filtrados com as confiabilidades e IoU pré-calculado:

#	Confiabilidade	X1	Y1	X2	Y2	IoU
A	0.95	60	50	190	265	1.0
B	0.80	100	5	260	230	0.34
C	0.35	25	10	140	280	0.41
D	0.93	380	35	520	290	1.0
E	0.50	290	1	580	180	0.30
F	0.55	310	10	590	190	0.34

Tabela 3.3: Valores das coordenadas dos retângulos envolventes do *input*

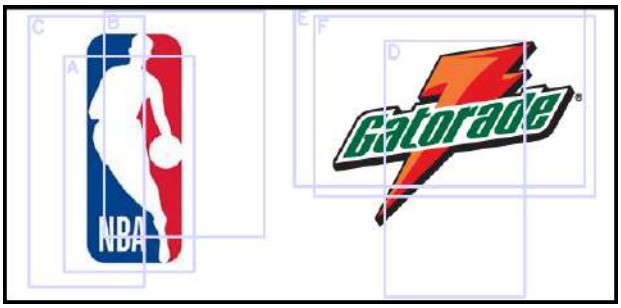

Figura 3.8: Exemplo dos retângulos envolventes retornados pelo modelo sem critério

Etapa 3: Aplica os critérios NMS, Soft-NMS, NMW e WBF observados na imagem 3.9. A cor vermelha identifica os retângulos do NMS e Soft-NMS, a cor amarela identifica os retângulos do NMW e a cor verde é a identificação do WBF.

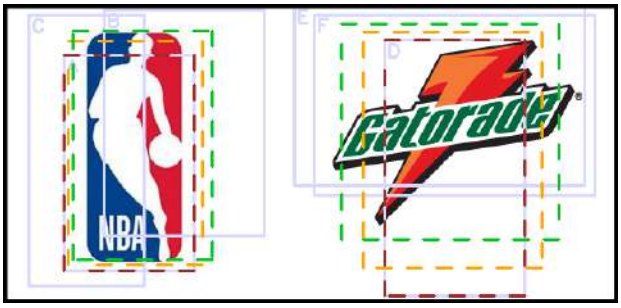

Figura 3.9: Exemplo dos retângulos envolventes selecionados pelos critérios

Conclusão: Observa-se que o método NMS não ajusta o resultado, mas apenas utiliza exatamente o mesmo retângulo com a maior confiabilidade. Já os métodos NMW e WBF ajustam o resultado, sendo que NMW realiza um ajuste mais suave que o WBF. Isso ocorre porque o WBF utiliza somente a confiabilidade no ajuste da localização dos retângulos envolventes, uma vez que o NMW multiplica o IoU na confiabilidade, fazendo com que quanto mais distante do retângulo envolvente principal do objeto, menor o peso para o ajuste.

3.7 Processamento em lotes

Normalmente as arquiteturas de redes neurais estão preparadas para serem executadas em lotes, com esse recurso as execuções possuem as seguintes vantagens:

- Otimiza o processamento de imagens em massa;
- Evita ociosidade do modelo entre um processo e outro;

- Executa apenas uma vez o programa para processar grandes quantidades de dados, otimizando mais o uso de memória e processador.

A desvantagem é que só é possível executar o próximo lote quando o lote em execução corrente for finalizado, gerando, assim, uma pequena latência.

Quando se pensa em tempo real é mandatório considerar o limite máximo da latência aceitável e a partir disso definir o tamanho do lote.

Este processamento não tem a funcionalidade de melhorar a acurácia de detecções ou as classificações dos objetos, mas sim melhorar o tempo de processamento dos métodos. Com isso, o processamento em lote será utilizado com os métodos propostos na medição do tempo de frames por segundo (fps), visualizando, assim, o ganho no tempo da execução dos métodos.

Na figura 3.10 segue um exemplo de um vídeo que possui uma característica de 30 (trinta) fps (frames por segundo) e a latência máxima de 2 (dois) segundos para o processamento. Deve-se criar um lote de no máximo 60 imagens (30 x 2), assim o processamento irá ser mais otimizado do que se processasse 60 imagens individualmente. Caso este processamento ocorra em menos de 2 (dois) segundos, o próximo lote provavelmente irá conter menos de 60 (sessenta) imagens.

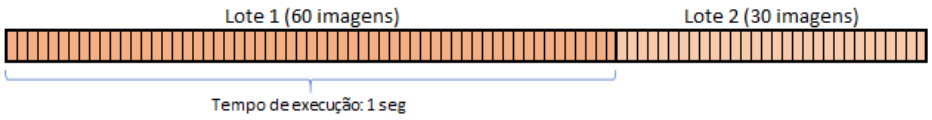

Figura 3.10: Quantidade máxima de frames com uma latência de 2 segundos

A latência permitida e o tamanho do lote precisam estar sempre casados com a configuração do video, caso o vídeo seja 30fps e a latência máxima de 5 segundos, o lote máximo precisa ser obrigatoriamente no máximo 150 (5 x 30) imagens. Assim não haverá atrasos nos processamentos.

3.8 Arquitetura Completa

Esta seção apresenta a proposta da implementação do fluxo completo utilizando lotes de imagens no modelo e nos métodos propostos nas seleções dos retângulos envolventes, com o intuito de aprimorar e otimizar os resultados. Isto irá permitir que a arquitetura de detecção de objetos seja processada com "N" imagens, possibilitando a classificação e detecção das imagens em uma única vez.

A implementação do fluxo completo dos métodos em lote é similar ao simples, a diferença é a quantidade de *inputs* que a rede neural irá processar. A figura 3.11 mostra as três etapas da arquitetura passando inputs em lotes.

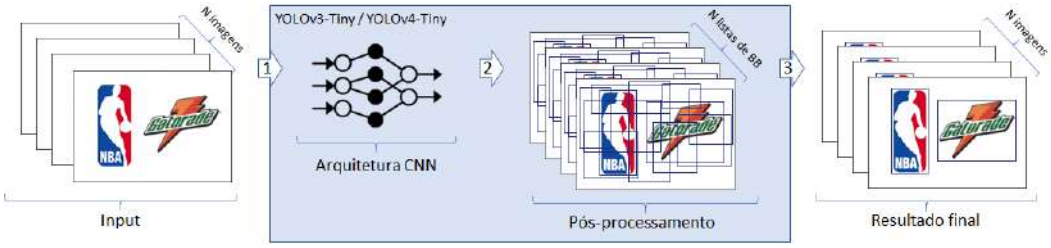

Figura 3.11: Etapas da arquitetura completa em lotes

Etapa 1: A rede neural convolucional (CNN) recebe "N" imagens para realizar as detecções.

Essas imagens irão se converter em uma matriz ($NxWxHx3$), sendo que "N" é a quantidade de imagens, "W" a largura e "H" a altura. Já o "3" indica as cores do RGB.

Esta nova estrutura de matriz será utilizada no modelo. Assim, a rede neural irá processar todas as imagens de uma vez só, agilizando o uso das CPUs.

Etapa 2: A rede neural convolucional realiza as detecções com centenas de possíveis resultados. A estrutura desse resultado será é um conjunto de "N" listas, pois cada posição do conjunto será referenciada a uma imagem.

Etapa 3: O pós-processamento também realiza uma etapa importante no processamento em lotes. Os dados retornados pelo modelo serão processados de uma vez só nos novos métodos adaptados para serem processados em lotes. A implementação desses métodos adaptados para lotes são apresentada nesta seção.

Os métodos desenvolvidos e aplicados nos testes foram implementados na linguagem Python[1] utilizando a biblioteca Pandas[2], pois ambos oferecem uma grande vantagem em tempo de execução nos agrupamentos dos *DataFrames*.

Os métodos em lote recebem um *DataFrame* através de um parâmetro contendo os retângulos envolventes com as oito propriedades a seguir:

- **XMin**, **YMin**, **XMax**, **YMax**: Os eixos X e Y dos dois pontos
- **ImageID**: Identificador da Imagem
- **LabelName**: Nome da classe do objeto
- **IoU**: Valor do IoU em relação ao objeto com maior confiabilidade
- **ObjectId**: Identificador que pertence ao mesmo objeto sobreposto da maior confiabilidade

[1]https://www.python.org/
[2]https://pandas.pydata.org/

As propriedades IoU e ObjectId precisam passar por um pré-processamento para serem pré-calculados. Como todos os métodos utilizam estas duas propriedades, eles não serão contabilizados no tempo, apenas o diferencial de cada método que será contabilizado. Assim, podemos calcular quanto há de incremento de *overhead* para cada método proposto em relação ao NMS.

Seguem como exemplos em Python a implementação dos métodos NMS no código 1, NMW no código 2 e WBF no código 3:

Código 1: Non-maximum Suppression (NMS) - Lote

```
import pandas as pd
def nms_lote(df):
    df = df.copy()
    df = df.sort_values(by='Conf', ascending=False)
    return df[~df.duplicated(subset=['ImageID', 'ObjectId', '
        LabelName'], keep='first')]
```

Código 2: Non-maximum weighted (NMW) - Lote

```
import pandas as pd
def nmw_lote(df):
    df = df.copy()
    df['w'] = df['Conf'] * df['IoU']
    df[['wXMin', 'wXMax', 'wYMin', 'wYMax']] = df[['XMin', 'XMax', '
        YMin', 'YMax']].multiply(df['Conf'], axis="index")
    df = df.groupby(by=['ImageID', 'ObjectId', 'LabelName']).agg(
        w=("w", 'sum'),
        XMin=("wXMin", 'sum'),
        XMax=("wXMax", 'sum'),
        YMin=("wYMin", 'sum'),
        YMax=("wYMax", 'sum')
    )
    df.reset_index(inplace=True)
    df[['XMin', 'XMax', 'YMin', 'YMax']] = df[['XMin', 'XMax', 'YMin
        ', 'YMax']].divide(df['w'], axis="index")
    df.drop('w', axis=1, inplace=True)
    return df
```

Código 3: Weighted boxes fusion (WBF) - Lote

```
import pandas as pd
def wbf_lote(df):
    df = df.copy()
    df[['XMin', 'XMax', 'YMin', 'YMax']] = df[['XMin', 'XMax', 'YMin
        ', 'YMax']].multiply(df['Conf'], axis="index")
    df = df.groupby(by=['ImageID', 'ObjectId', 'LabelName']).agg(
        ConfSum=("Conf", 'sum'),
        ConfCount=("Conf", 'count'),
```

```
8           XMin=("XMin", 'sum'),
9           XMax=("XMax", 'sum'),
10          YMin=("YMin", 'sum'),
11          YMax=("YMax", 'sum')
12      ).reset_index(drop=False)
13      df['Conf'] = df['ConfSum'] / df['ConfCount']
14      df[['XMin', 'XMax', 'YMin', 'YMax']] = df[['XMin', 'XMax', 'YMin
            ', 'YMax']].divide(df['ConfSum'], axis="index")
15      df.drop(['ConfSum','ConfCount'], axis=1, inplace=True)
16      return df
```

4
Resultados

Neste capítulo serão apresentados diferentes testes com os métodos propostos, com o intuito de comparar seus desempenhos utilizando as métricas mAP, *Recall* e *Precision*, tomando como base o *benchmark* gerado pelo método NMS.

Para avaliar o critério dos métodos propostos, será necessário considerar quatro importantes configurações em relação ao resultado mAP e matriz de confusão, tais como:

- **Limite de confiabilidade vs mAP:** Visualização do comportamento da métrica mAP na interpolação do limite de confiabilidade;
- **Limite do IoU vs mAP:** Visualização do comportamento da métrica mAP na interpolação do limite do IoU;
- **Matriz de confusão vs IoU:** Visualização do comportamento da matriz de confusão na interpolação do limite do IoU;
- **Tempo de execução:** Medição o tempo de execução com diferentes quantidades de retângulos envolventes.

Os testes desta pesquisa foram realizados em servidores com GPU Tesla T4 e 12 GB de RAM.

4.1
Dataset

O *dataset* utilizado nesta dissertação foi o QMUL-OpenLogo (Hang Su et al., 2018) que já existe no mercado e é utilizado em artigos acadêmicos, porém pode-se utilizar qualquer outro tipo de dataset. O dataset QMUL-OpenLogo é constituído por um conjunto de 7 (sete) outros *datasets* públicos, totalizando 27.083 imagens com 352 logotipos. Este *dataset* possui um conjunto extra de imagens em que os logotipos foram adicionados artificialmente, com intuito de agregar grandes quantidades de dados no treinamento.

Com base neste *dataset*, foram utilizados 3 (três) logotipos, sendo eles: Coca-cola, Pepsi e Shell. Segue um exemplo das imagens na figura 4.1

Cada logotipo possui 2 (dois) conjuntos de imagens, um para treinamento e outro para teste, com quantidades diferentes em cada conjunto. Seguem as quantidades de cada conjunto na tabela 4.1:

Figura 4.1: Imagens do *dataset* QMUL-OpenLogo

Conjunto	**Coca-Cola**	**Pepsi**	**Shell**
Treino	510	270	232
Teste	219	116	100

Tabela 4.1: Quantidade de imagens de teste e treinamento

Embora tenha uma diferença na quantidade de imagens para cada logotipo, foram utilizadas técnicas de *data augmentation*, criando imagens a partir de outras, para conhecimento de padrões no modelo de detecção de objetos. As técnicas utilizadas foram:

- **GaussNoise**: Adicionar ruídos aleatoriamente. (porcentagem entre 5% a 10%)
- **ShiftScaleRotate**: Rotacionar e aplicar zoom aletoriamente. (Zoom de até 0.1 e rotacionando entre -15º e +15º)
- **HueSaturationValue**: Alterar aleatoriamente o matiz e a saturação. (Alteração do matiz e saturação de até 10%)
- **RandomContrast**: Alterar aleatoriamente o contraste. (Alteração do fator de contraste de até 40%)
- **RandomBrightness**: Alterar aleatoriamente o brilho. (Alteração do fator do brilho de até 20%)

As técnicas citadas tiveram 50% de chances de serem aplicadas nas imagens.

A figura 4.2 contém um exemplo de cada técnica aplicada.

Nos treinamentos dos modelos foram utilizadas as mesmas classes de objeto, técnicas de *data argumentation* e conjunto de treinamento e testes.

Figura 4.2: Exemplos de *Data Augmentation*

4.2 Limite de confiabilidade vs Mean Average Precision

Pode-se observar, nas tabelas 4.2 e 4.3, os valores do mAP com a interpolação dos limites de confiabilidade variando de 0.30 a 0.95 com intervalos de 0.05. O limite de IoU foi fixado em 0.3 na tabela 4.2 e na tabela 4.3 o limite de IoU foi fixado em 0.5.

Conf.	**YOLOv3 Tiny**				**YOLOv4 Tiny**			
	NMS	**NMW**	**WBF**	**Ganho**	**NMS**	**NMW**	**WBF**	**Ganho**
0,30	50,88%	51,94%	51,08%	**2,07%**	59,27%	64,05%	63,97%	**8,06%**
0,35	50,14%	51,40%	50,93%	**2,51%**	57,82%	60,97%	59,18%	**5,45%**
0,40	48,66%	49,43%	49,60%	**1,93%**	56,82%	59,28%	58,40%	**4,34%**
0,45	45,80%	46,26%	46,39%	**1,30%**	54,68%	56,74%	56,29%	**3,77%**
0,50	43,76%	44,21%	44,36%	**1,37%**	51,09%	52,32%	51,62%	**2,40%**
0,55	37,56%	38,21%	38,44%	**2,34%**	47,37%	47,95%	47,42%	**1,23%**
0,60	34,75%	35,75%	36,33%	**4,55%**	43,78%	44,65%	44,05%	**2,00%**
0,65	31,15%	32,07%	32,26%	**3,56%**	38,63%	39,32%	39,01%	**1,77%**
0,70	27,21%	27,21%	27,10%	**0,00%**	31,97%	32,61%	32,36%	**2,00%**
0,75	22,35%	22,35%	22,10%	**0,00%**	22,88%	23,47%	23,34%	**2,60%**
0,80	16,59%	16,59%	16,52%	**0,00%**	16,63%	17,15%	17,11%	**3,13%**
0,85	10,11%	10,11%	10,15%	**0,46%**	10,46%	10,46%	10,45%	**0,00%**
0,90	4,14%	4,14%	4,14%	**0,00%**	4,18%	4,18%	4,18%	**0,00%**
0,95	0,56%	0,56%	0,56%	**0,00%**	1,11%	1,11%	1,11%	**0,00%**
Média	**32,95%**	**33,91%**	**34,30%**	**1,33%**	**41,21%**	**41,99%**	**41,53%**	**2,20%**

Tabela 4.2: Com limite de confiabilidade entre 0.30 e 0.95 e com limite do IoU fixado em 0.30

Os resultados de todos os cenários das tabelas 4.2 e 4.3 ratificaram que as duas redes convolucionais e os limites do IoU fixado em 0.3 e 0.5 possuíram melhores resultados no mAP usando os métodos de critérios WBF e NMW em relação ao NMS (método padrão) com limites de confiabilidade menores.

Na tabela 4.2 com o IoU fixado em 0.3, o YOLOv3-Tiny obteve ganhos de até 4.55% e o YOLOv4-Tiny um ganho de até 8.06%.

Na tabela 4.3 com o IoU fixado em 0.5, o YOLOv3-Tiny obteve ganhos de até 2.18% e o YOLOv4-Tiny um ganho de até 6.49%.

Conf.	YOLOv3 Tiny				YOLOv4 Tiny			
	NMS	**NMW**	**WBF**	**Ganho**	**NMS**	**NMW**	**WBF**	**Ganho**
0,30	50,83%	52,01%	51,81%	**2,32%**	60,52%	64,45%	63,40%	**6,49%**
0,35	50,00%	51,55%	51,11%	**3,11%**	58,61%	61,17%	58,86%	**4,38%**
0,40	48,62%	49,68%	50,04%	**2,92%**	57,45%	59,64%	58,43%	**3,80%**
0,45	45,89%	46,64%	46,62%	**1,64%**	55,12%	56,96%	56,52%	**3,33%**
0,50	43,67%	44,40%	44,62%	**2,18%**	51,32%	52,26%	51,55%	**1,83%**
0,55	37,56%	38,21%	38,44%	**2,35%**	47,37%	47,95%	47,42%	**1,23%**
0,60	34,75%	35,75%	36,33%	**4,55%**	43,78%	44,65%	44,05%	**2,00%**
0,65	31,15%	32,07%	32,26%	**3,56%**	38,63%	39,32%	39,01%	**1,77%**
0,70	27,21%	27,21%	27,10%	**0,00%**	31,97%	32,61%	32,36%	**2,00%**
0,75	22,35%	22,35%	22,10%	**0,00%**	22,88%	23,47%	23,34%	**2,60%**
0,80	16,59%	16,59%	16,52%	**0,00%**	16,63%	17,15%	17,11%	**3,13%**
0,85	10,11%	10,11%	10,15%	**0,46%**	10,46%	10,46%	10,45%	**0,00%**
0,90	4,14%	4,14%	4,14%	**0,00%**	4,18%	4,18%	4,18%	**0,00%**
0,95	0,56%	0,56%	0,56%	**0,00%**	1,11%	1,11%	1,11%	**0,00%**
Média	**32,95%**	**33,91%**	**34,30%**	**1,91%**	**41,21%**	**41,99%**	**41,53%**	**2,00%**

Tabela 4.3: Com limite de confiabilidade entre 0.30 e 0.95 e com limite do IoU fixado em 0.50

Vale ressaltar que os melhores resultados estão concentrados entre 0.3 e 0.6 no limite de confiabilidade, como destacado nas tabelas 4.2 e 4.3, pois com uma confiabilidade mais baixa, os critérios NMW e WBF possuirão mais retângulos envolventes para refinar os seus resultados.

A tabela 4.4 apresenta um resumo dos menores e maiores ganhos no intervalo entre 0.3 e 0.6 no limite de confiabilidade.

	YOLOv3-Tiny		YOLOv4-Tiny	
IoU	Menor ganho	Maior ganho	Menor ganho	Maior ganho
0.3	1.30%	4.55%	1.23%	8.06%
0.5	1.64%	4.55%	1.23%	6.49%

Tabela 4.4: Maiores e menores ganhos entre 0.3 e 0.6 no limite de confiabilidade

Em vista dos resultados obtidos, verifica-se que as arquiteturas de detecção de objeto da classe logotipos não necessitam possuir uma confiabilidade alta para dizer que o objeto foi detectado da forma correta, pois o mínimo de detalhe já é suficiente para detectar e classificar um logotipo.

Por exemplo, o nível de detalhe de uma classe logotipo não é tão complexo quanto o nível de detalhamento de uma classe animal, já que um animal pode estar em diversas posições e também pode estar realizando inúmeras ações. A classe logotipo é estática, uma vez que, geralmente, só existe um único formato por marca.

Segue um exemplo de resultado com alto (Figura 4.3) e baixo (Figura 4.4) limites de confiabilidade, o que demonstra que algumas detecções de logotipos

não são identificadas quando aumentamos a confiabilidade, ocasionando FN (*False Negative*).

Figura 4.3: Limites: Conf. 0.7 | IoU = 0.5

Figura 4.4: Limites: Conf. 0.3 | IoU = 0.5

4.3 Intersect over Union vs Mean Average Precision

Pode-se observar os valores do mAP com a interpolação do limite do IoU variando de 0.30 a 0.95 com intervalos de 0.05, na tabela 4.5 o limite de confiabilidade foi fixado em 0.3 e na tabela 4.6 o limite de confiabilidade foi fixada em 0.5.

IoU	YOLOv3 Tiny				YOLOv4 Tiny			
	NMS	NMW	WBF	Ganho	NMS	NMW	WBF	Ganho
0,30	50,88%	51,94%	51,08%	**2,07%**	59,27%	64,05%	63,97%	**8,06%**
0,35	50,86%	51,91%	51,00%	**2,07%**	59,63%	64,43%	64,10%	**8,05%**
0,40	50,53%	52,12%	51,32%	**3,14%**	59,78%	64,14%	64,28%	**7,52%**
0,45	50,90%	52,31%	51,75%	**2,77%**	59,72%	64,26%	63,69%	**7,60%**
0,50	50,83%	52,01%	51,81%	**2,32%**	60,52%	64,45%	63,40%	**6,49%**
0,55	50,29%	51,93%	50,85%	**3,27%**	60,03%	64,26%	62,43%	**7,05%**
0,60	50,27%	51,45%	49,28%	**2,34%**	59,05%	62,66%	60,68%	**6,12%**
0,65	49,53%	50,08%	45,74%	**1,11%**	58,36%	60,72%	59,02%	**4,05%**
0,70	47,18%	47,77%	43,28%	**1,26%**	56,83%	58,00%	55,96%	**2,05%**
0,75	44,66%	44,80%	40,85%	**0,32%**	54,66%	54,69%	52,29%	**0,05%**
0,80	41,10%	41,24%	39,39%	**0,36%**	49,93%	49,95%	48,50%	**0,04%**
0,85	37,49%	37,51%	36,11%	**0,05%**	46,57%	46,72%	45,19%	**0,32%**
0,90	34,55%	34,55%	33,72%	**0,00%**	42,85%	42,85%	42,27%	**0,00%**
0,95	32,95%	32,95%	33,05%	**0,31%**	40,63%	40,63%	40,54%	**0,00%**
Média	**49,90%**	**50,76%**	**47,51%**	**1,67%**	**58,71%**	**61,69%**	**59,85%**	**5,09%**

Tabela 4.5: Com limite de IoU entre 0.30 e 0.95 e com limite de confiabilidade fixado em 0.30

Os resultados da tabela 4.5, que possuem o limite da confiabilidade fixado em 0.3, mostram que as duas arquiteturas testadas apresentaram melhoras significativas para IoU com limites baixos e com os critérios WBF e NMW comparando com NMS, com um ganho de até 3.27% no YOLOv3-Tiny e de até 8.06% no YOLOv4-Tiny. Isso ocorre, pois quanto menor o limite IoU, mais informações de retângulos envolventes serão atribuídas a um mesmo

IoU	YOLOv3 Tiny				YOLOv4 Tiny			
	NMS	NMW	WBF	Ganho	NMS	NMW	WBF	Ganho
0,30	43,76%	44,21%	44,36%	**1,37%**	51,09%	52,32%	51,62%	**2,40%**
0,35	43,76%	44,21%	44,36%	**1,37%**	51,09%	52,32%	51,62%	**2,40%**
0,40	43,74%	44,47%	44,62%	**2,00%**	51,09%	52,32%	51,62%	**2,40%**
0,45	43,74%	44,47%	44,62%	**2,00%**	51,09%	52,31%	51,63%	**2,40%**
0,50	43,67%	44,40%	44,62%	**2,18%**	51,32%	52,26%	51,55%	**1,83%**
0,55	43,20%	43,91%	43,95%	**1,75%**	50,99%	51,94%	51,27%	**1,85%**
0,60	43,33%	43,60%	43,62%	**0,68%**	50,12%	51,13%	50,09%	**2,01%**
0,65	42,92%	42,91%	42,55%	**-0,01%**	49,56%	50,26%	49,23%	**1,41%**
0,70	41,53%	41,23%	40,85%	**-0,72%**	48,30%	48,99%	47,93%	**1,43%**
0,75	39,53%	39,65%	39,28%	**0,30%**	46,79%	46,59%	45,67%	**-0,42%**
0,80	36,89%	36,89%	36,09%	**0,00%**	43,21%	43,04%	42,42%	**-0,41%**
0,85	33,86%	33,86%	33,23%	**0,00%**	40,74%	40,59%	39,52%	**-0,39%**
0,90	31,22%	31,22%	30,70%	**0,00%**	37,47%	37,47%	37,13%	**0,00%**
0,95	29,74%	29,74%	29,85%	**0,34%**	35,48%	35,48%	35,37%	**0,00%**
Média	**43,06%**	**43,26%**	**43,09%**	**0,51%**	**49,84%**	**50,69%**	**49,66%**	**1,63%**

Tabela 4.6: Com limite de IoU entre 0.30 e 0.95 e com limite de confiabilidade fixado em 0.50

objeto, assim os critérios WBF e NMW possuem mais recursos para ajustar os resultados.

Na tabela 4.6 o limite da confiabilidade foi fixado em 0.5. Com isso, os resultados são inferiores, com um ganho de até 2.18% no YOLOv3-Tiny e de até 2.40% no YOLOv4-Tiny. Dessa forma, com o aumento do limite de confiabilidade, há uma diminuição da quantidade de retângulos para os critérios de WBF e NMW.

Cabe destacar que os resultados do IoU com um limite alto revelaram ganhos negativos, evidenciando que os resultados do critério NMS apresentaram um melhor desempenho.

Vale ressaltar que os melhores resultados estão concentrados entre 0.3 e 0.6 no limite de confiabilidade, seguindo o mesmo comportamento do capítulo anterior.

A tabela 4.7 exibe um resumo dos menores e maiores ganhos no intervalo 0.3 e 0.6 no limite do IoU.

	YOLOv3-Tiny		YOLOv4-Tiny	
IoU	Menor ganho	Maior ganho	Menor ganho	Maior ganho
0.3	2.07%	3.27%	6.12%	8.06%
0.5	0.68%	2.18%	1.83%	2.40%

Tabela 4.7: Maiores e menores ganhos entre 0.3 e 0.6 no limite de IoU

Segue um exemplo de resultado com alto (Figura 4.5) e baixo (Figura 4.5) limites de confiabilidade, o que mostra que algumas detecções de logotipos são

identificadas em maior quantidade para o mesmo objeto quando aumentamos o IoU, ocasionando FP (False Positive).

Figura 4.5: Limites: Conf. 0.3 | IoU = 0.7

Figura 4.6: Limites: Conf. 0.3 | IoU = 0.3

4.4 Matriz de confusão vs Intersect over Union

Observa-se a evolução da matriz de confusão com os seus valores *Recall* e *Precision* na interpolação do limite do IoU de 0.30 a 0.95 com intervalos de 0.05 na tabela 4.8 e no gráfico 4.7 com o limite de confiabilidade fixado em 0.3. Já na tabela 4.9 e no gráfico 4.8, o limite de confiabilidade foi fixado em 0.5.

IoU	TP			FP			FN		
	NMS	NMW	WBF	NMS	NMW	WBF	NMS	NMW	WBF
0,30	239	251	256	151	139	134	120	108	103
0,35	241	253	257	154	142	138	118	106	102
0,40	242	253	258	157	146	141	117	106	101
0,45	242	254	258	167	155	154	117	105	101
0,50	246	256	259	186	176	165	113	103	100
0,55	247	258	259	213	202	190	112	101	100
0,60	249	258	259	247	238	224	110	101	100
0,65	252	259	261	302	295	278	107	100	98
0,70	257	260	261	404	401	381	102	99	98
0,75	261	261	261	514	514	517	98	98	98
0,80	261	261	261	656	656	656	98	98	98
0,85	261	262	262	777	776	777	98	97	97
0,90	262	262	262	864	864	864	97	97	97
0,95	262	262	262	898	898	898	97	97	97

Tabela 4.8: Matriz de confusão com interpolação do limite do IoU de 0.30 a 0.95 e limite de confiabilidade igual a 0.3

Pela análise dos resultados da tabela 4.8, percebe-se um crescimento considerável dos valores TP (*True Positive*) no intervalo de limite entre 0.3 e 0.7 do IoU, com uma média de 14 detecções a mais no método WBF em relação ao NMS. No gráfico 4.7 pode-se perceber que o NMS do *Recall* tem uma redução mais acelerada em comparação aos demais métodos, isso ocorre

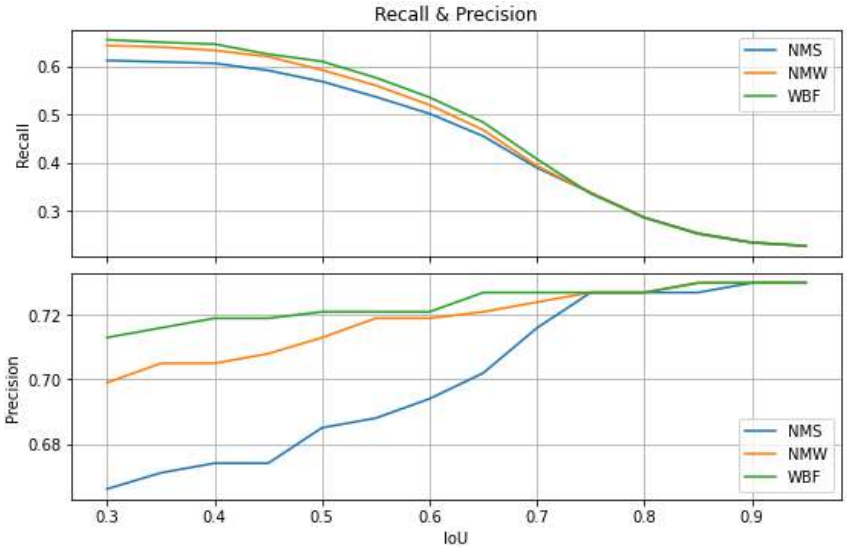

Figura 4.7: Comportamento do *recall* e *precision* baseado na matriz de confusão com interpolação do limite do IoU de 0.30 a 0.95 e limite de confiabilidade igual a 0.3.

porque os FN (*False Positive*) são maiores que os outros métodos, de acordo com a tabela 4.8. Analisando o *Precision*, o método NMS tem um crescimento menos acelerado dentre os demais métodos, pois os valores FP (*False Positive*) são maiores.

IoU	TP			FP			FN		
	NMS	NMW	WBF	NMS	NMW	WBF	NMS	NMW	WBF
0,30	200	202	202	44	42	42	159	157	157
0,35	200	202	202	44	42	42	159	157	157
0,40	200	202	202	44	42	42	159	157	157
0,45	200	202	202	45	43	43	159	157	157
0,50	201	202	202	45	44	44	158	157	157
0,55	201	202	202	52	51	49	158	157	157
0,60	201	202	202	60	59	55	158	157	157
0,65	202	203	202	71	70	66	157	156	157
0,70	202	203	202	86	85	81	157	156	157
0,75	204	203	203	118	119	122	155	156	156
0,80	204	203	203	170	171	170	155	156	156
0,85	204	203	203	226	227	227	155	156	156
0,90	204	204	204	273	273	273	155	155	155
0,95	204	204	204	296	296	296	155	155	155

Tabela 4.9: Matriz de confusão com interpolação do limite do IoU de 0.30 a 0.95 e limitando a confiabilidade a 0.5

Pela análise dos resultados da tabela 4.9, percebe-se um crescimento sutil dos valores TP (*True Positive*) no intervalo de limite entre 0.3 e 0.6 do IoU, com uma média de 1.5 detecções a mais nos métodos WBF e NMW em relação ao NMS. No gráfico 4.8 pode-se perceber que o NMS do *Recall* obteve valores

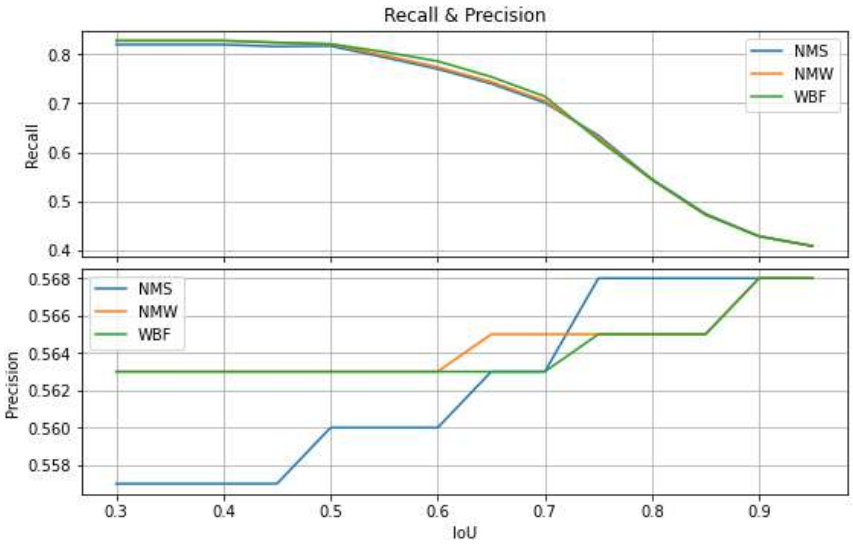

Figura 4.8: Comportamento do *recall* e *precision* baseado na matriz de confusão com interpolação do limite do IoU de 0.30 a 0.95 e limitando a confiabilidade a 0.5.

bem próximos aos métodos WBF e NMW, isso ocorre porque os FN (*False Positive*) apresentaram pouca diferença.

4.5 Tempo de execução

4.5.1 Tempo de execução vs Limite de Confiabilidade

Observa-se no gráfico da figura 4.9 a evolução do tempo de execução utilizando diferentes tipos de critérios de seleção dos retângulos envolventes. Os limites de interpolação foram testados com uma interpolação variando entre 0.30 a 0.95 com intervalos de 0.05 e o IoU fixado em 0.5.

Figura 4.9: Tempo de execução dos critérios de seleção de retângulos envolventes

O gráfico da figura 4.9 demonstra que o tempo de execução do NMS é aproximadamente 10% menor que os demais métodos testados. No processamento de 400 retângulos envolventes, o NMS atingiu aproximadamente 0.0025

segundos de processamento, enquanto os demais métodos executaram em 0.025 segundos.

Vale observar que o tempo é praticamente uma constante para cada método quando se utilizam diferentes limites de confiabilidade, já que o *overhead* é quase inexistente no processamento com uma quantidade de até 400 retângulos envolventes testadas.

4.5.2 Processamento em lotes

Nesta seção será calculado o tempo de execução dos critérios em lotes, citados no capítulo 3.7, com o objetivo de verificar se há a possibilidade de usá-los em processamentos tempo real.

Para tanto, na tabela 4.10, será calculada a quantidade de imagens por segundo em processamentos de 7 tamanhos diferentes de lotes, sendo eles: lotes com 1, 15, 30, 45, 60 e 180 imagens que são equivalentes a 1/30, 1/2, 1, 4/3, 2 e 6 segundos de latência no processamentos, com uma configuração de 30 fps. Estes experimentos serão analisados em um vídeo de 2 minutos, totalizando 3600 imagens.

Lote de Imagens	Imagens por segundo		
	NMS	NMW	WBF
1	453	35	35
15	5598	533	536
30	9353	1007	1073
45	12533	1515	1561
60	14183	1990	2044
180	26076	5600	5192

Tabela 4.10: Tempos dos processamentos dos critérios dos retângulos envolventes

Verifica-se que para um lote contendo apenas uma imagem, ou seja, o processamento uma imagem por vez, o NMS é superior na quantidade de imagens processadas por segundo, processando em média 10 vezes mais que os demais métodos. Em um comparativo, por conta do desempenho bastante inferior, os métodos NMW e WBF tornam-se impossíveis de serem aplicados em tempo real, uma vez que ficariam no limite da quantidade de imagens processadas para serem considerados tempo real. Contudo, considerando processamentos com latência, o tempo cai consideravelmente. Por exemplo, quando os métodos NMW e WBF utilizam um lote de 15 imagens, o tempo se equipara com a execução de NMS com um lote de apenas uma imagem. Segundo o capítulo 2.3, podemos usar uma latência de até 6 segundos para ainda ser considerada

uma latência baixa e, assim, o tempo de processamento dos métodos NMW e WBF se tornaria possível em tempo real.

4.6 Discussão

A maioria das redes convolucionais para detecção e classificação de objetos usa a saída NMS ou Soft-NMS como critério de retângulos envolventes. Esta dissertação propôs verificar se seria possível usar WBF ou NMW nas saídas dos modelos ao invés do NMS. Para o experimento, utilizou-se YOLOv3-Tiny e YOLOv4-Tiny com um conjunto de dados QMUL-OpenLogo. Ao desabilitar a camada NMS, obteve-se uma saída com mais de 100 retângulos envolventes contendo suas confiabilidades.

Consolidando os dados dos testes realizados nas seções 4.2 e 4.3, como mostra a figura 4.10, os ganhos obtidos com o critério NMW em relação ao NMS foram de até 8.06%.

YOLOv4 (Ganho - NMS -> NMW)

Score Threshold \ IoU Threshold	0,30	0,35	0,40	0,45	0,50	0,55	0,60	0,65	0,70	0,75	0,80	0,85	0,90	0,95
0,95	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
0,90	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
0,85	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
0,80	3,13%	3,13%	3,13%	3,13%	3,13%	3,13%	3,63%	3,65%	3,68%	0,00%	0,00%	0,00%	0,00%	0,00%
0,75	2,60%	2,60%	2,60%	2,60%	2,60%	2,60%	2,96%	2,98%	3,00%	0,00%	0,00%	0,00%	0,00%	0,00%
0,70	2,00%	2,00%	2,00%	2,00%	2,00%	2,02%	2,26%	2,27%	2,30%	0,00%	0,00%	0,00%	0,00%	0,00%
0,65	1,77%	1,77%	1,77%	1,77%	1,77%	1,78%	1,98%	1,99%	2,00%	0,00%	0,00%	0,00%	0,00%	0,00%
0,60	2,00%	2,00%	2,00%	2,00%	2,00%	2,02%	2,20%	1,09%	1,11%	-0,71%	-0,70%	-0,68%	0,00%	0,00%
0,55	1,23%	1,23%	1,23%	1,23%	1,23%	1,25%	1,41%	0,35%	0,38%	-0,70%	-0,69%	-0,68%	0,00%	0,00%
0,50	2,40%	2,40%	2,40%	2,40%	1,83%	1,85%	2,01%	1,41%	1,43%	-0,42%	-0,41%	-0,39%	0,00%	0,00%
0,45	3,77%	3,77%	3,77%	3,77%	3,33%	3,32%	3,48%	1,78%	1,79%	-0,42%	-0,42%	-0,39%	0,00%	0,00%
0,40	4,34%	4,34%	4,34%	4,25%	3,80%	3,79%	3,94%	2,14%	1,31%	-0,42%	-0,42%	-0,40%	0,00%	0,00%
0,35	5,45%	5,45%	5,45%	5,36%	4,38%	4,36%	3,89%	1,84%	1,05%	-0,66%	-0,64%	-0,34%	0,00%	0,00%
0,30	8,06%	8,05%	7,29%	7,60%	6,49%	7,05%	6,12%	4,05%	2,05%	0,05%	0,04%	0,32%	0,00%	0,00%

Figura 4.10: Ganhos consolidados do YOLOv4 com NMW

Cabe destacar que os resultados do IoU com um limite alto revelaram ganhos negativos, evidenciando que os resultados do critério NMS apresentaram um melhor desempenho.

Também é possível constatar que em todos os testes os maiores ganhos estão no intervalo entre 0.30 e 0.70 dos limites de confiabilidade e IoU, tanto no YOLOv3-Tiny e YOLOv4-Tiny. No YOLOv3-Tiny os ganhos ficaram entre 1.3% a 4.55% com o IoU 0.3 e 1.64% a 4.55% com o IoU 0.5. No YOLOv4.Tiny os ganhos ficaram entre 1.23% a 8.06% com o IoU 0.3 e 1.23% a 6.49% com o IoU 0.5.

No geral, os critérios WBF e NMW apresentaram melhores resultados sendo executados com baixos limites de confiabilidades ou baixos limites no

IoU, porque os critérios NMS e Soft-NMS excluem alguns retângulos, mas, por outro lado, os métodos WBF e NMW usam configurações dos retângulos envolventes do mesmo objeto. Assim, é possível corrigir a imprecisão de alguns retângulos envolventes previstos. O NMS deixará apenas o retângulo envolvente impreciso com uma maior confiabilidade, enquanto o WBF e o NMW ajustam a precisão usando informações dos demais retângulos.

Na seção 4.4 é possível perceber que os resultados obtiveram um crescimento considerável dos valores TP (*True Positive*) e diminuição de FP (*False Negative*) nos valores limite entre 0.3 e 0.7 do IoU, com uma média de 14 detecções a mais nos métodos WBF e NMW em relação ao NMS. Tal comportamento, fez com que o Recall e Precision tenham uma diferença positiva nos novos valores utilizando os critérios WBF e NMW, como pode-se observar na tabela 4.11.

IoU	**Recall**			**Precision**		
	NMS	**NMW**	**WBF**	**NMS**	**NMW**	**WBF**
0,30	0,613	0,644	0,656	0,666	0,699	0,713
0,35	0,610	0,641	0,651	0,671	0,705	0,716
0,40	0,607	0,634	0,647	0,674	0,705	0,719
0,45	0,592	0,621	0,626	0,674	0,708	0,719
0,50	0,569	0,593	0,611	0,685	0,713	0,721
0,55	0,537	0,561	0,577	0,688	0,719	0,721
0,60	0,502	0,520	0,536	0,694	0,719	0,721
0,65	0,455	0,468	0,484	0,702	0,721	0,727
0,70	0,389	0,393	0,407	0,716	0,724	0,727
0,75	0,337	0,337	0,335	0,727	0,727	0,727
0,80	0,285	0,285	0,285	0,727	0,727	0,727
0,85	0,251	0,252	0,252	0,727	0,730	0,730
0,90	0,233	0,233	0,233	0,730	0,730	0,730
0,95	0,226	0,226	0,226	0,730	0,730	0,730

Tabela 4.11: Valores recall e precision do YOLOv4 com limite de confiabilidade 0.3

Dependendo do cenário, a métrica da matriz de confusão pode ser mais relevante, pois um FP (*False Positive*) pode se destacar de maneira negativa em comparação com a quantidade de FN (*False Negative*), exigindo um valor do *Precision* mais elevado. Cada caso pode requerer uma configuração personalizada e mais adequada às necessidades do negócio, não existindo, assim, uma regra fixa.

Entretanto, a seção 4.5.1 apresenta uma piora no tempo de execução para os critérios NMW e WBF, pois a velocidade depende da quantidade do conjunto de retângulos envolventes e da quantidade de classes. Os algoritmos WBF e NMW são, em média, cerca de dez vezes mais lentos do que o NMS

padrão, impossibilitando a execução em tempo real. No entanto, é possível contornar este problema executando os métodos com lotes de imagens, como mostra na seção 4.5.2. A quantidade de imagens em lotes irá depender no tamanho da latência máxima definida.

Tendo em vista que as classes de logotipo normalmente não se sobrepõem uma nas outras, diferentemente da maioria das outras classes, as detecções dos logotipos tendem a apresentar melhores resultados com o limite do IoU mais baixo. Por exemplo, o "objeto cachorro", por sua vez, pode estar sentado, pulando, nadando e até mesmo brincando com outro cachorro. Na imagem 4.11, pode-se detectar facilmente 2 cachorros juntos, o que se difere notoriamente de um logotipo, visto que os logotipos possuem outra perspectiva, pois normalmente encontram-se estampados em uma embalagem ou até mesmo em um outdoor e, dificilmente, aparecem sobrepostos a outro, como é possível observar na imagem 4.12.

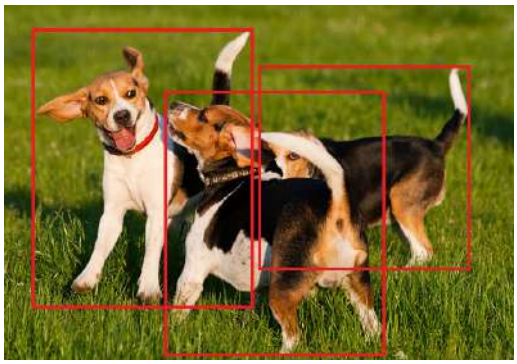

Figura 4.11: Retângulos envolventes sobrepostos

Figura 4.12: Retângulos envolventes de logotipos

5
Conclusão

Esta dissertação viabilizou o uso de uma nova técnica para combinar e selecionar retângulos envolventes nos modelos de detecção de objetos. Com isso, foi possível proporcionar ganhos no mAP sem a necessidade de desenvolver novas arquiteturas de redes neurais convolucionais e nem retreinar os modelos já disponíveis. Determinados ajustes realizados na etapa de pós-processamento, que otimizaram a seleção dos retângulos envolventes finais, foram decisivos para permitir a utilização de arquiteturas já existentes de redes convolucionais consideradas estados da arte, gerando, assim, grandes benefícios no tempo de processamento e também e na redução de gastos, uma vez que os retreinos possuem custos elevados.

Assim, com modificação apenas dos critérios de seleções de retângulos envolventes foi possível gerar ganhos de até 8.06% e ainda manter grandes quantidades de execuções por segundo, sem gerar um *overhead*, possibilitando o processamento em tempo real. Normalmente esses critérios de seleção não são levados em consideração na etapa de treinamento, o que permite o aproveitamento de diversas arquiteturas com modelos pré-treinados.

Os critérios utilizados nesta dissertação foram oriundos de publicações recentes, algumas, inclusive, com diferentes propósitos. Dessa forma, esta diversidade de recursos proporcionou soluções mais assertivas para uma melhor acurácia das detecções e classificações dos objetos.

Para tal, foram realizados vários testes utilizando duas arquiteturas apropriadas para tempo real e aparelhos embarcados, sendo elas: YOLOv3-Tiny e YOLOv4-Tiny, objetivando trazer um aumento de acurácia e desempenho. Geralmente, processamentos realizados em tempo real utilizam arquiteturas simples e leves, que não apresentam um bom desempenho no mAP. Entretanto, as arquiteturas mais pesadas apresentam uma precisão superior no mAP, mas não são apropriadas para tempo real, porque o tempo de execução é elevado.

Nesta dissertação foram realizados testes de processamento em lotes nos métodos NMW e WBF, sendo possível observar ganhos de até 200% em lotes de 15 imagens. Assim superando o tempo de execução do método NMS.

Como descrito na seção 4.1, o *dataset* utilizado para realização dos experimentos foi baseado na classe logotipos, sendo necessário realizar treinamentos nas arquiteturas utilizadas nos experimentos. Com isso, a qualidade do resultado está indiretamente relacionada à etapa do treinamento, pois existem técnicas que podem influenciar no resultado, como por exemplo: *data augmen-*

tation, tamanho em *batchs*, ajustes de parâmetros, etc. Para manter um padrão, a etapa de treinamento foi igualmente utilizada nas arquiteturas testadas.

Esta dissertação foi focada em arquiteturas com características baseadas em tempo real (mais que 30 fps) e detecção de objetos da classe logotipos, mas também pode se expandida para outras arquiteturas de detecção de objetos e outras classes de objetos, não necessariamente logotipos.

5.1 Trabalhos Futuros

Esta dissertação utilizou apenas os sistemas YOLOv3-Tiny e YOLOv4-Tiny, mas há possibilidades de realizar testes da arquitetura completa em outros sistemas para calcular a performance. Uma recomendação seria utilizar o dataset COCO para realizar testes com outros tipos de classes.

Os testes foram realizados utilizando altenativas dos métodos de seleçao dos retângulos envolventes, mas outra proposta seria substituir o método IoU pelo GIoU (Rezatofighi et al., 2019) ou CDIoU (Chen & Miao, 2021) e verificar se os resultados apresentam ganhos.

Também existe um espaço para testar outros critérios para reduções da confiabilidade e ajustes das coordenadas, utilizando, por exemplo, a distância de Manhattan, médias geométricas e médias harmônicas.

6
Referências bibliográficas

[Bochkovskiy et al., 2020] BOCHKOVSKIY, A.; WANG, C.-Y. ; LIAO, H.-Y. M.. **Yolov4: Optimal speed and accuracy of object detection**, 2020. 1.2, 2.2, 3.2

[Bodla et al., 2017] BODLA, N.; SINGH, B.; CHELLAPPA, R. ; DAVIS, L. S.. **Improving object detection with one line of code**. CoRR, abs/1704.04503, 2017. 3.2, 3.3

[Bombonato et al., 2017] BOMBONATO, L.; CAMARA-CHAVEZ, G. ; SILVA, P.. **Real-time single-shot brand logo recognition**. In: 2017 30TH SIBGRAPI CONFERENCE ON GRAPHICS, PATTERNS AND IMAGES (SIBGRAPI). IEEE, Oct. 2017. 1.5

[Chen & Miao, 2021] CHEN, D.; MIAO, D.. **Control distance iou and control distance iou loss function for better bounding box regression**, 2021. 5.1

[Chengcheng Ning et al., 2017] NING, C.; ZHOU, H.; SONG, Y. ; TANG, J.. **Inception single shot MultiBox detector for object detection**. In: 2017 IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA & EXPO WORKSHOPS (ICMEW). IEEE, July 2017. 23

[Cirean et al., 2012] CIREŞAN, D.; MEIER, U.; MASCI, J. ; SCHMIDHUBER, J.. **Multi-column deep neural network for traffic sign classification**. Neural Networks, 32:333–338, Aug. 2012. 2.1

[Girshick, 2015] GIRSHICK, R.. **Fast r-CNN**. In: 2015 IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV). IEEE, Dec. 2015. 2.2

[Girshick et al., 2016] GIRSHICK, R.; DONAHUE, J.; DARRELL, T. ; MALIK, J.. **Region-based convolutional networks for accurate object detection and segmentation**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(1):142–158, Jan. 2016. 2.2

[Grigorescu et al., 2020] GRIGORESCU, S.; TRASNEA, B.; COCIAS, T. ; MACESANU, G.. **A survey of deep learning techniques for autonomous driving**. Journal of Field Robotics, 37(3):362–386, Apr. 2020. 2.1

[Han et al., 2019] HAN, B.-G.; LEE, J.-G.; LIM, K.-T. ; CHOI, D.-H.. **Design of a scalable and fast YOLO for edge-computing devices**. Sensors, 20(23):6779, Nov. 2020. 1.2

[Hang Su et al., 2018] SU, H.; ZHU, X. ; GONG, S.. **Open logo detection challenge**. CoRR, abs/1807.01964, 2018. 1.5, 4.1

[ILSVRC, 2012] **Large scale visual recognition challenge 2012 (ilsvrc2012)**. http://www.image-net.org/challenges/LSVRC/2012/results.html (Accessed: 2010-09-30). 2.1

[Ivakhnenko, 1971] IVAKHNENKO, A. G.. **Polynomial theory of complex systems**. IEEE Transactions on Systems, Man, and Cybernetics, SMC-1(4):364–378, Oct. 1971. 2.1

[Junyi Chai & Anming Li, 2019] CHAI, J.; LI, A.. **Deep learning in natural language processing: A state-of-the-art survey**. In: 2019 INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND CYBERNETICS (ICMLC). IEEE, July 2019. 2.1

[Krizhevsky et al., 2012] KRIZHEVSKY, A.; SUTSKEVER, I. ; HINTON, G. E.. **Imagenet classification with deep convolutional neural networks**. In: PROCEEDINGS OF THE 25TH INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS - VOLUME 1, NIPS'12, p. 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc. 2.1

[Lecun et al., 1998] LECUN, Y.; BOTTOU, L.; BENGIO, Y. ; HAFFNER, P.. **Gradient-based learning applied to document recognition**. Proceedings of the IEEE, 86(11):2278–2324, 1998. 2.1

[Lin et al., 2020] LIN, T.-Y.; GOYAL, P.; GIRSHICK, R.; HE, K. ; DOLLAR, P.. **Focal loss for dense object detection**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(2):318–327, Feb. 2020. 2.2

[Liu et al., 2016] LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y. ; BERG, A. C.. **Ssd: Single shot multibox detector**. Lecture Notes in Computer Science, p. 21–37, 2016. 1.5

[Liu et al., 2016] LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y. ; BERG, A. C.. **SSD: Single shot MultiBox detector**. In: COMPUTER VISION – ECCV 2016, p. 21–37. Springer International Publishing, 2016. 2.2, 3.2

[Nassif et al., 2019] NASSIF, A. B.; SHAHIN, I.; ATTILI, I.; AZZEH, M. ; SHAALAN, K.. **Speech recognition using deep neural networks: A systematic review**. IEEE Access, 7:19143–19165, 2019. 1.2, 2.1

[Ning et al., 2017] NING, C.; ZHOU, H.; SONG, Y. ; TANG, J.. **Inception single shot MultiBox detector for object detection**. In: 2017 IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA & EXPO WORKSHOPS (ICMEW). IEEE, July 2017. 3.4

[Redmon & Farhadi, 2017] REDMON, J.; FARHADI, A.. **YOLO9000: Better, faster, stronger**. In: 2017 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR). IEEE, July 2017. 2.2

[Redmon & Farhadi, 2018] REDMON, J.; FARHADI, A.. **Yolov3: An incremental improvement**, 2018. 2.2, 3.2

[Redmon et al., 2016] REDMON, J.; DIVVALA, S.; GIRSHICK, R. ; FARHADI, A.. **You only look once: Unified, real-time object detection**. In: 2016 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR). IEEE, June 2016. 1.5, 2.2

[Ren et al., 2017] REN, S.; HE, K.; GIRSHICK, R. ; SUN, J.. **Faster r-CNN: Towards real-time object detection with region proposal networks**. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6):1137–1149, June 2017. 2.2

[Rezatofighi et al., 2019] REZATOFIGHI, H.; TSOI, N.; GWAK, J.; SADEGHIAN, A.; REID, I. ; SAVARESE, S.. **Generalized intersection over union: A metric and a loss for bounding box regression**, 2019. 5.1

[Shashi Pal et al., 2017] SINGH, S. P.; KUMAR, A.; DARBARI, H.; SINGH, L.; RASTOGI, A. ; JAIN, S.. **Machine translation using deep learning: An overview**. In: 2017 INTERNATIONAL CONFERENCE ON COMPUTER, COMMUNICATIONS AND ELECTRONICS (COMPTELIX). IEEE, July 2017. 1.2, 2.1

[Solovyev et al., 2019] SOLOVYEV, R.; WANG, W. ; GABRUSEVA, T.. **Weighted boxes fusion: ensembling boxes for object detection models**. arXiv preprint arXiv:1910.13302, 2019. 3.5

[Tan et al., 2020] TAN, M.; PANG, R. ; LE, Q. V.. **EfficientDet: Scalable and efficient object detection**. In: 2020 IEEE/CVF CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR). IEEE, June 2020. 1.2, 2.2

[WEBSITE AWS, 2021] **Latência de vídeo em streaming ao vivo**. https://aws.amazon.com/pt/media/tech/video-latency-in-live-streaming/ (Accessed: 2021-04-04). 2.3

[Wajih et al., 2020] WAJIH; ROSEBROCK, A.; ANNE; AHMED, W.; KIRAN; JSKY; ABBY; AAMER; ELSA; MIEJ ; ET AL.. **Intersection over union (iou) for object detection**, Apr 2020. 2.4.1