
UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Evair de Jesus Silva Cunha

# Melhorando a detecção de objetos pequenos com DETRAug

Recife

2023

Evair de Jesus Silva Cunha

# Melhorando a detecção de objetos pequenos com DETRAug

Dissertação apresentada ao Programa de Pós-Graduação Acadêmica em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

**Área de Concentração**: Inteligência Computacional

**Orientador**: Prof. Dr. Cleber Zanchettin

**Coorientador**: Prof. Dr. David Lopes de Macêdo

Recife

2023

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

C972m Cunha, Evair de Jesus Silva
Melhorando a detecção de objetos pequenos com DETRAug / Evair de Jesus Silva Cunha – 2023.
63 f.: il., fig., tab.

Orientador: Cleber Zanchettin.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2023.
Inclui referências.

1. Inteligência computacional. 2. Detecção de objetos. 3. DETR. 4. AUGMIX. 5. Data augmentation. I. Zanchettin, Cleber (orientador). II. Título

006.31 CDD (23. ed.) UFPE - CCEN 2024 – 007

**Evair de Jesus Silva Cunha**

**“Melhorando a detecção de objetos pequenos com DETRAug”**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 16 de agosto de 2023.

**BANCA EXAMINADORA**

_____

Prof. Dr. Cleber Zanchettin
Centro de Informática / UFPE
(**Orientador**)

_____

Prof. Dr. João Paulo Silva do Monte Lima
Departamento de Computação / UFRPE

_____

Prof. Dr. Tsang Ing Ren
Centro de Informática / UFPE

Dedico este trabalho aos meus pais.

À ciência.

## AGRADECIMENTOS

Agradeço aos meus pais por não terem desistido de me por na linha dos estudos me deram todo suporte, na medida do que foi possível, e serem minha maior inspiração de vida .

Agradeço a minha esposa, Morgana Silva, por ser meu bastião durante essa empreitada e por ser minha melhor amiga e ter me auxiliado quando necessitei e por acreditar em mim.

Agradeço em especial ao meu Orientador, Cleber Zanchettin e Coorientador David Macêdo por possibilitarem que eu expandisse meus horizontes, por terem me encorajado a superar os desafios que essa pesquisa propôs. Além disso, pela paciência, sabedoria e incentivo, sou muito grato por terem me acolhido como aluno e disporem do seu tempo para me ajudar a tornar esse trabalho possível.

Expresso toda minha gratidão ao corpo docente e discente do Centro de Informática da Universidade de Pernambuco, a quem fico lisonjeado por dele ter feito parte.

Também agradeço meu grande amigo Hygor Jardim desde a graduação por ter me auxiliado em vários momentos que com sua sabedoria sempre me deu as melhores dicas e os melhores conselhos.

Agradeço ao meu parceiro e dupla de dois que conheci no mestrado, Levi Ramos, por ter me sido um companheiro de guerra no decorrer do mestrado. Por ter me ensinado a ser mais avexado com as tarefas e obrigações.

Agradeço aos meus professores de graduação que permitiram eu chegar aqui e em especial aos professores José Jailton e Tássio Carvalho por terem dedicado um pouco do seu tempo para me auxiliarem nas ideias para o mestrado.

Agradeço a CAPES por ter me dado assistência durante o mestrado e permitido que os trabalhos fossem um pouco menos difíceis.

Agradeço a todos que direta e indiretamente me permitiram chegar tão longe, parafraseando Isaac Newton, se eu vi mais longe, foi por estar sobre ombros de gigantes.

"A imaginação muitas vezes nos leva a mundos que nunca sequer existiram. Mas sem ela não vamos a lugar algum." (SAGAN, 1980).

## RESUMO

Neste trabalho é proposta uma nova abordagem para detecção de objetos chamada DETRAug. A nova técnica combina estratégias dos modelos Deformable DETR e AUGMIX. O AUGMIX é uma técnica para melhorar a robustez no treinamento de modelos de aprendizagem de máquina com base em mudanças na distribuição dos dados de treinamento e teste. O Deformable DETR é uma variante do modelo DETR, a qual obtém melhores resultados com detecção de objetos pequenos, além de apresentar um tempo de treinamento mais rápido que sua forma original. No DETRAug também focamos na ampliação da quantidade de imagens de treinamento e consequente adaptação das estratégias para detecção de objetos. Desta forma, a abordagem proposta busca produzir, de maneira estocástica, pequenas imagens com diferentes transformações, que no decorrer do processo de treinamento são encadeadas com o intuito de gerar uma imagem única a ser adicionada no conjunto de treinamento do modelo. A divergência de Jensen-Shannon, uma métrica bastante útil para modelagens com distribuições, foi utilizada para avaliar a função de *Loss* do modelo. No decorrer dos experimentos com o uso do DETRAug, foi possível verificar que o modelo proposto apresentou uma diminuição na quantidade de detecções "no-object", ou seja, nas detecções errôneas que o modelo produz. Durante os experimentos, esta nova versão foi comparada com os modelos DETR e EfficientDet. A abordagem proposta foi avaliada em experimentos com seis datasets públicos. Ao fim dos experimentos, foi possível auferir uma melhoria de no mínimo 0.9% em relação a métrica mAP, também foi observada uma *Loss* de treinamento mais estável no modelo proposto. Além disso, é possível identificar visualmente uma melhora na detecção de objetos que, sob a mesma condição, são ignorados pelos outros modelos da literatura.



**Palavras-chave**: Detecção de Objetos; DETR; AUGMIX; Data Augmentation.

## ABSTRACT

In this study, we propose a new approach for small object detection, named DETRAug. This new technique combines strategies from the Deformable DETR and AUGMIX models. AUGMIX is a technique that enhances the robustness of machine learning model training, based on changes in the distribution of training and test data. On the other hand, Deformable DETR is a variant of the DETR model that achieves better results with small objects and has a faster training time compared to its original version. With DETRAug, we also focus on expanding the quantity of training images and adapting strategies for small object detection. Thus, the proposed approach aims to stochastically generate small images with different transformations, which are chained throughout the training process with the goal of creating a unique image to be added to the model's training set. We used the Jensen-Shannon divergence, a useful metric for modeling with distributions, to assess the model's Loss function. Throughout the experiments using DETRAug, it was observed that the proposed model showed a reduction in the quantity of "no-object" detections, i.e., erroneous detections produced by the model. During the experiments, this new version was compared with the DETR and EfficientDet models. The proposed approach was evaluated in experiments with six public datasets. At the end of the experiments, we observed an improvement of at least 0.9% in the mAP metric and a more stable training Loss in the proposed model. Additionally, it was possible to visually identify an improvement in the detection of objects that, under the same conditions, are ignored by other models in the literature.



**Keywords**: Small Object Detection; DETR; AUGMIX; Data Augmentation.

## LISTA DE FIGURAS

# LISTA DE CÓDIGOS

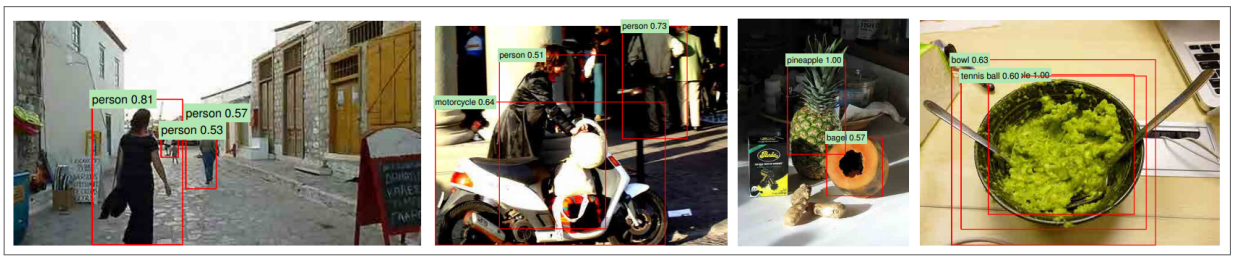
## LISTA DE TABELAS

## SUMÁRIO

# 1 INTRODUÇÃO

A Detecção de objetos é uma técnica da área de visão computacional que combina duas tarefas, classificação de imagem e localização de objetos. Portanto, tem como objetivo a localização de um ou mais objetos, através da marcação de caixas retangulares, com a respectiva classificação dessas marcações em torno dos objetos localizados na imagem ou vídeo (Figura 1). Os algoritmos de detecção de objetos normalmente aproveitam o aprendizado de máquina ou o aprendizado profundo para produzir resultados significativos. Atualmente, a detecção de objetos tem tido uma larga e frequente aplicação em projetos do mundo real, como identificação de placas de trânsito (SIOGKAS; DERMATAS, 2006), vigilância de áreas públicas para detecção de armas (BUCKCHASH; RAMAN, 2017), monitoramento de queimadas em florestas por satélite e imagem aérea (JIN; LU, 2019; YANG et al., 2019), sendo também uma tecnologia chave em carros com sistemas de assistência à condução (WEI et al., 2019).

Figura 1 – Detecção de objetos produzidos pelo R-CNN



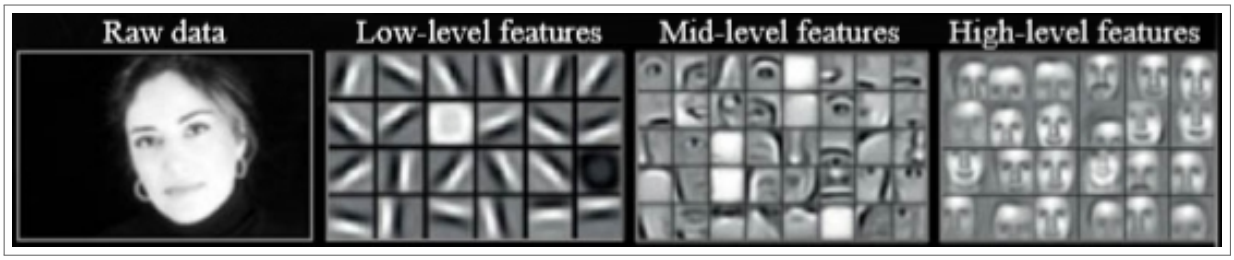

**Fonte:** (GIRSHICK et al., 2014)

No momento atual, existem dois tipos principais de frameworks quando se trata de características extraídas por CNN (Convolutional Neural Network), detectores one-stage e two-stage. Estes geralmente usam *backbones* ResNet (HE et al., 2016) ou *VGG* (**??**) para extrair uma grande quantidade de características da imagem de entrada, o que em outras palavras significa que a parte mais relevante da informação original é resumida em um novo conjunto de informações.

As técnicas de extração de características são tradicionalmente baseadas em algoritmos como Haar (Haar-like features), SIFT (scale-invariant feature transform), HOG (histogram of oriented gradient) e Principle Components Analysis (PCA) que são alguns dos mais conhecidos. As características extraídas, por esses algoritmos, buscam trazer uma informação de mais baixo nível do objeto observado, por exemplo, a silhueta e a textura, contudo, sofrem com o limitado desempenho de generalização para cenários complexos com múltiplos objetos. Modelos

baseados em CNN, para além de extrair as informações de baixo nível, também conseguem obter informações semânticas de alto nível (CAO; CHEN; GAO, 2020).

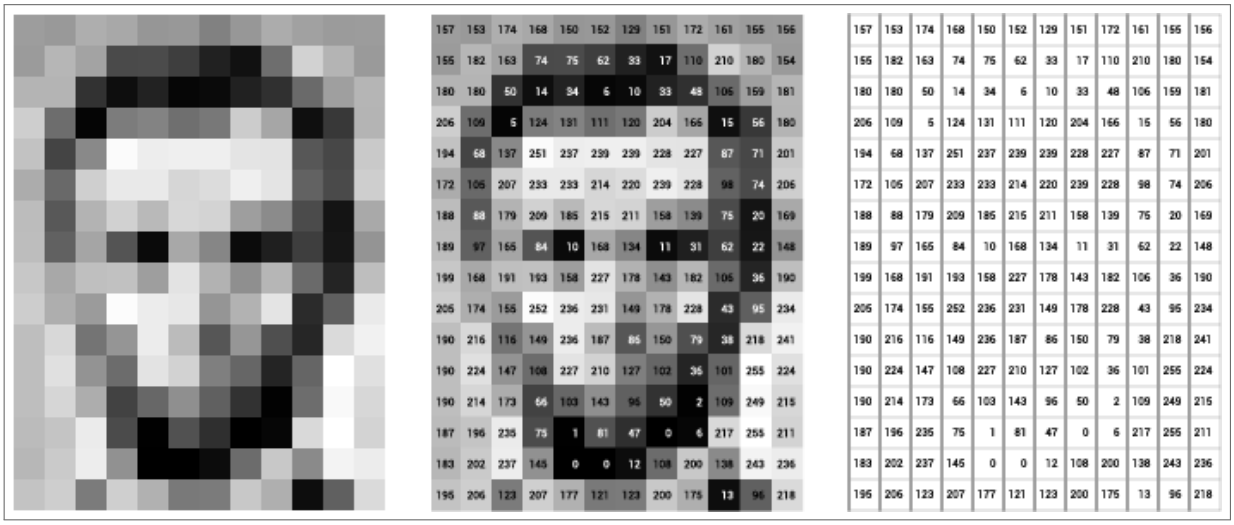
Figura 2 – Disposição das características extraídas a cada nível de uma rede neural profunda




**Fonte:** (LEE et al., 2009)

A Figura 2 ilustra, conceitualmente, o produto do que é extraído a cada camada nos modelos de redes neurais profundas modernos. À princípio, podemos não reconhecer o que está na imagem, mas quando levamos para o nível de visão computacional, temos apenas matrizes e tensores preenchidos com números, perfeitamente compreensíveis para o computador que, através de técnicas computacionais, consegue generalizar tais informações (Figura 3).

Figura 3 – Exemplo simplificado do processo de compreensão



**Fonte:** (LEVIN, 2017)

Em CNNs, ao passar trechos de uma imagem pelos filtros, apenas as características mais importantes são extraídas. Dado que um pixel sempre depende dos pixels que estão no seu entorno, isso ajuda no aprendizado, com o foco apenas no que é mais relevante na imagem. Os detalhes individuais de cada pixel não são considerados e, portanto, não são aprendidos. Trazendo para o espectro dos *transformers*, contudo, ao se inserir uma imagem neste tipo de modelo, ao invés de apenas se obter as partes que os filtros extraem, o modelo Transformer

considera todo o contexto da imagem e não apenas partes, o que de certa forma aumenta as chances de o modelo ter um desempenho melhor. Isso dá origem a modelos que combinam CNNs e transformadores codificadores-decodificadores (VASWANI et al., 2017). Incorporadas com essas ideias podemos citar a arquitetura SMILES (KARPOV; GODIN; TETKO, 2020), Detection Transformer (DETR) (CARION et al., 2020), Deformable DETR (ZHU et al., 2020) e Dynamic DETR (DAI et al., 2021) que exploram essa versatilidade e buscam integrar a melhor característica de cada arquitetura.

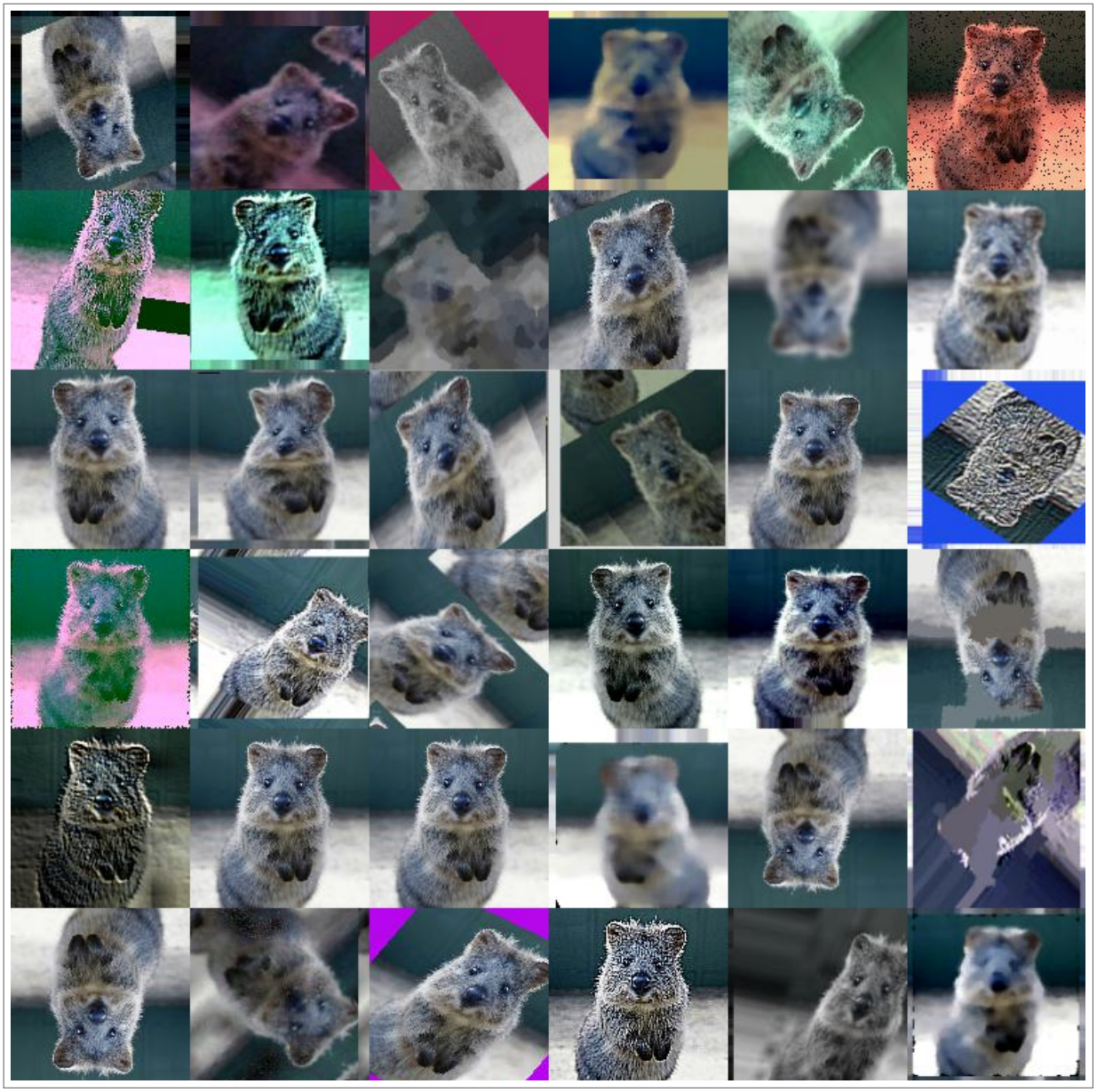
O DETR é um caso de sucesso, realizando a combinação das redes convolucionais juntamente com os Transformers, mesmo considerando seu design simples, tem desempenho comparável ao Faster-CNN (CARION et al., 2020). No entanto, ele oferece um baixo desempenho para detectar objetos pequenos, além disso, por considerar toda a imagem ele sofre de convergência lenta e um custo computacional relativamente alto. O Deformable DETR vem para mitigar esses problemas, reduzindo o tempo de treinamento para convergência em torno de 10x e pode alcançar melhor desempenho que o DETR, principalmente para objetos pequenos, tudo isso sem aumentar significativamente o custo computacional (KHAN et al., 2022).

A introdução das CNNs foi um grande marco na evolução de detecção de objetos, tanto que a maioria dos modelos atuais o utilizam de alguma maneira. Há pouco tempo, as tarefas de classificação e localização de objetos eram considerados problemas de extrema complexidade, hoje, porém, é uma tarefa possível. Entretanto, ainda existem alguns desafios a serem solucionados em detecção de objetos, como: variação do ponto de vista, oclusão, localização de objetos devido a dualidade de classificação e determinação de posição, detecção em tempo real, iluminação, desequilíbrio de classe e a limitação dados.

Tratando especificamente sobre dados, existe um problema que afeta no desempenho dos classificadores: a usual incompatibilidades entre a distribuição os dados de treinamento e teste. Embora esse problema seja recorrente, pesquisas sobre esse tema não são comuns. É notório, atualmente, que as tecnologias de visão computacional, inclusive outras como processamento de linguagem natural ou reconhecimento de fala, necessitam de uma grande quantidade de dados para ter um bom desempenho. Além disso, é de suma importância que os dados de treinamento e de teste tenham a mesma distribuição. É difícil, todavia, que se tenha à disposição grande quantidade dos dados do problema que se quer tratar, eles normalmente são escassos ou não existem (principalmente se considerarmos dados rotulados). Podem existir dados alternativos ou similares ao do problema tratado e o conjunto deles pode auxiliar no desenvolvimento do modelo que se pretende treinar. Entretanto, o uso deles pode enviesar

o modelo treinado a ter um bom desempenho apenas para os dados alternativos, tendo um efeito contrário do que se deseja (ASSAWIEL, 2018).

Figura 4 – Ilustração de técnicas de augmentation aplicadas a mesma imagem.



**Fonte:** (JUNG et al., 2020)

Em machine learning, os sistemas estão fortemente atrelados ao que foi aprendido no treinamento. Sempre é esperado que a distribuição desses dados represente de forma fiel a realidade. Caso essa premissa seja quebrada, os resultados de saída ou inferência do modelo treinado podem não ter um bom nível de confiabilidade. Portanto, devido à suscetibilidade de variância nessa distribuição, mesmo que pequena, isso pode causar uma perda de desempenho dos classificadores, não tendo um bom funcionamento para determinado problema

(HENDRYCKS; DIETTERICH, 2019).

Para solucionar essa adversidade, uma das várias soluções é a utilização de dados gerados artificialmente, conhecido como *data augmentation*, um dos métodos mais utilizados em *deep learning*, sobretudo na área de reconhecimento de imagens. Essa solução surge quando é inviável a captura de milhões de imagens para determinado problema ou mesmo quando estes existem, mas se quer introduzir diversidade no conjunto de treinamento. Então se usam tratamentos diversos sobre as imagens que estão à disposição. Esse tratamento aumenta a diversidade do conjunto de treinamento aplicando diversas transformações na imagem, como orientação da imagem, ampliação, rotação, recorte, *flipping* vertical e horizontal, e por consequência isso resultará em um aumento na quantidade de dados (Figura 4).

## 1.1 OBJETIVO

Esta dissertação investiga como estratégias de *data augmentation* possam ajudar na evolução de técnicas de detecção de objetos. São investigadas as técnicas DETR e AUGMIX, que tratam, respectivamente, sobre a tarefa de detecção de objetos e classificação. Nosso objetivo é avaliar como o arranjo dessas técnicas pode aprimorar o desempenho do framework DETR e sua variante DETR Deformable. As melhorias de desempenho são baseadas na redução da *loss* que os modelos geram e no aumento na detecção de objetos.

### 1.1.1 Objetivos específicos

Com base no que foi exposto, alguns objetivos específicos foram delineados para cumprir o objetivo principal, quais sejam:

- Implementação e avaliação do estado da arte das técnicas discutidas.
- Estudo da área de detecção de objetos, identificando as principais abordagens utilizadas e oportunidades para desenvolvimento de melhorias do método.
- Propor uma variante dos modelos Deformable DETR que também se adapte a outros modelos, sejam CNNs ou Transformers.
- Avaliar o comportamento do método proposto em relação aos modelos base, considerando as métricas consolidadas na literatura para esse tipo de tarefa.

## 1.2 ESTRUTURA DA DISSERTAÇÃO

O restante deste trabalho está descrito da seguinte forma:

- **Capítulo 2 - Fundamentação:** Neste capítulo é apresentado a fundamentação teórica de conceitos essenciais para a elaboração e entendimento deste trabalho.
- **Capítulo 3 - Metodologia:** Este capítulo tem o intuito de apresentar a proposta de melhoria nos modelos Deformable DETR que também se adapte a outros modelos, sejam CNNs ou Transformers, utilizando-se de técnicas de aumento de dados.
- **Capítulo 4 - Resultados:** Neste capítulo serão apresentados os resultados obtidos nas etapas descritas na metodologia, bem como a análise dos resultados de forma a validar a metodologia proposta para melhoria de detecção de objetos.
- **Capítulo 5 - Conclusão:** São apresentadas as conclusões para a presente pesquisa e propostas de futuros trabalhos.

# 2 FUNDAMENTAÇÃO

Neste capítulo são apresentados alguns conceitos essenciais que servirão para um melhor entendimento do presente trabalho.

## 2.1 DETECÇÃO DE OBJETOS

A detecção de objetos é um tópico amplamente estudado e um segmento importante no campo da visão computacional e robótica. A Detecção de objetos visa identificar em uma imagem todas as categorias de objetos de uma ou várias classes conhecidas, previamente, como carros, pessoas e animais, por exemplo. Simplificando a tarefa seria responder às perguntas do tipo: Quais objetos estão na imagem e onde estão? Embora pareça simples, a detecção de objetos é uma tarefa que envolve um grande nível de complexidade, dada a sua dependência de uma variedade de fatores como a variação de escala de um objeto, as posições em que o objeto aparece, a orientação, o formato e o contexto no entorno do objeto. Além disso, a imagem pode apresentar problemas com excesso ou escassez de iluminação, ruídos, distorção na imagem e obstruções, que são alguns dos principais fatores que tornam essa tarefa ainda mais desafiadora.

A AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017), uma rede neural convolucional, foi apresentada ao mundo em 2012, quando venceu a competição anual da ImageNet Large Scale Visual Recognition Challenge (ILSVRC) daquele mesmo ano. Esse trabalho foi um marco para trabalhos envolvendo Deep Learning (DL, em português "aprendizagem profunda"), considerando que até aquela época as técnicas tradicionais de redes neurais e redes convolucionais tinham bom desempenho em tarefas específicas como identificação de números escritos à mão. No entanto, havia a escassez de recursos computacionais, o que significava um entrave no progresso dessa área. Foi neste momento que surgiram as otimizações nas GPUs (Graphics Processing Unit, ou Unidade de Processamento Gráfico) que permitiram o uso em larga escala no treinamento de redes neurais. A partir dessas microevoluções, que nos anos subsequentes houve uma enorme profusão de pesquisas e aplicações com os conceitos de redes neurais convolucionais profundas.

A área de detecção de objetos em especial se beneficiou muito desse progresso, pois já em 2014 houve o lançamento do modelo conhecido como R-CNN (Regions with Convolutional

Neural Network features, ou Regiões com características de Rede Neural Convolucional), que utiliza CNNs e abre o ramo dos algoritmos de detecção em dois estágios. Enquanto isso, o modelo OverFeat, proposto em 2013 por (SERMANET et al., 2013), é um dos pioneneiros quando se trata de algoritmos de detecção em estágio único.

Houve essa ramificação, a partir da AlexNet, das técnicas de detecção de objetos que formaram caminho entre os algoritmos de dois estágios e os que se apresentam em um único estágio. Abaixo estão resumidos alguns dos principais modelos, inicialmente os modelos de um estágio, seguidos pelos modelos de dois estágios.

### 2.1.1 Arquiteturas de detectores de um estágio

#### *2.1.1.1 OverFeat*

O modelo proposto por (SERMANET et al., 2013) traz a ideia de unificar em um framework as tarefas, quais sejam, por ordem de aumento de complexidade: classificação, localização e detecção. Para facilitar essa comunicação entre diferentes tarefas, foi utilizado uma base de extração de características comum.

O OverFeat — nome dado ao extrator de características — tem como objetivo principal a geração de múltiplos *bounding boxes*, que são gerados pelo funcionamento simultâneo do classificador e uma rede de regressão. A rede de regressão é um classificador treinado que teve as últimas camadas de classificação substituídas por uma rede de regressão. Com os *bounding boxes* gerados e tendo os *scores* de cada um, é possível fazer a fusão entre todos, que nesse trabalho, foi utilizada uma estratégia gananciosa, muito pouco utilizada atualmente. Além de utilizar uma arquitetura similar à da AlexNet, esse modelo utiliza a técnica de janela variável juntamente com uma variação de escalas da imagem de entrada. Isso permitiu que a classificação tivesse uma melhor desempenho utilizando a média das escalas, ao invés de uma única. Como é comum dos detectores de estágio único, o OverFeat tem melhor desempenho em termos de velocidade, no entanto, possui baixo desempenho quando se trata de acurácia.

#### *2.1.1.2 YOLO*

A arquitetura YOLO, acrônimo para *You Only Look Once*, é uma das mais conhecidas no meio dos algoritmos de detecção de objetos. Seguindo a tendência de unificação de processos,

Figura 5 – Parte do *pipeline* de detecção, a imagem demostra a geração dos múltiplos bounding boxes com a posterior mescla tendo como produto o objeto detectado e o nível de confiança

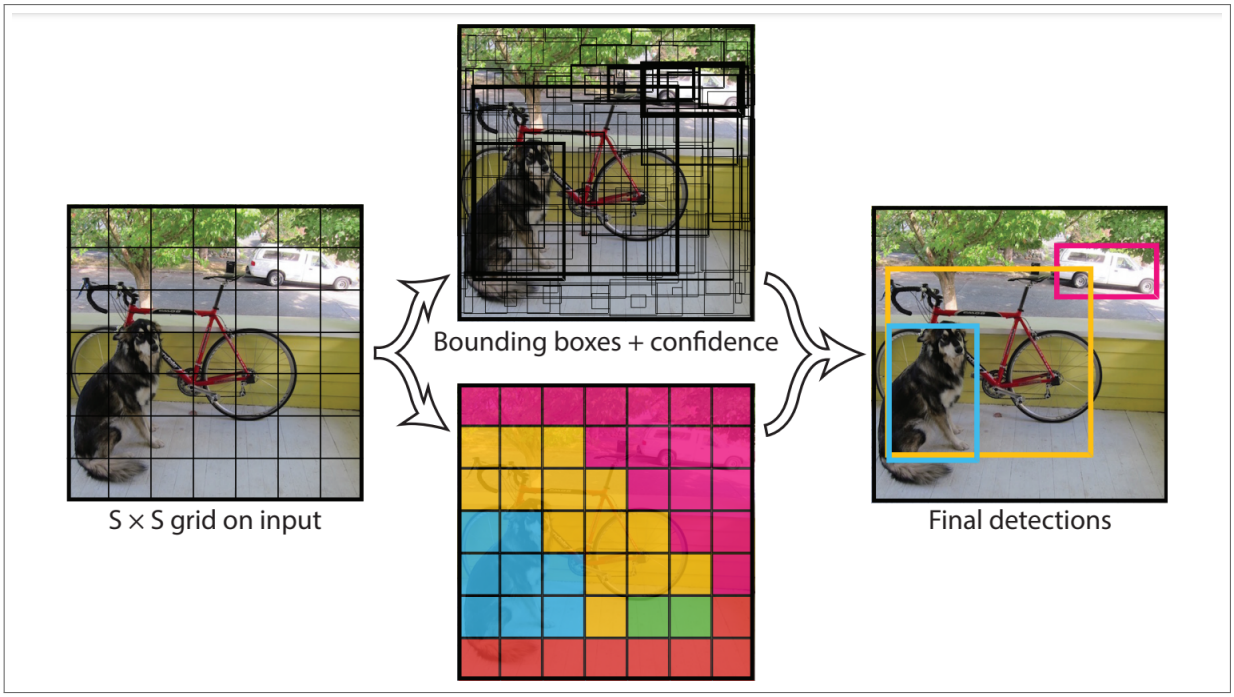
**Fonte:** Adaptado de (SERMANET et al., 2013)

vista também no OverFeat, a YOLO concentra os processos de classificação e predição de *bounding boxes* e faz as estimativas "olhando" apenas uma vez sobre toda a imagem. As estimativas são feitas em uma única rede convolucional, o que a torna mais rápida do que as arquiteturas que utilizam redes separadas para cada processo. Além disso, a característica de varredura sobre toda a imagem da YOLO a coloca um passo à frente em relação à arquitetura R-CNN que se concentra em potenciais lugares que possam conter objetos, ou seja, apenas partes da imagem.

Um dos procedimentos iniciais dentro da YOLO traz o conceito de dividir a imagem inteira em células idênticas formando uma grade de tamanho $S \times S$. Com a união delas, cada célula é responsável pela detecção de múltiplos *bounding boxes* e o nível de confiança para cada *box* gerado. Embora a célula possa fazer mais de um *bounding box* por célula, a arquitetura não é capaz de detectar mais de um objeto, se houverem vários objetos na mesma célula. Além disso, cada célula também gera a classificação do objeto. A Figura 6 consegue sintetizar um

pouco de como o modelo se comporta.

Figura 6 – Na imagem é demonstrado como é a segmentação da YOLO utilizando células, além de mostrar o mapa de probabilidade de classe, os bounding boxes das inúmeras células com a subsequente detecção final




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

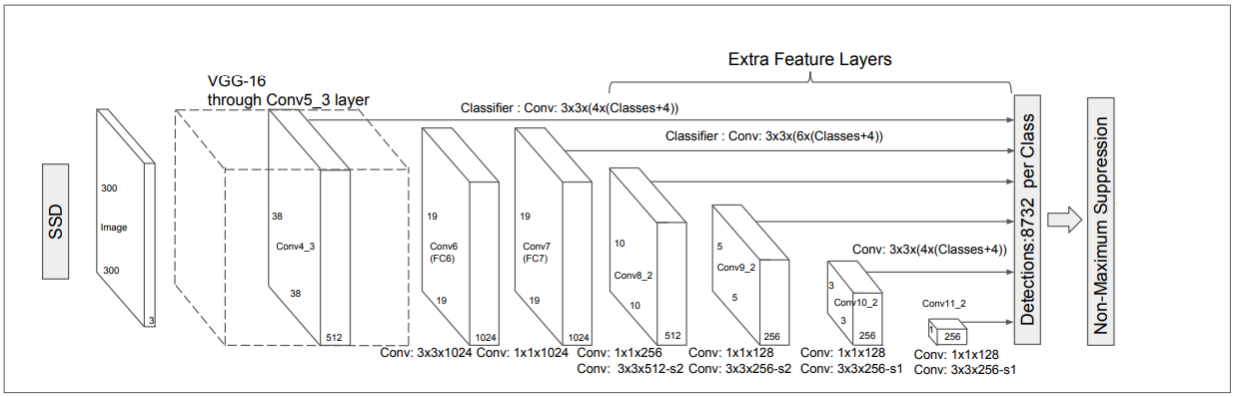
Para avaliar o nível de confiança do modelo é utilizado o *Intersection over Union* (IoU). Essa métrica mede a intersecção entre o *ground-truth*, que é a marcação manual prévia que se deseja que o modelo faça igual ou próximo. Para cada *bounding box* feito em cada célula é avaliado o essa intersecção e tem-se como saída um valor entre 0 e 1 que diz o quão semelhante está o *bounding box* do *ground-truth*. O que se almeja é que o *bounding box* seja igual ao *ground-truth*, portanto, se houver sobreposição entre a área dos dois. Ou ainda a intregalização da área de um sobre o outro. Desta forma, maior é o nível de confiança, o que em termos percentuais deixa o valor mais próximo de 1, do contrário, quanto menor a sobreposição mais próximo de será 0. Caso não exista objetos na célula o nível de confiança é 0.

#### *2.1.1.3 SSD*

Assim como a YOLO, a arquitetura da *Single Shot Detector* (SSD) foi desenvolvida com foco em detecção de objetos em tempo real. Proposta por (LIU et al., 2015), essa arquitetura utiliza como *backbone* a VGG-16. Além disso, foi reutilizada com modificações na parte final, especificamente, as camadas totalmente conectadas fc6 e fc7 foram convertidas em camadas

convolucionais, enquanto fc8 e as camadas de dropout foram todas removidas. Ademais, foram incluídas mais quatro camadas convolucionais na parte final após fc7. Esse empilhamento de camadas convolucionais e a progressiva diminuição de tamanho das camadas é o que define a ideia da arquitetura, qual seja realizar extração hierárquica de características.

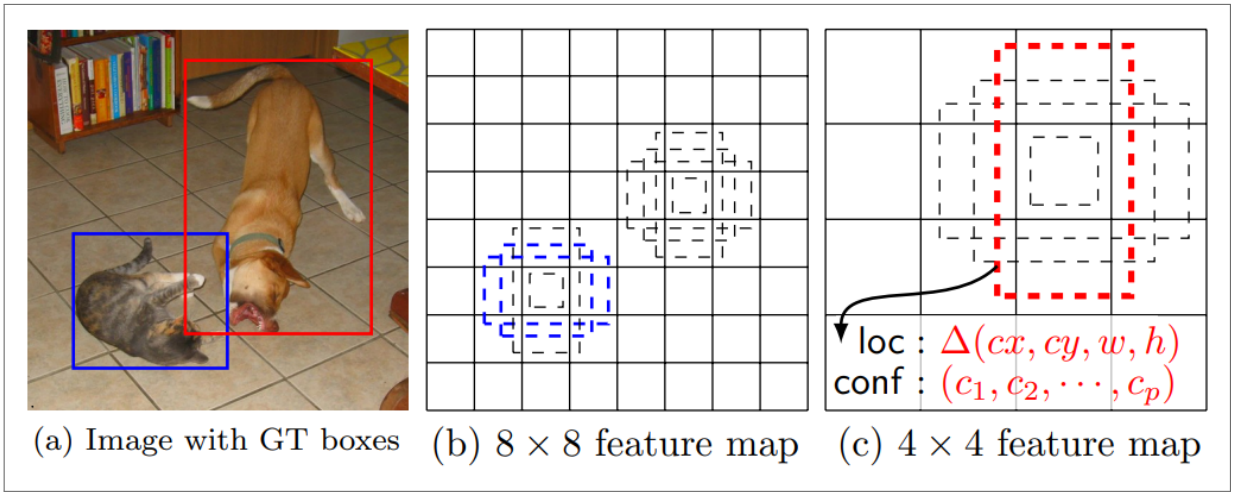
Figura 7 – Arquitetura SSD e suas camadas mostrada em detalhes




**Fonte:** Adaptado de (LIU et al., 2015)

A parte inicial da arquitetura atua na classificação, enquanto as camadas convolucionais, conforme diminuem, permitem uma melhor predição em várias escalas. Como a SSD utiliza o mapa de características segmentado, de forma semelhante à YOLO, ela também utiliza âncoras, equivalentes àquelas utilizadas no Faster R-CNN, que servem para orientar melhor o modelo a detectar uma variedade maior de objetos, assim como facilitar o treinamento. No artigo são utilizadas quatro formas de âncoras, as âncoras são calculadas em cada célula do mapa de características que podem ter o tamanho de $8 \times 8$ e $4 \times 4$. Esse tamanho do mapa influencia na detecção de objetos pequenos ou grandes. Quanto maior for o mapa, $8 \times 8$ por exemplo, ao invés de $4 \times 4$, melhor ele consegue detectar objetos pequenos. A Figura 8 apresenta um exemplo, enquanto o cachorro ocupa uma larga faixa da imagem e só é detectado no mapa de características de $4 \times 4$, o gato por ser menor é detectado por duas âncoras no mapa de $8 \times 8$. Ao final das predições é utilizado o *Non-maximum suppression* para remover as múltiplas predições de um mesmo objeto, utilizando como critério os maiores valores de nível de confiança e os *bounding boxes* que tenham IoU maior que 0.45 para uma mesma classe, mantendo as 200 melhores predições por imagem.

Figura 8 – Utilização de âncoras em diferentes tamanho de mapa de características.



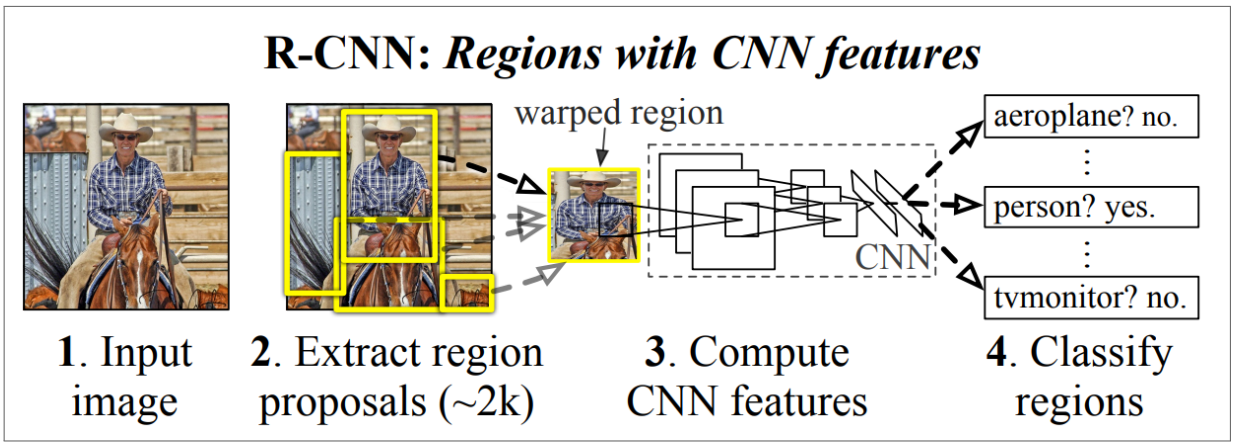

**Fonte:** (LIU et al., 2015)

### 2.1.2 Arquiteturas de detectores de dois estágios

#### *2.1.2.1 R-CNN*

Embalado com o sucesso do modelo da AlexNet, (GIRSHICK et al., 2014) propôs a utilização desse modelo em seu trabalho como um *backbone* para extração de características, fugindo um pouco de trabalhos que naquela época buscavam utilizar métodos tradicionais como histograma de gradientes orientados (do inglês, *histogram of oriented gradients*), padrão binário local (do inglês, *local binary pattern*) e transformação de característica invariável a escala (do inglês, *Scale-invariant feature transform*).

Figura 9 – Pipeline de execução da R-CNN




**Fonte:** Adaptado de (GIRSHICK et al., 2014)

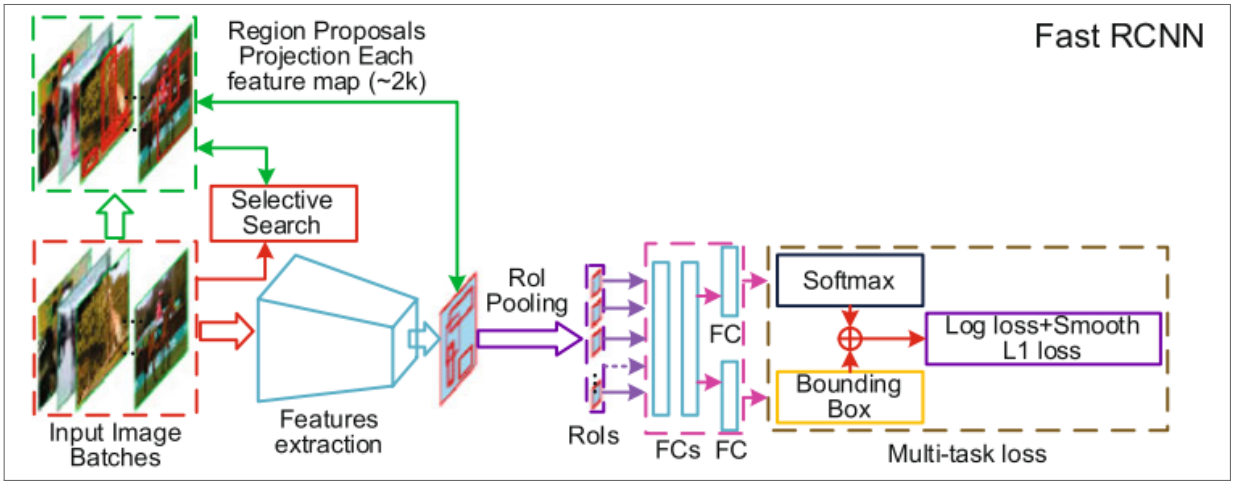
O modelo *Regions with CNN features* (R-CNN) implementou um *pipeline* para realizar

a detecção passando por três estágios. Como apresentado na Figura 9, a geração de *region proposals* que, em outros termos, são os *bounding boxes*, é feita utilizando a técnica conhecida como *Selective Search*, técnica que utiliza uma hierarquização de grupos de regiões similares, com base na textura, cor e tamanho, formando assim um mapa de segmentação. O segundo estágio realiza a computação de cada uma das 2000 regiões candidatas através de uma CNN, obtendo-se um tensor de 4096 dimensões que será passado para o terceiro estágio onde realizará a classificação com o algoritmo conhecido como *Support Vector Machine* (SVM). Para realizar o *fine-tuning* é utilizado NMS e a regressão de *bounding box*.

#### *2.1.2.2 Fast R-CNN*

A evolução do modelo R-CNN foi o trabalho proposto por (GIRSHICK, 2015), no qual é descrito o Fast R-CNN. As melhorias foram focadas nos pontos fracos que incluem alto consumo de memória e o não compartilhamento das características que as CNNs podem realizar. Para resolver o problema de velocidade foram propostas duas alterações: a adição da camada de *Region of Interest* (RoI) e a unificação dos processos de classificação e regressão de *bounding boxes*, antes separados, com a utilização de uma *Loss* multitarefa. Essa junção no treino permitiu o compartilhamento das características.

Figura 10 – A imagem abaixo representa melhor as alterações e como é feito o pipeline desse modelo




**Fonte:** Adaptado de (XIAO et al., 2020)

Nesta versão, o algoritmo de *Selective Search* ainda é utilizado, no entanto, a seleção de regiões de interesse são usadas em cima de um mapa de características extraído com o backbone VGG-16. Após isso, a camada de RoI *pooling* produz um mapa de características

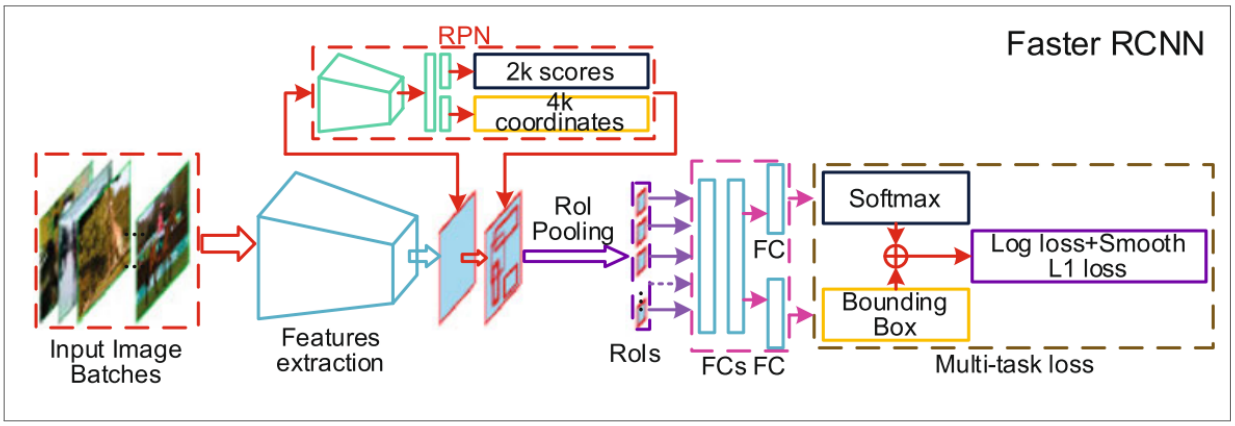
de tamanho fixo, que será injetado em camadas totalmente conectadas (FC) (do inglês, *Fully Connected Layers*). A fase final fica por conta de duas pontas: uma responsável pela predição das classes utilizando o Softmax para produção de probabilidades de $N$ classes, e a outra é responsável pela produção de quatro valores reais, que são as posições refinadas de cada um ponto dos cantos do *bounding boxes*. Essas alterações permitiram que o modelo Fast R-CNN tivesse um desempenho $9\times$ maior que o R-CNN durante o treino e $213\times$ durante o teste.

#### *2.1.2.3 Faster R-CNN*

Com o salto em relação a velocidade, na Fast R-CNN, a busca passou a ser a melhoria dos gargalos que existiam no modelo. Assim, surgiu o trabalho de (REN et al., 2015) propondo um novo modelo com melhorias nos algoritmos de proposta de regiões, o que agilizaria a detecção em tempo real. Em suma, esse modelo é o Faster R-CNN.

O *pipeline* do Faster R-CNN mantém em grande parte o que foi feito no Fast R-CNN. Nessa nova versão, é removido o *Selective Search* e incluído o algoritmo *Region Proposal Networks* (RPN) que recebe imagens de qualquer tamanho, e consegue devolver um conjunto de regiões de interesse, cada um com seu índice de pertencimento a uma classe. Outra característica é que o RPN consegue compartilhar as camadas de convolução com o *backbone*, já que o RPN é conectado à última camada do *backbone*. Isso reduz o custo de computação e diminui o tempo das regiões propostas. Além de melhorar a qualidade das regiões de interesse, é necessário uma menor número de regiões - 300 por imagem.

Figura 11 – Ilustração do funcionamento do modelo Faster R-CNN.

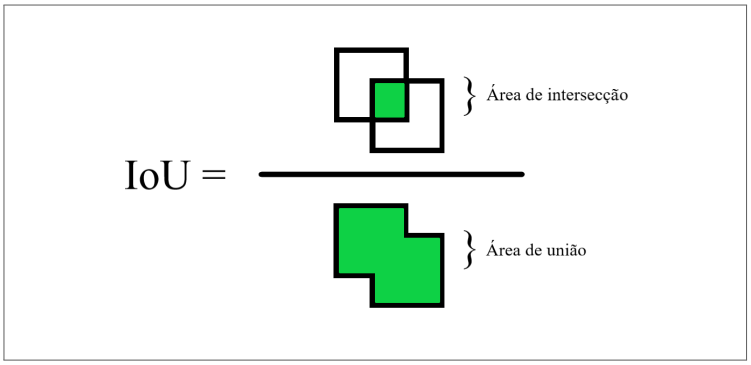



**Fonte:** Adaptado de (XIAO et al., 2020)

## 2.2 INTERSECTION OVER UNION (IOU)

Uma das principais e precursoras métricas dentro de detecção de objetos é o *Intersection over Union* (ROSEBROCK, 2016), também conhecido como índice de Jaccard, é utilizado para comparar a similaridade entre duas formas arbitrárias. Essa métrica é invariável a escala, pois para realizar as comparações entre duas formas, sejam de área ou volume, ela leva em consideração apenas a largura, altura e localização dos *bounding boxes*. Para elucidar a forma como o cálculo do IoU é feito, a Figura 12 demonstra que, tendo duas áreas é possível se obter o valor da porcentagem de similaridade entre o *bounding box* proposto e o *ground-truth*.

Figura 12 – Diagrama de cálculo do IoU




**Fonte:** Elaborado pelo Autor

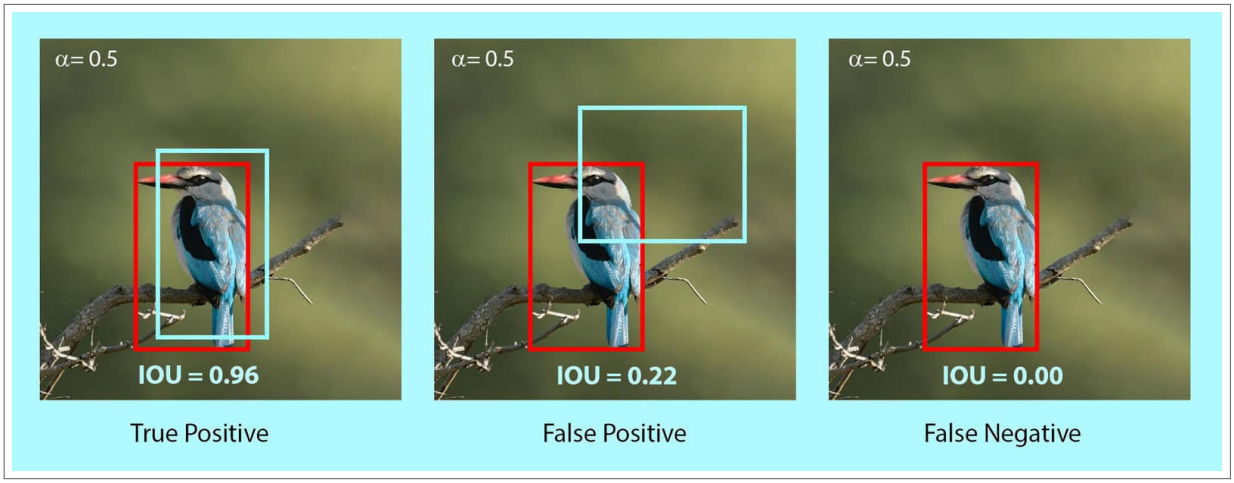
Essa técnica tem um forte uso juntamente com NMS, de maneira a filtrar todas as predições. Para isso, além do IoU, é avaliado o nível de confiança de cada predição, sendo que somente os maiores são selecionados. Para o IoU, é utilizado um threshold, geralmente de 0.5. Ao aplicar o IOU, podemos dizer se uma detecção é válida ou inválida, respectivamente, verdadeiro positivo e falso positivo.

Os valores de IoU podem variar entre excelente, bom e ruim, o que define isso é o quanto uma área está se sobrepondo a outra. É difícil que ambas áreas sobreponham-se perfeitamente alinhadas. Alguns dos casos relatados podem ser observados na Figura 13. Outro fato importante é que o IoU serviu de base para outra técnica que viria a ser proposta por (REZATOFIGHI et al., 2019) conhecida como *Generalized Intersection over Union* (GIoU).

O resultado da classificação que o IoU gera servirá de parâmetro para outras duas métricas que são a Precisão e o Recall. Sabendo disso, podemos definir as possíveis saídas da classificação que o IoU poderá gerar, como descritas na Figura 13.

- **Verdadeiro positivo (VP):** É uma detecção correta, as áreas entre o que foi predito e

Figura 13 – Cenários de diferentes porcentagens para o IoU de acordo com a sobreposição




**Fonte:** (KUKIL, 2023)

o *ground truth* está acima da margem do *threshold*, ou seja, IoU $\geq$ *threshold*.

- **Falso positivo (FP):** É uma detecção errada de fato, neste caso a razão entre o que foi predito e o *ground truth* está abaixo da margem do *threshold*, IoU $\leq$ *threshold*.
- **Falso negativo (FN):** Neste caso o *ground truth* não foi encontrado, ou seja, o objeto não foi detectado. Portanto não há sobreposição de predição e *ground truth*
- **Verdadeiro negativo (VN):** Do contrário do verdadeiro positivo que detecta a classe correta do *ground truth* o verdadeiro negativo é a detecção correta da classe negativa, o que pode representar um grande universo de detecções dentro de apenas uma imagem. Esse tipo de métrica geralmente não é utilizado.

A Precisão e Recall podem ser definidas da seguinte maneira:

$$Precisão = \frac{VP}{VP + FP} \tag{2.1}$$

$$Recall = \frac{VP}{VP + FN} \tag{2.2}$$

## 2.3 AVERAGE PRECISION (AP)

Como forma de avaliar o desempenho de um detector, surge a relação entre as métricas de Precisão e Recall. Essa nova maneira de avaliar o modelo julga como o modelo age de acordo

com as mudanças de valores de confiança. O detector pode ser dito como bom se os valores de Precisão e Recall permanecem altos, enquanto o valor de *threhold* varia. Se considerarmos que o valor de $VP + FN$ = todos ground truths = valor constante, temos que apenas a quantidade de $VP$ irá influenciar, pois os valores de $FN$ irão diminuir e consequentemente a precisão irá se manter em um patamar alto.

Tendo como base a curva que Precisão e Recall fazem, surge a definição de Average Precision, pois ela é calculada através da área de baixo da curva (AUC do inglês, *Area under Curve*). Na prática, o AP é a média do valor de Precisão ao longo de todos os valores de Recall de 0 a 1. Para calcular o valor de AP é possível através da segmentação em onze partes iguais ou através de uma integral, conforme as seguintes equações:

$$AP = \frac{1}{11} \sum_{Recall} Precisão(Recall) = 1 \tag{2.3}$$

$$AP = \int_0^1 Precisão(Recall)\, d(Recall) \tag{2.4}$$

### 2.3.1 Mean Average Precision (mAP)

Pela definição de Mean Average Precision, temos que ele é a média de todos os AP para cada classe $i$, a equação a seguir demonstra o cálculo da métrica:

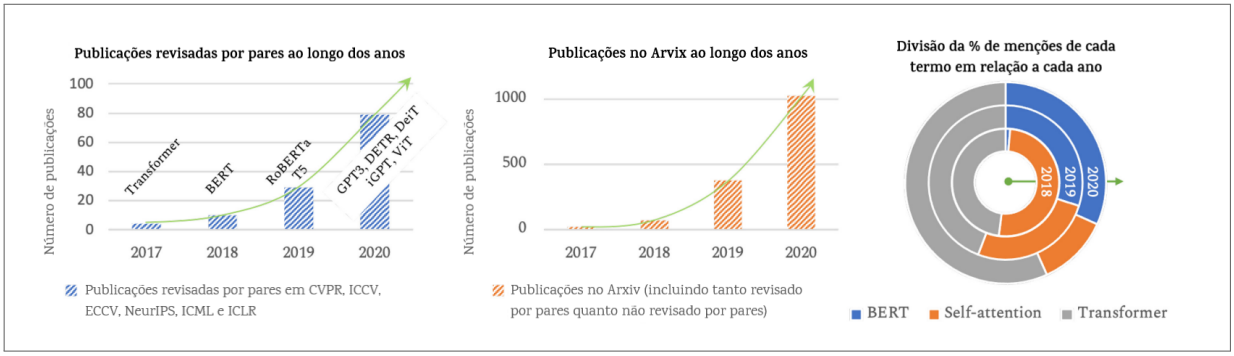
$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.5}$$

## 2.4 TRANSFORMER

O trabalho com Transformers foi proposto inicialmente por (SUTSKEVER; VINYALS; LE, 2014) que demonstrou a habilidade de modelos que se baseiam em sequências. A partir deste trabalho, houve um crescente interesse nesse método (Figura 14), que então começou a ser adotado por áreas como linguagem natural, visão computacional e processamento de fala.

Os Transformers demonstraram grande desempenho em várias tarefas, especialmente em linguagem de processamento natural, tornou-se a preferida, fazendo novos marcos do estado da arte nesse campo. Isso despertou um grande interesse na comunidade de visão computacional em adaptar essa técnica para trabalhar com imagens e também obter melhores resultados.

Figura 14 – É notável o aumento do número de publicações a respeito dos Transformers nos principais eventos de visão computacional desde 2017.



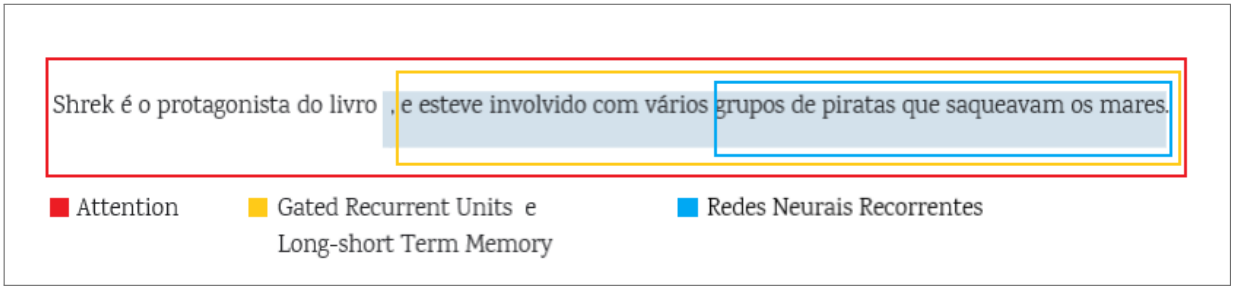

**Fonte:** Adaptado de (KHAN et al., 2022)

Em 2017, surgiu um proeminente trabalho com grande impacto nesse campo, o modelo Vision Transformer (ViT) (VASWANI et al., 2017). A ideia de concepção do modelo ViT passa pela aplicação da arquitetura de encoder do Transformer na sua estrutura, ou seja, trata-se de uma implementação genuína de modelo Transformer, sem o emprego ou intervenção de blocos convolucionais, onde após o treinamento desse modelo ele será utilizado para classificação de imagens. Inclusive, quando treinado com grandes quantidades de dados, o modelo consegue superar modelos do estado da arte que utilizam CNN, como a ResNet (HE et al., 2016), além de necessitarem de menos recursos computacionais (SARKAR, 2021).

Os Transformers são arquiteturas que funcionam com sequências de dados como conjuntos de palavras ou uma imagem dividida em vários pedaços. Dentro dos Transformers existe um mecanismo chamado self-attention que consegue interpretar as relações entre cada elemento da sequência, permitindo descobrir quão importante é um elemento em meio a outros. Para realizar a extração das características dos elementos (palavras ou cluster de imagem), nenhuma unidade recorrente é utilizada. Nesse sentido, são utilizadas apenas somas ponderadas e ativações, permitindo que os Transformers possam ser eficientes e paralelizáveis. O mecanismo de self-attention, também permite que os Transformers tenham uma memória de longo prazo extremamente grande, se comparada com Redes Neurais Recorrentes (RNN), Gated Recurrent Units (GRU's) e Long-short Term Memory (LSTM's). A Figura a seguir exemplifica um cenário hipotético das capacidade de janela de referência para cada mecanismo.

Indo mais a fundo sobre os detalhes da arquitetura dos Transformers, essa arquitetura apresenta uma interessante disposição de duas estruturas chamadas de Encoder e Decoder, ambas estruturas são constituídas de blocos empilhados de feed-forward e attention repetidos $N$ vezes (Figura 17).

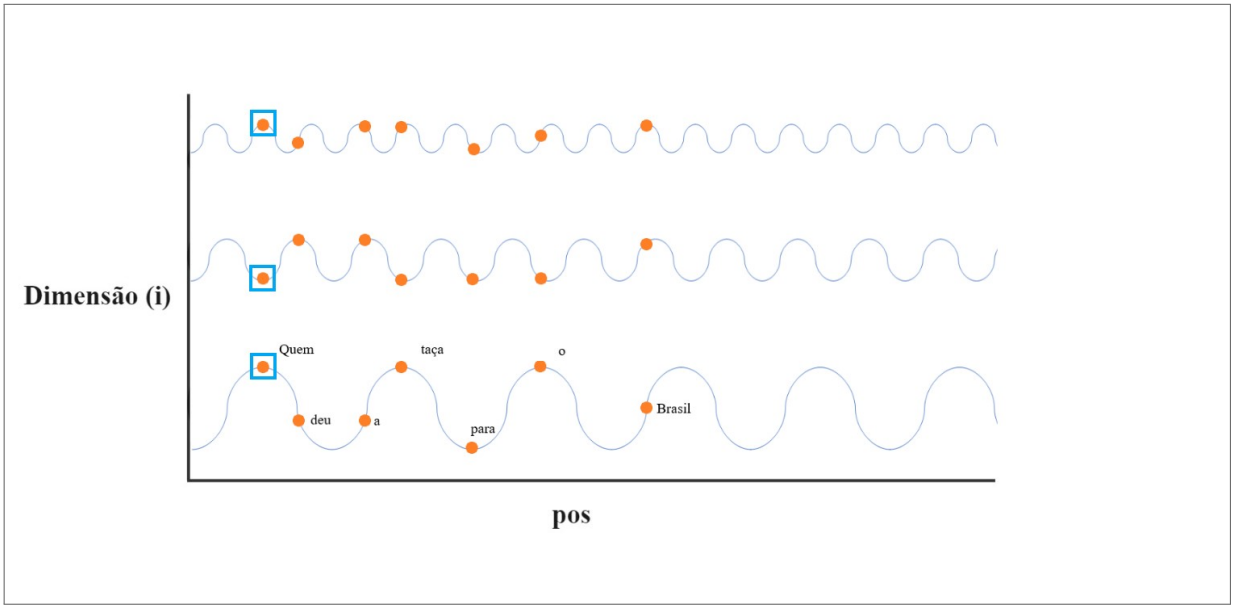
Figura 15 – Dados os cenários hipotéticos o mecanismo attention possui uma grande janela de referência sendo capaz de usar todo o contexto da história para produzir ainda mais texto




**Fonte:** Adaptado de (PHI, 2020)

Antes de passar pelo Encoder, os vetores de entrada passarão pelos embeddings para que a cada elemento da sequência seja atribuído um token que o identificará. A camada de embedding fará então a conversão desse token para um vetor com valores contínuos que representará cada elemento. Como a ordem dos elementos não importa para o modelo, foi adicionada uma técnica para a preservação da informação na sequência correta, chamada de Positional Encoding. Por não ter a recorrência das redes neurais recorrentes, os Transformers utilizam essa técnica para injetar as informações posicionais em cada embedding de entrada, onde cada elemento pode ter a mesma amplitude em uma dimensão, entretanto quando se eleva a dimensão, cada elemento terá um vetor de posições que diferirá dos outros. A Figura 16 exemplifica esse conceito utilizando funções senoidais com frequências diferentes à medida que se aumenta a dimensão.

Figura 16 – Ilustração do conceito de dimensão em relação a posição.




**Fonte:** Elaborado pelo autor

Para codificar em informação de entrada são utilizadas funções de seno e cosseno. O seno é aplicado quando a posição no vetor do elemento for par e a função cosseno quando for ímpar. A equação que descreve essas funções são as seguintes:

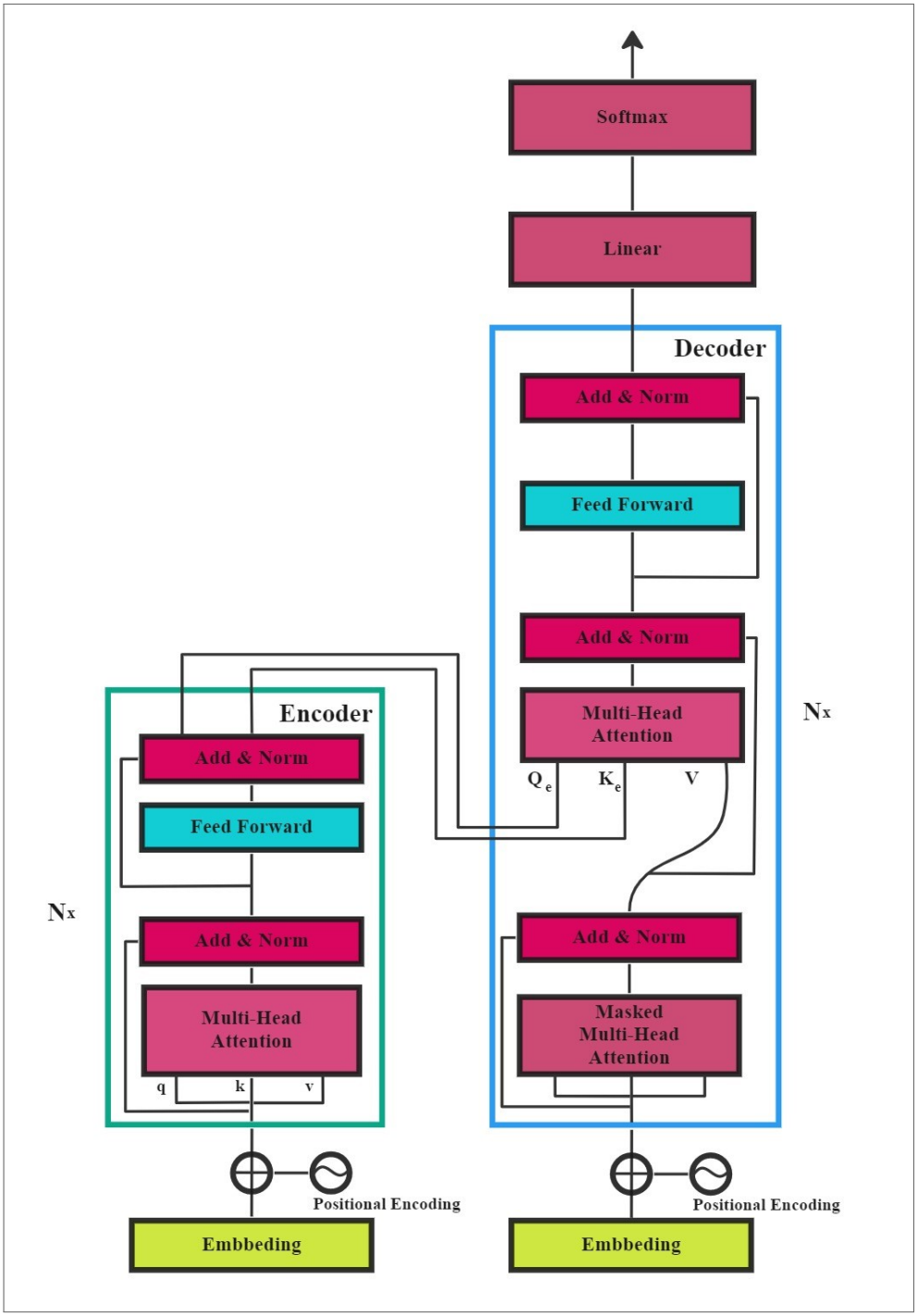
$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{model}}})$$
$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{2i/d_{model}}}) \quad (2.6)$$

A ferramenta fundamental em qualquer Transformer é a operação de self-attention. Ela possibilita que a sequência dos vetores de entrada de tamanho $k$ tenham como saída sequências de vetores com média ponderada de tamanho $k$. Esses vetores resultantes são conhecidos como $Query$ (q), $Key$ (k) e $Value$ (v). Os componentes mencionados vêm do conceito de sistemas de recuperação de informação, onde a Query é a entrada bruta, sem processamento, e que após a realização de alguns procedimentos de pré-processamento é obtido a Key que realizará uma busca no database e encontrará o melhor Value de saída.

O melhor exemplo disso é a pesquisa de um vídeo, onde é inserido uma frase sobre algum assunto e o servidor retorna o resultado que melhor combina com o pareamento da Key com o Value do vídeo. No contexto dos Transformers, existe uma pequena diferença, todos os vetores de entrada no Query, Key e Value serão iguais. Dentro do Multi-Head Attention (Figura 18), são aplicados transformações lineares nesses três vetores. Como forma de torná-los controláveis são adicionados matrizes de pesos $W_q$, $W_k$, $W_v$ que resultarão nos vetores $q_i = W_q x_i$, $k_i = W_k x_i$ e $v_i = W_v x_i$, onde $x_i$ é o vetor de entrada. Para calcular a correlação entre o $q_i$ e $k_j$ é feito o produto escalar entre os dois, que terá como resultado uma matriz de correlação. Portanto, para a posição 1 da sequência, o vetor $q_1$ multiplicará por $k_1$, depois seguirá com $q_1.k_2$, $q_1.k_3$, ..., $q_i.k_j$.

A matriz de correlação, também conhecida como filtro de attention, irá produzir a correlação de cada elemento na sequência em relação a outra sequência de mesmo conteúdo. Como esses valores podem crescer conforme a dimensão de entrada, logo eles teriam valores extremamente grandes, e isso seria um problema, pois o Softmax é muito sensitivo a valores de grande magnitude. Uma maneira que os pesquisadores encontraram de não permitir esse aumento demasiado foi realizando a multiplicação de $Q.K^T$ pelo fator de $1/\sqrt{d_k}$, onde $d_k$ é a tão somente dimensão de entrada. Ao final dessa conversão de escala, é feita por seguinte o Softmax que transforma os números para valores muito mais palpáveis entre 0 e 1, em outros termos, probabilidades. A Figura 19 mostra um caso arbitrário de como seria um filtro de

Figura 17 – Arquitetura do modelo Transformer

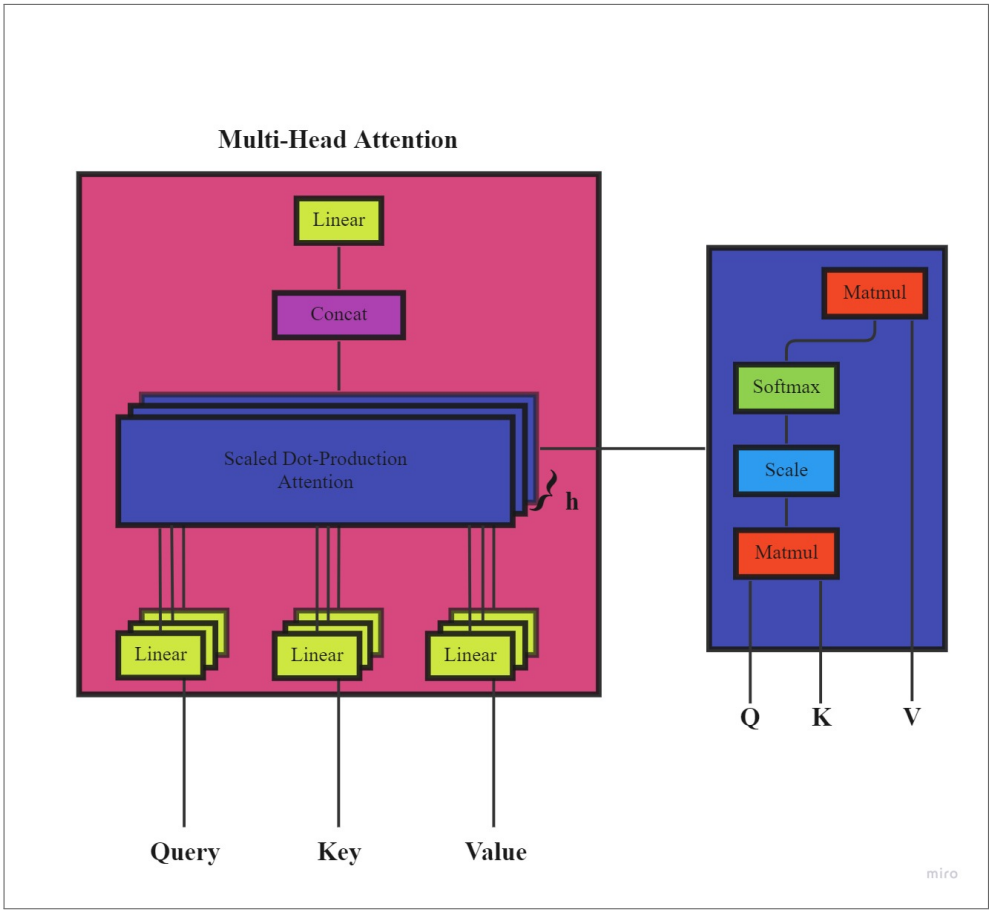



**Fonte:** Adaptado de (VASWANI et al., 2017)

attention após o Softmax, o número de dimensões para esse caso hipotético seria $d_k = 6$.

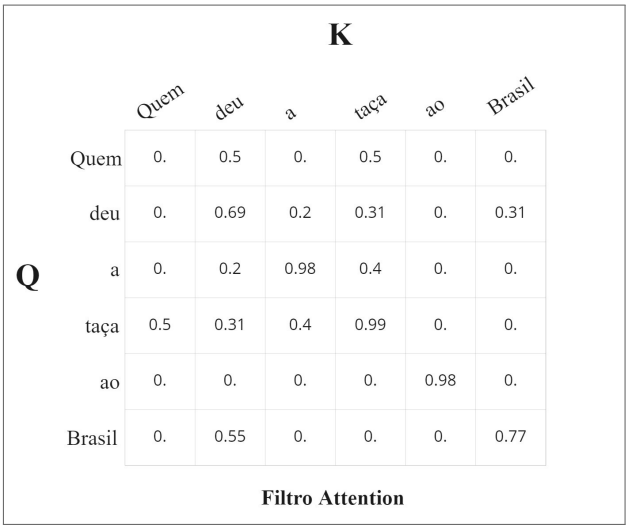
Por fim, ainda no Scale Dot-Production, os resultados vindos do filtro de attention são multiplicados pelos valores originais de $V$, a fim de se obter as características mais importantes. Perceba que o valor de $V$ foi utilizado apenas no final. Todas essas operações podem ser sintetizadas, matematicamente, da seguinte maneira:

Figura 18 – Visão geral do Multi-head com detalhamento do Scale Dot-Production




**Fonte:** Adaptado de (VASWANI et al., 2017)

Figura 19 – Cenário hipotético com valores arbitrários de um possível filtro attention



K

| Q | Quem | deu | a | taça | ao | Brasil |
|---|---|---|---|---|---|---|
| Quem | 0. | 0.5 | 0. | 0.5 | 0. | 0. |
| deu | 0. | 0.69 | 0.2 | 0.31 | 0. | 0.31 |
| a | 0. | 0.2 | 0.98 | 0.4 | 0. | 0. |
| taça | 0.5 | 0.31 | 0.4 | 0.99 | 0. | 0. |
| ao | 0. | 0. | 0. | 0. | 0.98 | 0. |
| Brasil | 0. | 0.55 | 0. | 0. | 0. | 0.77 |

Filtro Attention

**Fonte:** Elaborado pelo autor

$$Attention(Q, K, V) = Sofmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.7}$$

O caso descrito acima é particular de um bloco de Scale Dot-Production, conhecido também como Attention Head, mas como os pesquisadores tinham a intenção de obter-se o máximo de informação, para isso eles aumentaram o número de Heads $h$, funcionando de forma paralela. Essa estratégia permitiu que fossem agregadas mais informações vindas de diferentes representações, porque a mesma entrada tem valores de saída distintos para cada attention head e ao combiná-los é dado um maior poder de discriminação ao modelo. A concatenação de múltiplos attention head é descrita da seguinte forma:

$$\begin{aligned} &MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W_O \\ &\text{onde } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.8}$$

Dados os recursos computacionais necessários, em teoria o mecanismo Attention funciona perfeitamente para textos, entretanto, no caso das imagens, possuem um nível de dificuldade maior, pois, nos textos as palavras são tokenizadas antes de serem passadas para o Transformer, e ao passar pelo Transformer é adicionado o Attention. Imagens podem ter de milhares a milhões de pixels e em uma simples operação sobre uma imagem de tamanho 800*800 o mecanismo de Attention irá custar um número exorbitante de operações. Então, ao invés de usar o conjunto global de pixels, os pesquisadores decidiram utilizar clusters de pixels, ou seja, a imagem original seria dividida em pequenos pedaços com partes iguais.

## 2.5 DETECTION TRANSFORMER (DETR)

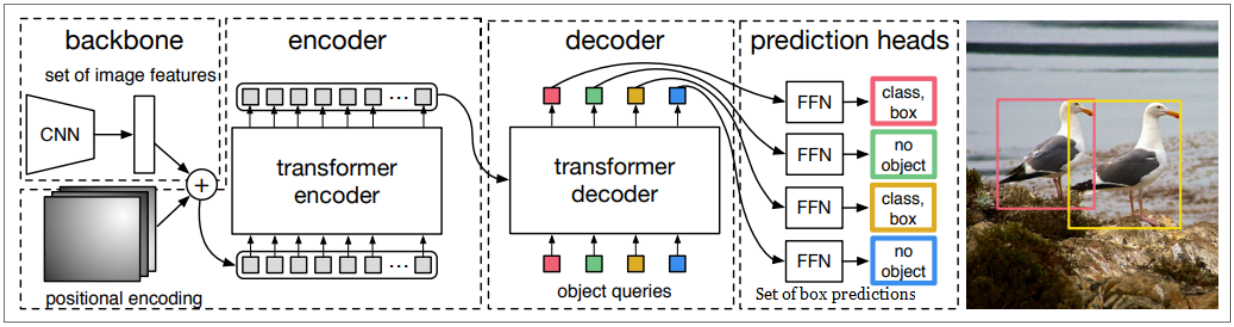
O modelo *Detection Transformer* foi desenvolvido pelo time de pesquisa do Facebook. Ele foi criado de ponta a ponta para detecção de objetos, saindo um pouco das técnicas utilizadas em arquiteturas clássicas e do "mundo autoregressivo", utilizando uma estrutura Transfomer *encoder-decoder*. Existem alguns fatos interessante nesse modelo, pois as técnicas de proposta de regiões de interesse, como *Selective Search* e RPN, empregadas nos modelos Fast R-CNN e Faster R-CNN não fizeram parte desse trabalho. Além disso, é removida a necessidade da aplicação das técnicas NMS e âncoras.

O framework do DETR é composto dos seguintes elementos:

- É colocado um *backbone* que antecede o Transformer e serve como extrator de características.

- Uma arquitetura de Transformer *encoder-decoder* adaptado do modelo de (VASWANI et al., 2017).
- Um conjunto de *Loss* global utilizando a correspondência bipartida entre as predições e o *ground-truth*.

Figura 20 – Diagrama do esquema de funcionamento e os elementos que compõe o DETR.



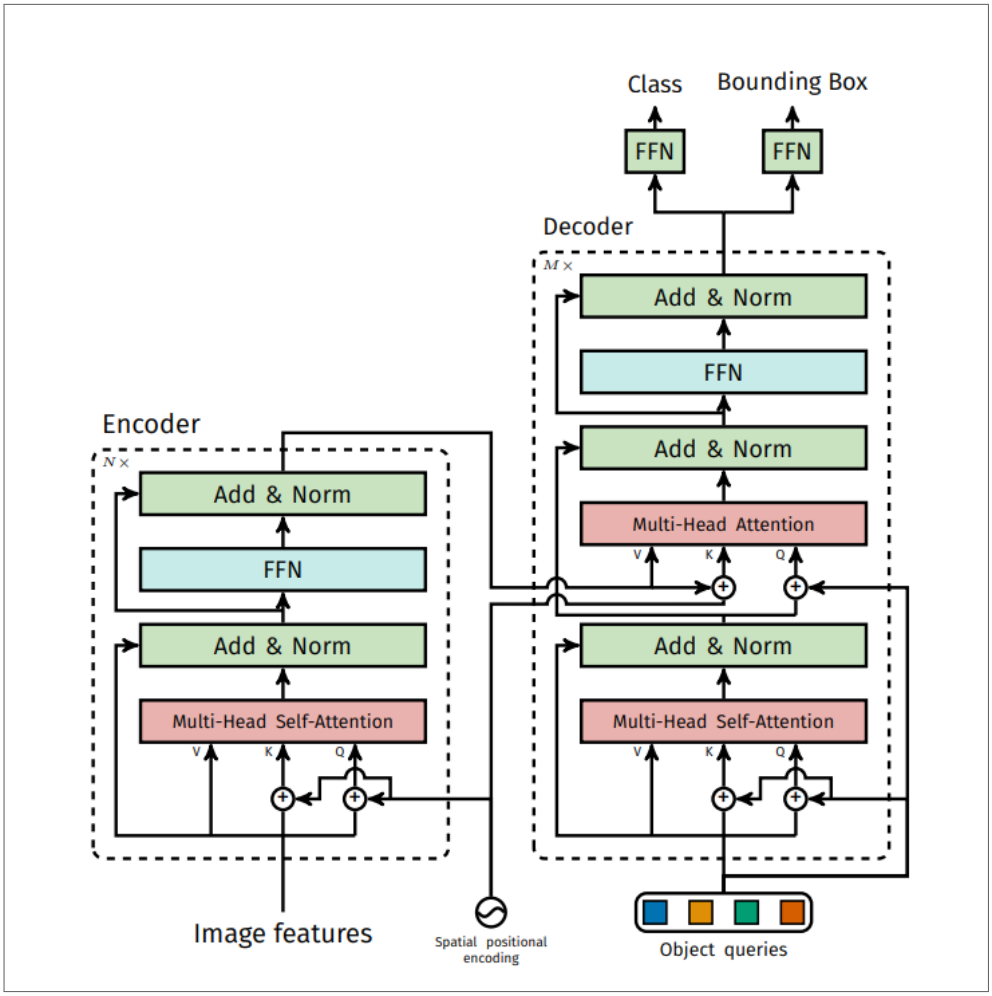

**Fonte:** Adaptado (CARION et al., 2020)

Inicialmente a imagem é passada por um *backbone* pré-treinado na ImageNet, ele pode ser tanto a ResNet-50 quanto a ResNet-101. Essa fase é importante para gerar um mapa de característica bidimensional, de modo a preservar a informação. Antes de ser passada adiante, é adicionado o codificador posicional (positional encoding) as *queries* e *keys*. Isso permitirá que o modelo estabeleça relações entre qualquer elemento em posições diferentes do vetor, mesmo que pareçam semelhantes.

Após a adição de informação posicional e o achatamento da informação, ela é passada para o *encoder* que espera como entrada uma sequência que é um mapa de característica em formato de um vetor. O estágio de *encoder* possui uma estrutura típica, assim como o *decoder*, com múltiplos módulos de *self-attention*, vide Figura 18, seguido de um normalizador, uma rede *feed forward* e novamente um normalizador. Os modelos foram constituídos de 6 encoders e 6 decoders com 8 attention heads. O *encoder* fica responsável pela codificação da informação de maneira que haja uma representação do contexto global.

A saída do encoder serve como condicionante para os *object queries*, que em suma são vetores iniciados aleatoriamente que vão se modificando conforme é aprendido algo. O decoder transforma $N$ *object queries* em uma saída chamada de *embedding* que é passada por uma rede feed forward que prediz as coordenadas normalizadas do centro, altura e largura em relação à imagem. Também é passada através de uma camada linear que prediz a classe utilizando *Softmax*.

Figura 21 – Arquitetura DETR e sua forma Transformer de forma detalhada




**Fonte:** (CARION et al., 2020)

O modelo DETR tem uma limitação na quantidade de predições, a cada imagem que passa o decoder gera $N$ predições de uma vez. Esclarecendo que $N$ é o valor limitante e geralmente tem o valor definido para ser maior que a quantidade de objetos na imagem analisada e tem como nome *object queries*. Uma classe especial se encarrega de agregar todas as não detecções de objetos.

Esse trabalho traz uma proposta de *Loss* global, fazendo pareamento entre o conjunto de predições ($\hat{y}$) e o conjunto do *ground truth* ($y$), com $N$ permutações das predições, na intenção de encontrar uma função injetora onde cada predição tem apenas um par com o *ground truth* ao menor custo possível. O custo do pareamento é calculado da seguinte maneira:

$$\hat{\sigma} = \underset{\sigma \in S_N}{argmin} \sum_{i}^{N} \mathcal{L}_{pareamento}(y_i, \hat{y}_{\sigma(i)}) \tag{2.9}$$

De maneira que pudesse permitir a computação do melhor pareamento e que levasse em

conta a classe e as coordenadas, foi utilizado o algoritmo húngaro que assim nos habilita a ter a *Loss* de todos os pares. A combinação linear da função de log-verossimilhança negativa juntamente com a *Loss* das caixas de predições, que também é a combinação linear da função *Loss* L1 e o generalized IoU (GIoU) podem ser definidos como:

$$\mathcal{L}(y, \hat{y}) = \sum_{i}^{N} [-log\hat{p}_{\hat{\sigma}(i)}(c_i) + 1_{c_i \neq \varnothing} \mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}(i)})] \tag{2.10}$$

## 2.6 AUGMIX

Desenvolvido pela equipe de pesquisa do Google, o AUGMIX apresentou um bom desempenho no ImageNet (DENG et al., 2009), e existem outros trabalhos promissores na área de classificação, como (NASEER et al., 2021), (SCHNEIDER et al., 2020), (TAORI et al., 2020) e (HENDRYCKS et al., 2021). O AUGMIX é uma técnica poderosa de aumento de dados que pode ser facilmente incorporada ao processo de treinamento de um modelo de aprendizado de máquina. Ele gera augmentations estocásticas e diversas, encadeando uma série de transformações, resultando em uma imagem aumentada diferente a cada vez. Esse alto nível de diversidade ajuda a evitar que o modelo sofra overfitting e memorize augmentations fixas. Além disso, a natureza estocástica do AUGMIX garante que o modelo não seja influenciado por nenhuma augmentation fixa específica, tornando-o mais robusto e eficaz.

Portanto, é possível melhorar os classificadores de detecção de objetos ao focar em métodos que detectam efetivamente objetos e possuem uma rápida taxa de convergência. Como referência para nossa abordagem, utilizaremos um método específico com essas características e o mesclaremos com o algoritmo AUGMIX. Essa combinação nos permitirá aproveitar as vantagens de ambos os métodos, resultando em um sistema de detecção de objetos mais robusto e preciso.

Nossa abordagem visa melhorar a robustez do modelo aumentando sua resistência a mudanças na distribuição dos dados, garantindo perdas consistentes e tornando-o insensível a entradas de tamanhos diferentes. Para alcançar isso, adicionamos a divergência Jensen-Shannon (JS) ao esquema de aumento de dados. A divergência JS é uma versão simétrica e suavizada da divergência Kullback-Leibler (KL)

Matematicamente, podemos representar a distribuição das variantes geradas $X_{orig}, X_{augmix_1}$ e $X_{augmix_2}$ da seguinte maneira $p_{orig} = \hat{p}(y|X_{orig}), p_{augmix_1} = \hat{p}(y|X_{augmix_1}), p_{augmix_2} =$

$\hat{p}(y|X_{augmix_2})$ ao substituir a Loss original $\mathcal{L}$ temos:

$$\mathcal{L}(p_{orig}, y) + \lambda JS(X_{orig}; X_{augmix_1}; X_{augmix_2}) \tag{2.11}$$

As informações dessas três distribuições são extraídas de cada imagem gerada, revelando, em média, quão diferentes as distribuições das variantes criadas são da distribuição original. O uso da média é necessário, pois as distribuições têm uma resposta média. A média ponderada não foi utilizada, pois todas as distribuições têm contribuições iguais. Por exemplo, a única razão para atribuir pesos maiores a uma imagem com augmentations de solarize ou posterize é se isso for feito estocasticamente. Portanto, para calcular a perda de JS, é necessário $M = \frac{1}{3}(p_{orig} + p_{augmix_1} + p_{augmix_2})$, o que torna a perda mais estável. Tendo calculado *M*, a perda pode ser computada usando KL para cada distribuição da seguinte forma:

$$JS(X_{orig}; X_{augmix_1}; X_{augmix_2}) = \frac{1}{3} \times (KL[p_{orig}||M] + KL[p_{augmix_1}||M] + KL[p_{augmix_2}||M]) \tag{2.12}$$

O objetivo de usar essa divergência é buscar semelhança entre as distribuições geradas pelas transformações realizadas, sempre minimizando essa diferença. Como resultado, o conteúdo semântico da imagem sofre um mínimo de deterioração, tornando-a muito próxima do que era antes. Conforme descrito em (HENDRYCKS et al., 2019), várias arquiteturas de redes convolucionais se beneficiam da entrada desses dados devido à robustez proporcionada a elas após as transformações, e como os modelos Transformer discutidos aqui possuem redes convolucionais, essa característica se aplica igualmente a eles.

O AUGMIX é utilizado como uma operação simples de aumento para aumentar a diversidade do conjunto de dados. A técnica utiliza transformações simples de imagem, como translação e deformação nos eixos x e y, além de rotação, posterização e solarização, por exemplo. Cada função de transformação recebe uma imagem e um parâmetro de nível que controla a intensidade da transformação. Esse parâmetro de nível é equivalente ao parâmetro de severidade descrito no artigo (HENDRYCKS et al., 2019) e varia de 1 a 10, sendo que 10 indica o aumento mais intenso possível.

# 3 METODOLOGIA PROPOSTA

Este capítulo trará um detalhamento da metodologia proposta para os modelos de detecção de objetos. A primeira parte é propriamente dedicada na constituição e configuração de um conjunto de dados que permita uma avaliação do desempenho dos modelos em relação a detecção de objetos. Além disso, foi modificada a estratégia de *augmentation* do modelo Deformable DETR para que fossem possível a inserção do AUGMIX, respeitando as capacidades de Hardware disponíveis. O resultado do protocolo de experimentos realizados é demonstrado no capítulo 4, onde é possível fazer uma comparação do desempenho com outros dois modelos utilizados aqui, isto é, a forma originária do Deformable - o DETR, e a aplicação do modelo estágio único, EfficientDet. Apesar de ter sido utilizado um *dataset*, que na realidade é a junção de vários outros sobre o mesmo tema, o modelo permanece inalterado para utilização em outros tipos de aplicação com outros objetos. Outro ponto importante que é a questão de ter sido utilizado apenas um *dataset* será discutida adiante.

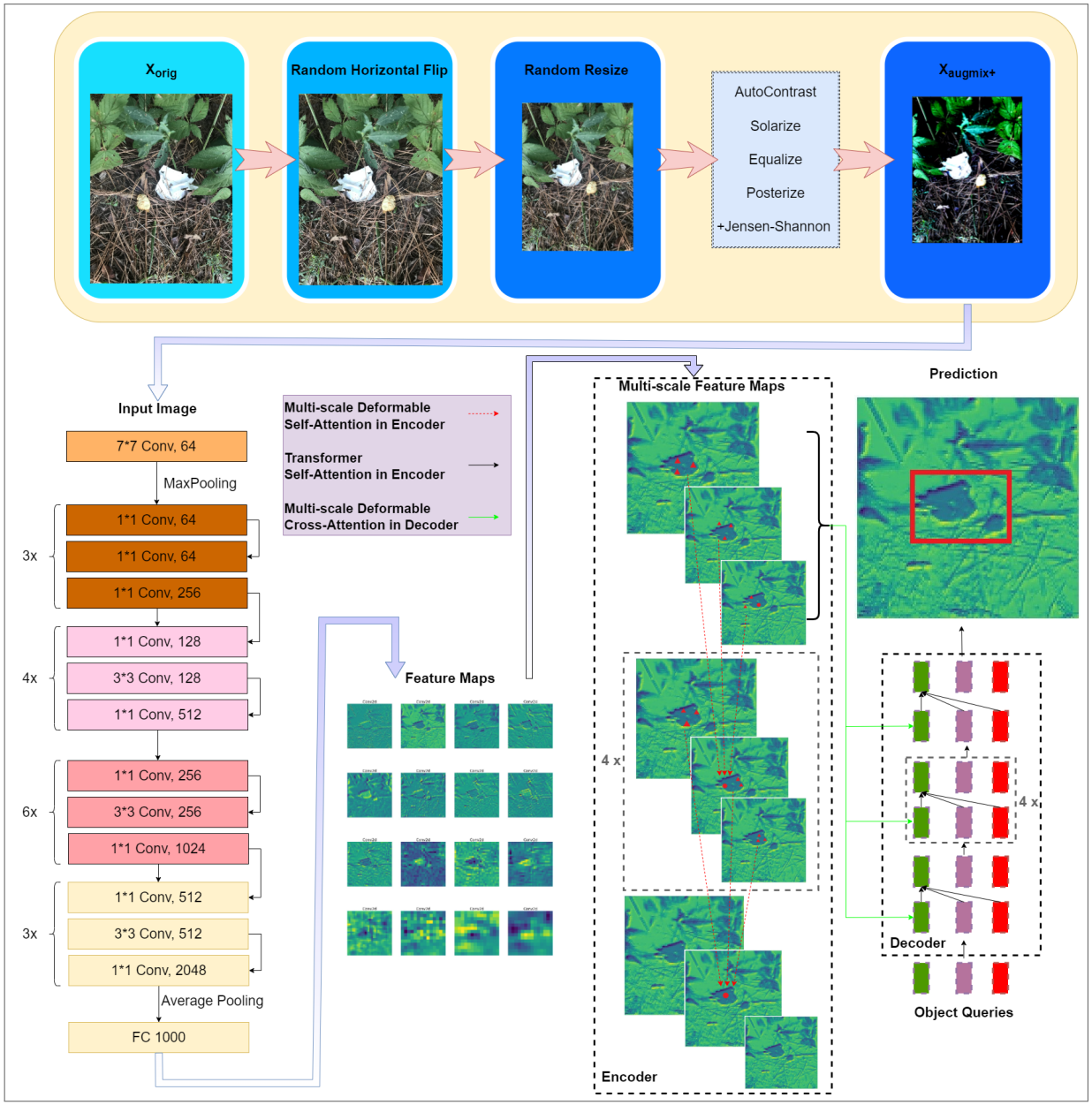
## 3.1 VISÃO GERAL

O ajuste no modelo ocorreu a partir do *pipeline* de transformações que, originalmente, já eram feitas no modelo DETR e, por consequência, estão presentes no Deformable, tendo em vista que este incorpora o processo de treinamento do modelo anterior.

A Figura 22 ilustra o fluxo de algumas das transformações encadeadas que ocorrem desde o processamento de imagem até a predição. Parte da implementação original foi removida e a melhor configuração é a apresentada na imagem. Algumas das transformações do AUGMIX que afetaram espacialmente a imagem não foram utilizadas, pois afetaram negativamente o desempenho quando aplicadas em conjunto com as transformações espaciais originais. Além disso, quando todas as transformações originais foram eliminadas e apenas o AUGMIX permaneceu, o desempenho do AUGMIX permaneceu praticamente o mesmo, às vezes inferior, com uma diferença mínima em comparação com o original.

A seguir, de forma breve, são descritos quais foram as etapas que se sucederam para a completa execução da pesquisa, incluindo a parte de preparação dos dados que não está presente na Figura 22, pois esta traz aspectos mais relacionados a aplicação da técnica com os detalhes visuais de cada atividade.

Figura 22 – Ilustração do fluxo da abordagem proposta. Ele mostra o fluxo de processamento de imagem que começa com horizontal invertendo, seguido de redimensionamento e, em seguida, incorporando operações AUGMIX. Estas operações visam aumentar a robustez do modelo. Após essas etapas de pré-processamento, as imagens passam por uma CNN para extração de suas características, que então alimentam um codificador. O codificador realiza a autoatenção multi-head, o que ajuda na contextualização dos recursos da CNN. O decodificador, composto de consultas, interage com os recursos do codificador e agrega informações por meio de atenção cruzada multicabeça para gerar a saída final.




**Fonte:** Elaborado pelo autor

**Etapa 1:** Como etapa inicial, vamos considerar o estágio de junção dos *datasets* individuais, haja vista que para esse trabalho foram considerados 12 repositórios que continham *datasets* isolados. Essa etapa é fundamental, pois a constituição de um *dataset* é uma das ferramentas que permitirá a geração de resultados dos modelos e uma análise do comportamento dos

diferentes modelos com base nesse mesmo *dataset*.

**Etapa 2:** Ajuste no corpo do código para agregação do AUGMIX. O código original com o *pipeline* de treino encontrado no repositório dos modelos DETR e Deformable segue o esquema de código presente no código fonte 3. Tendo isso como base, o AUGMIX em si é uma coletânea de transformações que podem ser condensadas em um único arquivo, dadas as devidas alterações no código, é possível inseri-las posteriormente dentro do código.

**Etapa 3:** Tendo o *dataset* preparado, é possível dar início aos treinamentos. De início, os modelos são inalterados e postos em funcionamento, ficando o para o pós-processamento a parte importante desta etapa, pois é onde se irá averiguar as capacidades de cada modelo a partit dos resultados obtidos, e a partir deste ponto intervir com as modificações no escopo das transformações nas imagens.

Além disso, como este trabalho é influenciado pela limitação orçamentária, que permitiu ter a disponibilidade de apenas uma placa gráfica GPU NVIDIA A100 40GB alugada no serviço de computação científica da Google Colab, os modelos foram modificados para que pudessem rodar nesse cenário. Por fim, os códigos desenvolvidos nesta pesquisa estão presentes em (CUNHA; MACÊDO; ZANCHETTIN, 2023).

Como resultado, temos que a presente dissertação objetiva propor uma alternativa aos modos de transformações que antecedem o processo de treinamento - o AUGMIX, permitindo que os modelos não sejam fortemente afetados por *overfitting*, por exemplo. As seções seguintes trarão maiores detalhes de cada etapa proposta.

## 3.2 MANIPULAÇÕES DOS DATASETS

Com o intuito de demonstrar que o desempenho dos modelos em imagens com objetos de tamanhos variados, foi crucial a decisão da escolha do *dataset*. Para isso, era necessário que, na maioria das imagens, houvessem os elementos que satisfizessem esse requisito. Além disso, por si só, o *dataset* deveria conter diversidade no seu conteúdo e, considerando as limitações de hardware, ter um tamanho viável de repetição do protocolo experimental.

Portanto, para satisfazer esses requisitos, a constituição do *dataset* principal foi realizado pela unificação de oito *datasets* de domínio público. Os *datasets* escolhidos são variados, como ambientes na natureza, ambiente urbano e até doméstico. Os objetos que geralmente estão nesses lugares podem aparecer de formas variadas em diferentes contextos, o que permite treinar modelos nessas situações e avaliar seu poder de generalização em casos parecidos. Em

cima dos dados que foram utilizados nesta pesquisa, houve outra pesquisa feita por (MAJCHROWSKA et al., 2021) que propôs um framework para fazer a classificação e detecção de objetos. A seguir é feita uma descrição de cada *dataset* que compôs essa pesquisa:

- **TrashNet:** É resultado de um trabalho feito por (YANG; THUNG, 2016). Ele propõe, entre outras coisas, a formação de um banco de dados com imagens de múltiplas categorias de lixo. As imagens foram obtidas manualmente com o uso de smartphones em estúdio dos objetos que geralmente são encontrados no meio ambiente e, portanto, considerados como lixo. Durante o uso deste *dataset* para os experimentos, existiam 2.100 imagens com tamanho de 512x384 que estavam distribuídas entre seis classes, como: vidro, papel, papelão, plástico, metal e lixo.

- **TACO versão extendida:** Consiste de um *dataset* criado por (PROENÇA; SIMOES, 2020) para ser de domínio público e desenvolvido pela comunidade da internet. Inicialmente, o *dataset* era composto de 1.500 imagens de alta resolução com as respectivas anotações, entretanto existiam 3.000 imagens sem anotações, anotações essas que foram posteriormente acrescentadas por (MAJCHROWSKA et al., 2021). As anotações seguem o padrão do formato COCO e não possuem um tamanho fixo para as imagens.

- **UAVVaste:** É um *dataset* criado fundamentalmente do ponto de vista de um drone com dados coletados dos centros urbanos e dos ecossistemas ambientais. Esse conjunto de dados traz anotações no formato COCO, além de conter 772 imagens e suas respectivas anotações, conta com uma única classe (KRAFT et al., 2021).

- **TrashCan e Trash-IRCA:** Ambas contêm imagens do fundo do mar com as imagens possuindo iluminação variável e até oclusão da visão. Além disso, as imagens também captam a flora e fauna presentes no meio marinho juntamente com o lixo presente no local. As imagens são extraídas de frames de vídeo e possuem dois tamanhos possíveis, sejam 480x270 e 480x360. O trabalho de (HONG; FULTON; SATTAR, 2020) com Trash-Can e (FULTON; HONG; SATTAR, 2020) com Trash-IRCA possuem, respectivamente, 7.212 imagens com 16 categorias e 7.668 imagens com 7 categorias.

- **Drinking Waste:** Como o nome sugere, ele traz uma abordagem com imagens de objetos que geralmente são produtos que contêm bebida. (SEREZHKIN, 2020) propôs o uso de 4 classes para classificar as 4800 imagens, que assim como o TrashNet, todas as imagens foram feitas dentro de um estúdio com smartphone.

- **MJU-Waste:** (WANG et al., 2020) produziu um *dataset* que se diferencia dos outros por trazer o poder que os sensores RGBD conseguem extrair dos objetos, pois além das cores, ele traz a profundidade do elemento, a captura das imagens foi feita utilizando uma câmera da Microsoft Kinect. No total existem 2475 imagens, que foram tiradas manualmente e, os objetos estão nas mãos de pessoas.

- **Wade-AI:** (HAAMER, 2018) propôs a criação de um *dataset* a partir das imagens do Google Street View, essa tarefa foi possível utilizando o modelo de detecção que encontrasse pilhas de lixo nos mais variados lugares. Esse banco de imagens é composto de 1.396 imagens, além de possuir uma única classe para determinar os objetos da imagem. Atualmente existe uma nova versão desse *dataset* chamado *Trash-AI*.

Em nossos experimentos, usamos o conjunto de dados *detect-waste*, que consistia em aproximadamente 19.000 imagens para treinamento e 5.000 imagens para validação. As classes de objetos nesse conjunto de dados incluem itens como papel, vidro, garrafas e plástico. Para avaliar o desempenho do nosso modelo, todas as classes de objetos foram combinadas em uma única categoria chamada ***litter***, representando todos os tipos de resíduos presentes no conjunto de dados. Uma breve introdução do conjunto de imagens é delineada na Figura 23.

Figura 23 – A diversidade e a complexidade do conjunto de dados públicos do *detect-waste* é demonstrado na imagem a seguir. Algumas imagens foram selecionadas aleatoriamente para exemplificar o desafio e a aplicabilidade no mundo real da abordagem proposta.

**Fonte:** Elaborado pelo autor

## 3.3 AUGMIX DENTRO DO DETR

Para realizar os avanços que propomos no modelo, foi necessário descobrir como funcionava o fluxo de treinamento e todas as transformações feitas durante o tratamento das imagens. Durante os experimentos, foi entendido que o modelo geralmente roda em oito GPUs. Desta maneira, era necessário adaptar o código para que pudesse rodar em apenas uma GPU, o que foi possível através do comando 1. Como o *dataset* possui um tamanho pequeno, em termos de imagens, quando comparados com outros como COCO e ImageNet, foi configurado que os modelos teriam um limite de 70 épocas, pois sempre que esse limite era ultrapassado os modelos não obtinham melhora, acrescente a esse limite o tamanho de *batch* deixado em 4, tamanho que permitia a realização dos experimentos sem que a GPU ficasse sem memória.

Código Fonte 1 – Exemplo do código de execução do treinamento

```
!python main.py --dataset_file custom --data_path /content/coco --output_dir /
    content/Output --batch_size 3 --epochs 10
```

**Fonte:** Elaborado pelo autor

Além disso, antes do passo anterior, foi preciso ajustar para a versão que funciona em compatibilidade com a GPU NVIDIA A100, que até o momento é a versão torch 1.8.0+cu111 juntamente com torchvision 0.9.0+cu111 e torchaudio 0.8.0. Como juntamos todos os datasets e definimos usar apenas uma classe para todos os elementos de detecção, foi necessário adaptar o arquivo de *deformable_detr.py*, pois o mesmo utiliza a configuração para receber 90 classes, o qual corresponde ao número de classes do dataset COCO.

Na linha 445, dentro da classe *main*, retiramos o que estava presente e colocamos a quantidade de classes que usaríamos, como descrito no código 2. Como é possível perceber, o número de classes está em 2. Isso significa que uma das classes é utilizado para a classe ***litter*** e o outro é para ***no-object***, que é a classe de detecção para objetos que não fazem parte do que se procura detectar.

Código Fonte 2 – Ajuste do código para permitir a quantidade de classes utilizada

```
def build(args):
    #num_classes = 20 if args.dataset_file != 'coco' else 91
    #if args.dataset_file == "coco_panoptic":
    #    num_classes = 250
    num_classes = 2
    device = torch.device(args.device)
```

**Fonte:** Elaborado pelo autor

Analisando os arquivos, foi possível identificar que dentro do arquivo *coco.py* são feitas as relações entre imagens e as anotações de cada, relações essas que servem para ajustar as anotações da imagem após as transformações para que não haja desalinhamento do objeto e as anotações do *bounding boxes*.

Feitas as combinações entre as imagens e anotações, o que vem na sequência são as definições de normalização dos tensores, a criação de um arranjo de escalas variando entre 480 e 800, esse é o valor mínimo que a imagem poderá assumir sendo o máximo o tamanho de 1.333. A implementação original utiliza seis métodos de aumento nos dados, variando entre *RandomHorizontalFlip* *RandomSelect*, *RandomResize* e *RandomSizeCrop* como descrito no excerto de código 3.

Código Fonte 3 – Segmento original do código de rotina de treino e validação

```
import datasets.transforms as T

def make_coco_transforms(image_set):

    normalize = T.Compose([
        T.ToTensor(),
        T.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])

    scales = [480, 512, 544, 576, 608, 640, 672, 704, 736, 768, 800]

    if image_set == 'train':
        return T.Compose([
            T.RandomHorizontalFlip(),
            T.RandomSelect(
                T.RandomResize(scales, max_size=1333),
                T.Compose([
                    T.RandomResize([400, 500, 600]),
                    T.RandomSizeCrop(384, 600),
                    T.RandomResize(scales, max_size=1333),
                ])
            ),
            normalize,
        ])

    if image_set == 'val':
        return T.Compose([
            T.RandomResize([800], max_size=1333),
            normalize,
        ])
```

```
32      raise ValueError(f'unknown {image_set}')
```

**Fonte:** Elaborado pelo autor

Algumas alterações foram necessárias para implementação do AUGMIX dentro da rotina de transformações, começando pela criação de um arquivo personalizado do *coco.py*, as alterações passam pelo acréscimo de um valor da variação de escalas, mantemos a transformação de *RandomHorizontalFlip* e *RandomResize*, as outras transformações não tiveram boa compatibilidade com o AUGMIX e chegaram a causar erros na execução. A redução do tamanho máximo de 1333 para 1024 foi necessária, pois de alguma forma estava influenciando no tamanho total e o *batch* de 4 já não era capaz de suportar, causando erro de memória insuficiente.

Código Fonte 4 – Segmento modificado do código de rotina de treino e validação

```
import datasets.transforms as T

def make_custom_transforms(image_set):

    normalize = T.Compose([
        T.ToTensor(),
        T.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])

    scales = [448, 480, 512, 544, 576, 608, 640, 672, 704, 736, 768, 800]

    if image_set == 'train':
        return T.Compose([
            T.RandomHorizontalFlip(),
            T.RandomResize(scales, max_size=1024),
            T.AugMix(),
            normalize,
        ])

    if image_set == 'val':
        return T.Compose([
            T.RandomResize([800], max_size=1333),
            normalize,
        ])

    raise ValueError(f'unknown {image_set}')
```

**Fonte:** Elaborado pelo autor

A Figura 24 demonstra através da comparação do antes e depois das transformações na imagem feitas pelo AUGMIX. Vale salientar que os procedimentos variam a cada passagem da

Figura 24 – Amostra das diversas e múltiplas transformações das imagens. A abordagem cria várias variações para uma única imagem, resultando em um conjunto diversificado de imagens transformadas para treinar o modelo.


**Fonte:** Elaborado pelo autor

imagem pela técnica, o que significa que se você passar várias vezes uma mesma imagem pela técnica de aumento para cada saída a transformação da imagem será diferente das outras.

# 4 RESULTADOS

Neste capítulo serão apresentados os resultados obtidos nas etapas descritas no capítulo anterior, bem como a análise dos resultados de forma a validar a metodologia proposta para a melhoria de detecção de objetos em imagens.

## 4.1 MÉTRICAS E MODELOS

Para avaliar as alterações propostas no modelo Deformable DETR, foram escolhidos mais dois modelos. Um deles é o próprio DETR, que é a versão que deu origem aos modelos que seguem esse tipo de arquitetura para detecção de objetos com desempenho semelhante a modelos de dois estágios. O outro modelo, EfficientDet, da categoria dos detectores de estágio único, é uma versão criada com os *backbones* da EfficientNet utilizando BiFPN.

Como critério para avaliar os experimentos foram utilizadas as métricas de mAP, $AP_s$, $AP_{50}$ e $AP_l$, todas compõem o conjunto de métricas para avaliação de detecção da base de dados COCO (LIN et al., 2014), além do detalhamento de quantidade de parâmetros e épocas.

## 4.2 EXPERIMENTOS

Os experimentos permitiram avaliar a combinação do AUGMIX com o Deformable DETR. Conforme mostrado na Tabela 1, a arquitetura EfficientDet precisou de menos da metade do número de épocas para convergir, o que está relacionado ao número de parâmetros, que neste caso é aproximadamente 5 vezes menor do que as outras arquiteturas que utilizam um Transformer. Portanto, isso revela a eficiência de realizar a mesma tarefa com menos parâmetros. Em certas métricas, na comparação direta com o DETR, o EfficientDet apresenta desempenho similar e até superior em AP, $AP_m$, $AP_l$. No entanto, o desempenho com objetos pequenos $AP_s$ tem sido bastante baixos.

As variantes do modelo, como refinamento iterativo das *bounding boxes* e Deformable DETR em duas etapas, não foram utilizadas neste trabalho; aqui estão apenas as formas mais básicas desses modelos. Outro ponto em especial, é que apenas dois modelos aparecem na Figura 26, isso se deu pelo fato de que a saída do modelo EfficientDet não permitiu uma captura da evolução do mAP a cada época, embora houvesse um arquivo final que mostrasse

Tabela 1 – Comparação dos modelos no conjunto de validação do dataset *detect-waste*. A variação Deformable DETR* significa que o modelo possui 300 *queries* e Deformable DETR$^+$ representa 300 *queries* junto com o AUGMIX.

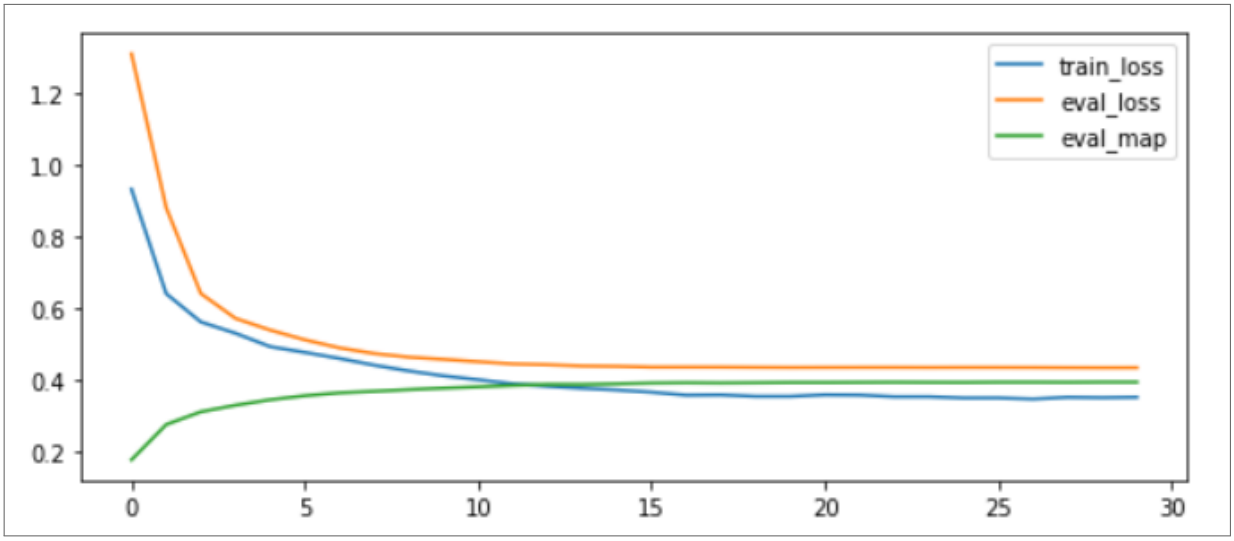
| Modelo | Épocas | Parâmetros | AP | $AP_{50}$↓ | $AP_s$ | $AP_m$ | $AP_l$ |
|---|---|---|---|---|---|---|---|
| EfficientDet | 30 | 8M | 39.6 | 54.4 | 6.8 | 42.2 | 55.9 |
| DETR | 63 | 41M | 35.8 | 54.9 | 10.1 | 32.8 | 47.0 |
| Deformable DETR | 63 | 39.7M | 46.4 | 67.5 | 18.2 | 43.3 | 58.7 |
| Deformable DETR* | 63 | 39.8M | 47.1 | 68.0 | 18.6 | 43.6 | 59.3 |
| Deformable DETR$^+(nosso)$ | 63 | 39.8M | **48.0** | **70.3** | **19.0** | **45.7** | **59.8** |

**Fonte:** Elaborada pelo autor

as curvas de *Loss* e mAP, que não era compatível com o formato de saída dos outros dois, dado esse fato, essa falta de comparação visual das curvas entre os três modelos não interferiu na avaliação final de qual foi o desempenho de cada um, pois todos possuem uma saída com os valores de cada uma das doze métricas COCO, de onde foram retirados praticamente todos os dados que formam a Tabela 1. Com os dados que o EfficientDet forneceu no arquivo final, foi possível gerar as curvas de *Loss* e mAP e, é possível verificar que entre a 10° e a 15° época o modelo começa a estabilizar, conforme visto na Figura 25, tendo conhecimento disto, limitamos para que modelo fosse até 30 épocas, isso pode ser o ponto forte desse modelo a convergência bem mais rápida que os outros, entretanto o modelo não teve um desempenho bom em relação ao mAP, reforçando a ideia de que arquiteturas de estágio único geralmente perdem em precisão quando comparadas com arquiteturas de dois estágios.

Quando comparamos o Deformable DETR com o segundo melhor desempenho, vemos que mesmo sem a otimização do AUGMIX, o Deformable se torna o melhor em todas as métricas. Dando uma atenção especial ao $AP_s$, lado a lado, o DETR e nosso modelo apresentam uma diferença de 8.9%. Como é sabido, o DETR não funciona bem com objetos pequenos e o EfficientDet mostrou uma eficiência baixa, ficando pelo menos 2.7 vezes atrás do rank-1. Portanto, para uma comparação justa, fizemos uma comparação do Deformable contra ele mesmo. A configuração padrão do Deformable utiliza 100 *queries*. Nesse caso, o $AP_s$ alcançado é de 18.2, mas quando aumentamos o número de *queries* de 100 para 300, podemos perceber um aumento de 2.19%, passando de 18.2 para 18.6 na mesma métrica. Por fim, incluir o AUGMIX sem aumentar o número de queries de 300 foi responsável por um ganho de 2.15%,

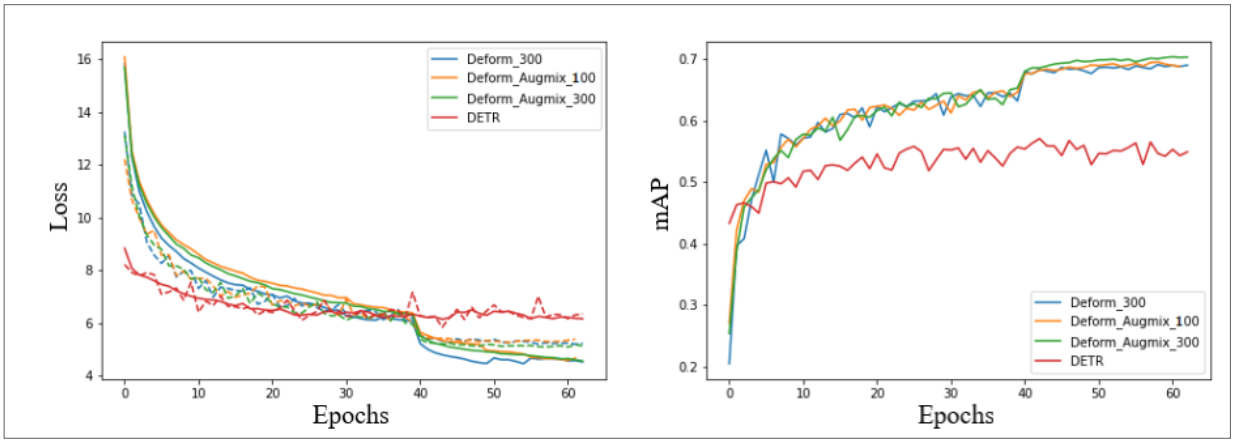
Figura 25 – Desempenho do modelo EfficientDet quanto a sua *Loss* e mAP.




**Fonte:** Elaborado pelo autor

resultando em $AP_s$ =19.0.

Figura 26 – A figura mostra a convergência da *loss* e do mAP para os modelos Deformable DETR e DETR em função das épocas. Os resultados indicam que o uso do AUGMIX resultou em uma redução na variação nas curvas de perda e em uma melhoria na consistência do início ao fim.

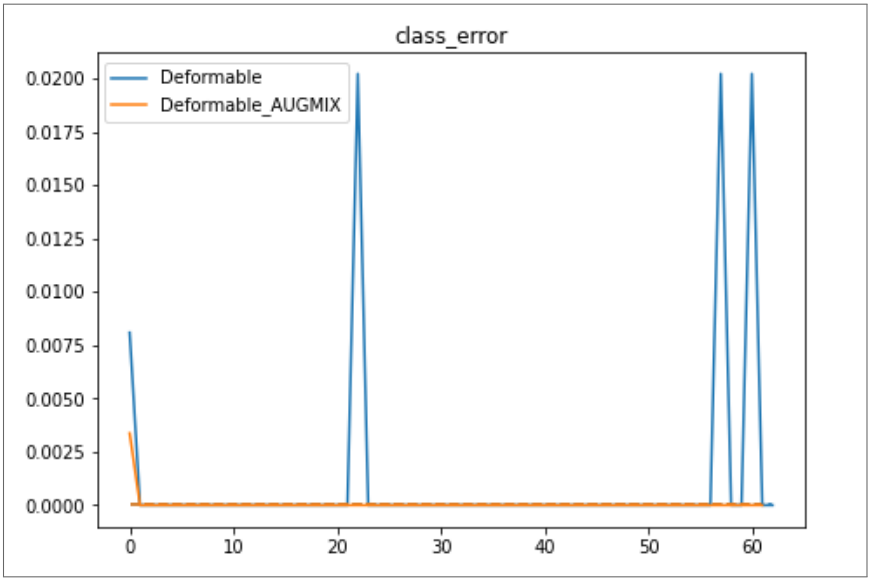



**Fonte:** Elaborado pelo autor

Além disso, o AUGMIX foi responsável pelo desaparecimento das detecções consideradas "no-object", que geralmente representam o plano de fundo do que está sendo observado. A Figura 27 mostra a curva de perda com a diminuição das detecções mencionadas, mas ainda foi possível encontrar algumas detecções de class_error nas imagens, o que significa que a técnica pode mitigar, não eliminar totalmente, esse tipo de detecção.

Isso também pode ser uma das razões pelas quais o Deformable teve um desempenho melhor do que os outros modelos, considerando que no conjunto de dados existem muitas imagens com objetos pequenos e isolados em um ponto da imagem. Além disso, ter mais

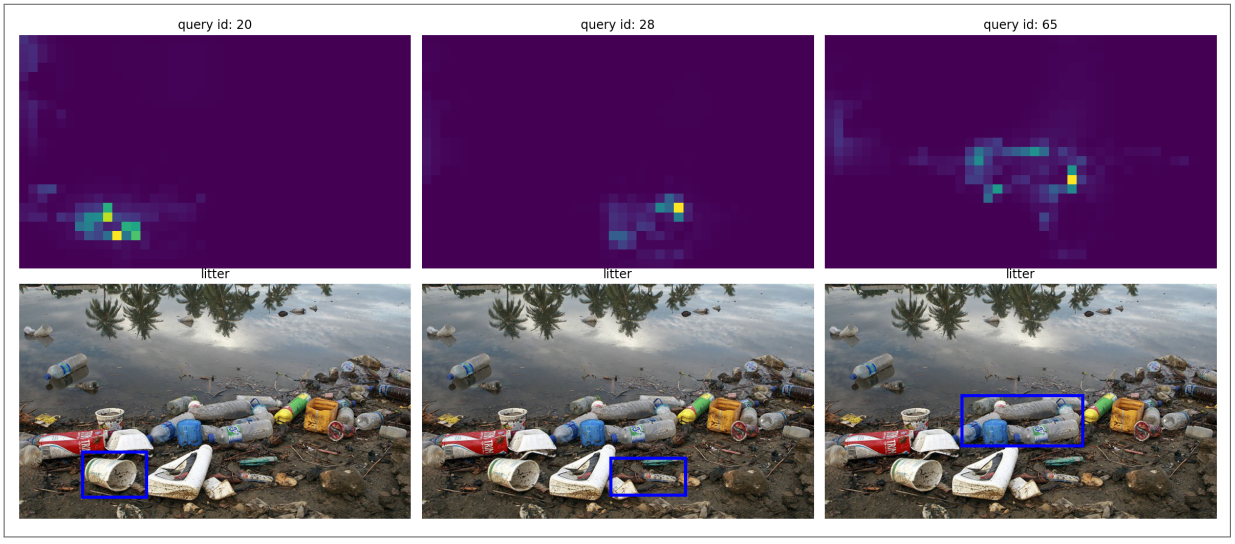
Figura 27 – A relação class_error é uma função das épocas. Os picos indicam que foram detectados objetos "no-object", ou seja, predições sem correspondência com as caixas de *ground truth*. Com a introdução do AUGMIX, os picos desapareceram.




**Fonte:** Elaborado pelo autor

de um objeto na mesma imagem é comum, e outros objetos podem ocultar certas partes do elemento. Por essa razão, a estratégia de recorte foi implementada para suportar esse cenário e melhorar a generalização, mas não produziu bons resultados. Além disso, durante o treinamento, verificou-se que os níveis de certeza para "no-object"eram mais altos do que os objetos que deveriam ser detectados.

Figura 28 – Excerto do codificador *self-attention* do modelo DETR, onde é possível identificar as instâncias individuais. Entretanto, é possível perceber que o codificador não tem a capacidade suficiente de abranger todos os elementos da imagem.




**Fonte:** Elaborado pelo autor

Figura 29 – A figura apresenta a progressão dos decodificadores do modelo Deformable, exibindo as etapas inicial e final e pontos de referência (pontos roxos). A comparação visual proporciona uma visão clara das transformações e avanços realizados durante o processo de decodificação.

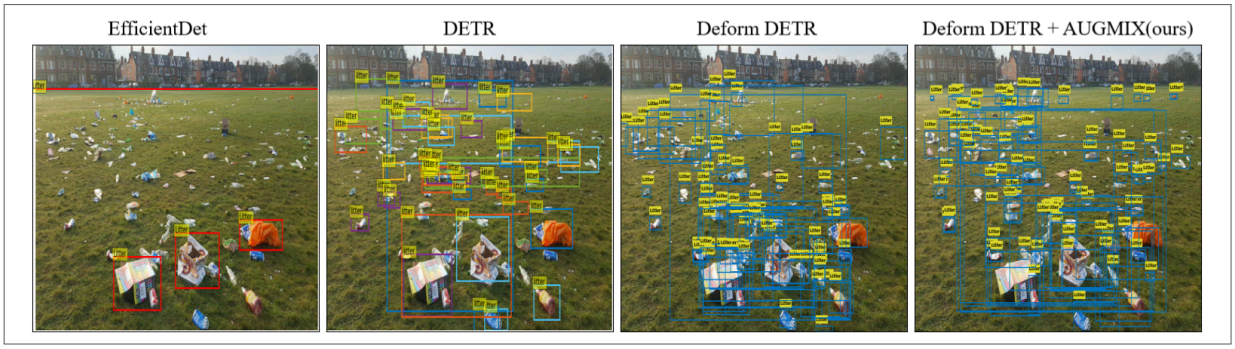

**Fonte:** Elaborado pelo autor

Assim como ocorre no DETR com suas seis etapas de refinamento das características, o Deformable também depende dessa cascata de decodificadores. Uma vez que as *queries* são inicializadas aleatoriamente no início de cada treinamento e, a cada iteração, são atualizadas, as *queries* têm uma forte relação com o desempenho, já que as predições finais são baseadas nelas por meio de uma cabeça de detecção. No entanto, o modelo Deformable incluído nesse conjunto possui um novo componente chamado ponto de referência baseado em *queries*, pois elas as predizem. A Figura 29 mostra as mudanças que ocorrem ao longo das iterações do decodificador, à medida que há uma iteração das etapas do decodificador, os pontos começam a se agrupar em torno dos objetos. No geral, os modelos tiveram um movimento de convergência da *Loss* dentro das 60 épocas, o que significa que o tamanho do bancos de imagens deve ter sido o grande influenciador no desempenho dos mesmos, haja vista que o mesmo modelo DETR atinge 300 épocas percorrendo as 118k imagens de treinamento para chegar na convergência.

Um ponto a ser contestado e que pode ser melhorado nesta pesquisa é a utilização de apenas um *dataset*, em tese, para a validação do método proposto. Essa aglutinação de vários bancos de imagem serviram para dar uma generalização para os modelos que, ao invés de trabalharem com poucas imagens e gerarem resultados fracos, conseguiram um melhor desempenho rodando em cima de uma quantidade maior de imagens. Durante a pesquisa foram

pesquisados vários *datasets* que fossem pequenos e tivessem uma formatação em COCO para que pudessem agilizar o processo, entretanto, foram encontrados poucos que necessitassem de pouca configuração e com o tamanho adequado. Outro ponto é a limitada quantidade de modelos para comparações, devido as restrições orçamentárias pela quantidade de tempo necessária desses modelos rodando na nuvem. Atualmente, o Google Colab, onde foram feitos os experimentos, define uma cota para utilização das GPUs o que limita a possibilidade de variações nos estudos e a inclusão de outros modelos que dependem de muito do uso dessas placas.

Figura 30 – O modelo Deformable proposto supera o seu antecessor e o EfficientDet por capturar de forma mais acurada os elementos no cenário. Isso pode distinguir objetos individuais e detectar mais deles quando comparado ao EfficientDet, o qual, frequentemente, identificou de forma errônea objetos como lixo. A melhora pela utilização da técnica é perceptível através da colocação das *bounding boxes* no entorno dos objetos detectados




**Fonte:** Elaborado pelo autor

Portanto, nos experimentos realizados, foi possível demonstrar que a arquitetura Deformable DETR poderia se beneficiar do ganho de robustez do AUGMIX, levando em consideração o desempenho exibido em uma ampla variedade de redes convolucionais, como Wide ResNet e ResNeXt-29. Utilizando os mesmos níveis de *threshold* para uma comparação justa, o modelo com AUGMIX ainda conseguiu se sobressair. É possível evidenciar essas diferenças de desempenho, conforme descrito nas curvas de mAP, e conforme pode ser visto na Figura 30, onde claramente o modelo que se utilizou da técnica implementada conseguiu abranger uma maior quantidade de objetos que os outros modelos não detectaram.

Recapitulando alguns dos apresentados anteriormente, podemos, de forma concisa, realizar a sumarização dos pontos mais relevantes da seguinte maneira:

- **Melhor Desempenho com AUGMIX:** O Deformable DETR$^{+}$, que utiliza 300 *queries* e a técnica AUGMIX, apresentou o melhor desempenho em todas as métricas. Isso indica que a utilização de técnicas de aumento de dados, como AUGMIX, pode ter

um impacto significativo na melhoria da performance do modelo, possivelmente por aumentar a diversidade dos dados de treinamento e ajudar o modelo a generalizar melhor para dados novos ou não vistos.

- **Impacto do Aumento de Queries:** A comparação entre o Deformable DETR e o Deformable DETR* (que tem 300 *queries*) sugere que o aumento do número de *queries* pode melhorar o desempenho do modelo. Isso sugere que a modelagem detalhada das interações entre os objetos na imagem pode ajudar a melhorar a performance do modelo. Essa mudança de *queries* afeta diretamente na quantidade do número de parâmetros. Em geral, os modelos com mais parâmetros (como DETR e Deformable DETR) tendem a ter um melhor desempenho, possivelmente porque eles são capazes de aprender representações mais complexas dos dados. No entanto, mais parâmetros também podem levar a um maior risco de overfitting e a requisitos de computação mais elevados.

- **Comparação entre Transformers e CNNs:** A tabela compara modelos baseados em Transformers (DETR e Deformable DETR) e um modelo baseado em redes neurais convolucionais (EfficientDet). Os modelos baseados em Transformers geralmente superaram o EfficientDet, mas este último ainda apresentou um desempenho notável, especialmente na detecção de objetos grandes. Isso reitera a relevância das redes neurais convolucionais na detecção de objetos e sugere que diferentes tipos de arquiteturas podem ter seus próprios pontos fortes em diferentes aspectos da detecção de objetos.

- **Número de Épocas de Treinamento:** O número de épocas de treinamento também pode influenciar o desempenho de um modelo. Nesta tabela, os modelos que foram treinados por mais épocas (63 épocas) geralmente tiveram um desempenho melhor do que o modelo treinado por menos épocas (30 épocas). Isso pode ser porque os modelos tiveram mais oportunidades de aprender a partir dos dados durante o treinamento. No entanto, treinar por um número excessivo de épocas pode levar a overfitting.

- **Desafio com Objetos Pequenos:** Todos os modelos apresentaram um desempenho significativamente mais baixo na detecção de objetos pequenos ($AP_s$) em comparação com objetos de tamanho médio e grande. Isso indica que a detecção de objetos pequenos é um desafio significativo nessa área, possivelmente devido à dificuldade de distinguir detalhes finos ou ao ruído em escalas menores.

# 5 CONCLUSÃO

Neste trabalho, foi proposta uma nova abordagem como forma de data augmentation para o Deformable DETR, substituindo parte da implementação original do modelo pela técnica de aumento de dados proposta. A abordagem proposta foi adaptada para a detecção de objetos, combinado a estratégia original de classificação com Transformers Deformable DETR. Nossa abordagem foi avaliada em um conjunto unificado de oito bases de dados e comparada com outros modelos baseados em Transformers e redes neurais convolucionais.

Foram realizados experimentos avaliando a variação no número de *queries* da abordagem proposta, que geralmente produzem bons resultados com seu aumento. No entanto, isso aumenta o tamanho dos parâmetros do modelo com base nos valores apresentados. Por enquanto, o uso de um Transformer continua limitado a sistemas que não possuem restrição de recursos computacionais. No entanto, os experimentos mostraram que a abordagem proposta foi capaz de superar os outros dois modelos da literatura, incluindo o próprio modelo base, sem a técnica proposta, no conjunto de dados *detect-waste*.

Embora o modelo proposto tenha exibido um bom desempenho, ainda foi possível verificar que algumas detecções errôneas podem ocorrer, ocasionadas por sombras ou formatos semelhantes aos vistos no conjunto de dados de treinamento. Outro problema, já relacionado a quantidade de GPUs disponíveis, pode ter influenciado no resultado final, tendo em conta que, o modelo DETR tem um desempenho melhor quanto mais houverem épocas de treinamento, o que é desaconselhado com *dataset* pequenos por um dos autores no repositório oficial do artigo, porém não foi possível averiguar de fato se o acréscimo de épocas teria um impacto positivo no contexto do presente trabalho, problema que poderia ser resolvido com uma maior quantidade de GPUs que reduziriam o tempo necessário para o treino do modelo e abririam um maior universo de possibilidades a serem experienciadas.

Durante nossos experimentos, observamos uma melhoria de pelo menos 0,4% na métrica $AP_s$ ao comparar o desempenho inicial com o segundo desempenho. Além disso, a diferença mais significativa ocorreu nas detecções de objetos pequenos, com uma diferença de 12,2% entre o primeiro e o último. Cabe ressaltar que, embora tenha menos parâmetros, o modelo EfficientDet obteve um bom desempenho quando se trata de objetos grandes, pois ele atingiu 55.9% na métrica correspondente aos objetos grandes $AP_l$, inclusive superando o DETR que possui mais parâmetros. Isso mostra uma boa eficiência no uso dos parâmetros e a indicação de

uso mais focado para objetos maiores. Entre os vários métodos testados, o Deformable obteve o melhor desempenho com e sem a implementação de nossa técnica de melhoria. Usando a versão aprimorada do Deformable com AUGMIX, conseguimos uma melhoria ainda mais significativa, aumentando +0,9% AP.

As contribuições deste trabalho baseiam-se no fato da incorporação de uma técnica de *augmentation* utilizada em tarefas de classificação e a adaptação da mesma para o cenário de detecção de objetos e a possibilidade de expansão dessa adaptação para qualquer outro modelo, embora isso não signifique que sempre haverão bons resultados, é necessário enfatizar o valor promissor desta abordagem, considerando-se o bom desempenho em todos as escalas das métricas de COCO, dado o desempenho do modelo Deformable é possível dizer que houve uma acentuação do desempenho para objetos pequenos. Uma das possiveís implicações desse trabalho é a utilização em ambientes reais com aplicações robóticas na indústria de alimentos, vídeo segurança e até rastreamento de pessoas através dos pés. Além disso, como contribuição significativa, o resultado final deste trabalho foi publicado no International Joint Conference on Neural Networks (IJCNN) de 2023, com classificação A1 no sistema Qualis da Capes o que faz ressaltar a importância deste estudo.

Para trabalhos futuros, pretendemos utilizar exemplos não realistas em cima de objetos reais. Outra abordagem é combinar DeepAugmentation com Augmix, que no Domínio de Generalização em ImageNet-R obteve melhor desempenho do que AUGMIX. Além disso, pretendemos utilizar outros bancos de imagens que permitam a avaliação e validação dos modelos em diferentes cenários e utilizar as modificações propostas em mais de um modelo possível. Propomos a utilização de ensemble de técnicas de *augmentation* que sejam feitas em tempo real e que permitam aos modelos alcançarem um melhor desempenho.

# REFERÊNCIAS

ASSAWIEL, N. *What to do when your training and testing data come from different distributions*. 2018. Disponível em: <https://www.freecodecamp.org/news/what-to-do-when-your-training-and-testing-data-come-from-different-distributions-d89674c6ecd8/>.

BUCKCHASH, H.; RAMAN, B. A robust object detector: application to detection of visual knives. In: IEEE. *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. [S.l.], 2017. p. 633–638.

CAO, D.; CHEN, Z.; GAO, L. An improved object detection algorithm based on multi-scaled and deformable convolutional neural networks. *Human-centric Computing and Information Sciences*, SpringerOpen, v. 10, n. 1, p. 1–22, 2020.

CARION, N.; MASSA, F.; SYNNAEVE, G.; USUNIER, N.; KIRILLOV, A.; ZAGORUYKO, S. End-to-end object detection with transformers. In: SPRINGER. *European conference on computer vision*. [S.l.], 2020. p. 213–229.

CUNHA, E.; MACÊDO, D.; ZANCHETTIN, C. Improving small object detection with detraug. In: IEEE. *2023 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2023. p. 1–8.

DAI, X.; CHEN, Y.; YANG, J.; ZHANG, P.; YUAN, L.; ZHANG, L. Dynamic detr: End-to-end object detection with dynamic attention. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2021. p. 2988–2997.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2009. p. 248–255.

FULTON, M. S.; HONG, J.; SATTAR, J. Trash-icra19: A bounding box labeled dataset of underwater trash. 2020.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2014. p. 580–587.

GIRSHICK, R. B. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. Disponível em: <http://arxiv.org/abs/1504.08083>.

HAAMER, K. *Letsdoitworld/wade-ai: An AI algorithm for detecting trash in geolocated images*. 2018. Disponível em: <https://github.com/letsdoitworld/wade-ai>.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778.

HENDRYCKS, D.; BASART, S.; MU, N.; KADAVATH, S.; WANG, F.; DORUNDO, E.; DESAI, R.; ZHU, T.; PARAJULI, S.; GUO, M. et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2021. p. 8340–8349.

HENDRYCKS, D.; DIETTERICH, T. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

HENDRYCKS, D.; MU, N.; CUBUK, E. D.; ZOPH, B.; GILMER, J.; LAKSHMINARAYANAN, B. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

HONG, J.; FULTON, M. S.; SATTAR, J. Trashcan 1.0 an instance-segmentation labeled dataset of trash observations. 2020.

JIN, S.; LU, X. Vision-based forest fire detection using machine learning. In: *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*. [S.l.: s.n.], 2019. p. 1–6.

JUNG, A. B.; WADA, K.; CRALL, J.; TANAKA, S.; GRAVING, J.; REINDERS, C.; YADAV, S.; BANERJEE, J.; VECSEI, G.; KRAFT, A.; RUI, Z.; BOROVEC, J.; VALLENTIN, C.; ZHYDENKO, S.; PFEIFFER, K.; COOK, B.; FERNáNDEZ, I.; RAINVILLE, F.-M. D.; WENG, C.-H.; AYALA-ACEVEDO, A.; MEUDEC, R.; LAPORTE, M. et al. *imgaug*. 2020. <https://github.com/aleju/imgaug>. Online; accessed 01-Feb-2020.

KARPOV, P.; GODIN, G.; TETKO, I. V. Transformer-cnn: Swiss knife for qsar modeling and interpretation. *Journal of Cheminformatics*, BioMed Central, v. 12, n. 1, p. 1–12, 2020.

KHAN, S.; NASEER, M.; HAYAT, M.; ZAMIR, S. W.; KHAN, F. S.; SHAH, M. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, ACM New York, NY, v. 54, n. 10s, p. 1–41, 2022.

KRAFT, M.; PIECHOCKI, M.; PTAK, B.; WALAS, K. Autonomous, onboard vision-based trash and litter detection in low altitude aerial images collected by an unmanned aerial vehicle. *Remote Sensing*, v. 13, n. 5, 2021. ISSN 2072-4292. Disponível em: <https://www.mdpi.com/2072-4292/13/5/965>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, AcM New York, NY, USA, v. 60, n. 6, p. 84–90, 2017.

KUKIL, K. *Intersection over union IOU in object detection segmentation*. 2023. Disponível em: <https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/>.

LEE, H.; GROSSE, R.; RANGANATH, R.; NG, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: *Proceedings of the 26th annual international conference on machine learning*. [S.l.: s.n.], 2009. p. 609–616.

LEVIN, G. Openframeworks, 2017. Disponível em: <https://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html>.

LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft coco: Common objects in context. In: SPRINGER. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. [S.l.], 2014. p. 740–755.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S. E.; FU, C.; BERG, A. C. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. Disponível em: <http://arxiv.org/abs/1512.02325>.

MAJCHROWSKA, S.; MIKOŁAJCZYK, A.; FERLIN, M.; KLAWIKOWSKA, Z.; PLANTYKOW, M. A.; KWASIGROCH, A.; MAJEK, K. Waste detection in pomerania: non-profit project for detecting waste in environment. *arXiv preprint arXiv:2105.06808*, 2021.

NASEER, M. M.; RANASINGHE, K.; KHAN, S. H.; HAYAT, M.; KHAN, F. S.; YANG, M.-H. Intriguing properties of vision transformers. *Advances in Neural Information Processing Systems*, v. 34, p. 23296–23308, 2021.

PHI, M. *Illustrated guide to transformers- step by step explanation*. Towards Data Science, 2020. Disponível em: <https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0>.

PROENÇA, P. F.; SIMOES, P. Taco: Trash annotations in context for litter detection. *arXiv preprint arXiv:2003.06975*, 2020.

REDMON, J.; DIVVALA, S. K.; GIRSHICK, R. B.; FARHADI, A. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. Disponível em: <http://arxiv.org/abs/1506.02640>.

REN, S.; HE, K.; GIRSHICK, R. B.; SUN, J. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. Disponível em: <http://arxiv.org/abs/1506.01497>.

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. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. [S.l.: s.n.], 2019. p. 658–666.

ROSEBROCK, A. Intersection over union (iou) for object detection. *Diambil kembali dari PYImageSearch https//www. pyimagesearch. com/2016/11/07/intersection-over-union-iou-for-object-detection*, 2016.

SAGAN, C. *Cosmos (A personal voyage): The Shores of the Cosmic Ocean*. 1980. Disponível em: <https://archive.org/details/CosmosAPersonalVoyage>.

SARKAR, A. *Are transformers better than CNN's at image recognition?* Towards Data Science, 2021. Disponível em: <https://towardsdatascience.com/are-transformers-better-than-cnns-at-image-recognition-ced60ccc7c8>.

SCHNEIDER, S.; RUSAK, E.; ECK, L.; BRINGMANN, O.; BRENDEL, W.; BETHGE, M. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, v. 33, p. 11539–11551, 2020.

SEREZHKIN, A. *Drinking waste classification*. 2020. Disponível em: <https://www.kaggle.com/datasets/arkadiyhacks/drinking-waste-classification>.

SERMANET, P.; EIGEN, D.; ZHANG, X.; MATHIEU, M.; FERGUS, R.; LECUN, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

SIOGKAS, G. K.; DERMATAS, E. S. Detection, tracking and classification of road signs in adverse conditions. In: IEEE. *MELECON 2006-2006 IEEE Mediterranean Electrotechnical Conference*. [S.l.], 2006. p. 537–540.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, v. 27, 2014.

TAORI, R.; DAVE, A.; SHANKAR, V.; CARLINI, N.; RECHT, B.; SCHMIDT, L. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, v. 33, p. 18583–18599, 2020.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017.

WANG, T.; CAI, Y.; LIANG, L.; YE, D. A multi-level approach to waste object segmentation. *Sensors*, MDPI, v. 20, n. 14, p. 3816, 2020.

WEI, J.; HE, J.; ZHOU, Y.; CHEN, K.; TANG, Z.; XIONG, Z. Enhanced object detection with deep convolutional neural networks for advanced driving assistance. *IEEE transactions on intelligent transportation systems*, IEEE, v. 21, n. 4, p. 1572–1583, 2019.

XIAO, Y.; TIAN, Z.; YU, J.; ZHANG, Y.; LIU, S.; DU, S.; LAN, X. A review of object detection based on deep learning. *Multimedia Tools and Applications*, Springer, v. 79, p. 23729–23791, 2020.

YANG, F.; FAN, H.; CHU, P.; BLASCH, E.; LING, H. Clustered object detection in aerial images. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 8311–8320.

YANG, M.; THUNG, G. Classification of trash for recyclability status. *CS229 project report*, Stanford University, v. 2016, n. 1, p. 3, 2016.

ZHU, X.; SU, W.; LU, L.; LI, B.; WANG, X.; DAI, J. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.