UNIVERSIDADE FEDERAL DE ALAGOAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

BRUNO GEORGEVICH FERREIRA

Novo Modelo de Rede Neural para Detecção de Objetos Aplicado à Inspeção Industrial

Maceió-AL

Abril de 2021

BRUNO GEORGEVICH FERREIRA

Novo Modelo de Rede Neural para Detecção de Objetos Aplicado à Inspeção Industrial

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal de Alagoas.

Orientador: Tiago Figueiredo Vieira

Maceió-AL

Abril de 2021

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Bibliotecário Responsável: Jone Sidney Alves de Oliveira CRB-4 - 1485

F383v Ferreira, Bruno Georgevich.
Novo Modelo de Rede Neural para Detecção de Objetos Aplicado à Inspeção Industrial / Bruno Georgevich Ferreia Neto. – 2021.
53 f. : il.

Orientador: Tiago Figueiredo Vieira.
Dissertação (mestrado em Informática) – Universidade Federal de Alagoas. Instituto de Computação. Programa de Pós-Graduação em Informática. Maceió, 2021.

Bibliografia: f. 52-54.

1. Rede Neural. 2. Monitoração Industrial. 3. Processamento de Imagens. 4. Software – Monitoramento.
I. Título.

CDU: 004.896

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Programa de Pós-Graduação em Informática – PPGI
Instituto de Computação/UFAL
Campus A. C. Simões BR 104-Norte Km 14 BL 12 Tabuleiro do Martins
Maceió/AL - Brasil CEP: 57.072-970 | Telefone: (082) 3214-1401

Folha de Aprovação

BRUNO GEORGEVICH FERREIRA

NOVO MODELO DE REDE NEURAL PARA DETECÇÃO DE OBJETOS APLICADO À INSPEÇÃO INDUSTRIAL

Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas e aprovada em 23 de abril de 2021.

Banca Examinadora:

Prof. Dr. TIAGO FIGUEIREDO VIEIRA
UFAL – Instituto de Computação
Orientador

Prof. Dr. THALES MIRANDA DE ALMEIDA VIEIRA
UFAL – Instituto de Computação
Examinador Interno

Prof. Dr. DOUGLAS CEDRIM OLIVEIRA
IF Goiano - Instituto Federal Goiano
Examinador Externo

Aos meus pais Reynaldo e Joyce, meus irmãos Amanda, Thiago e Tathi e a minha companheira Lívia, por me guiarem nos momentos mais difíceis.

AGRADECIMENTOS

Agradeço, primeiramente, aos meus pais Joyce e Reynaldo, por me ensinarem os princípios da vida e valores que levo comigo sempre. Por me mostrarem a vida sobre uma óptica verdadeira e por serem meus amigos em todas as ocasiões.

Aos meus amados irmãos que são e sempre serão meus amigos, por viverem e crescerem comigo e por nunca terem desistido de mim.

À minha companheira, Lívia Enders, por ser a minha melhor amiga, por sempre acreditar em mim, por me apoiar em todas as minhas empreitadas, por ter me ajudado a realizar todos os passos da minha trajetória.

Agradeço aos meus professores, coordenadores, diretores, especialistas, mestres e doutores que agregaram conhecimento e experiências à minha trajetória acadêmica.

Agradeço aos meus amigos do laboratório Charles Babbage e do EDGE, por sempre acreditarem em mim, por me ajudarem e me incentivarem a ser o meu melhor.

Agradeço ao pessoal do RAS, do DIACOM e do Ramo IEEE, por me incentivarem a ser melhor e tornarem essa trajetória mais leve.

Agradeço aos meus amigos de infância, Felipe, Leonardo e Luciano, por sempre estarem ao meu lado e serem tão importantes na minha vida e trajetória.

Por fim, agradeço em especial ao meu orientador, Tiago Figueiredo Vieira, por sempre me apoiar a encarar desafios e tentar vencê-los. Por toda dedicação, paciência, sabedoria e incentivo, sendo também um dos responsáveis por essa Dissertação se tornar possível.

Enfim, agradeço a todos que tornaram a realização desse sonho possível.

"O óbvio é aquilo que ninguém enxerga, até que alguém o expresse com simplicidade".

(Khalil Gibran)

RESUMO

Em muitas indústrias, a montagem de componentes específicos para serem inseridos em um recipiente de plástico é um procedimento manual. Cada kit deve ser composto por peças específicas seguindo uma receita pré-definida, que pode ser atualizada ao longo do tempo. Kits montados de forma inadequada causam retrabalho, reduzindo a qualidade e o tempo de produção. Aqui propomos melhorias em um modelo de detecção de objetos, capaz de realizar uma inspeção de qualidade, incrementando as funcionalidades do *Few-Shot Object Detection* (FSOD) baseado no modelo OS2D, previamente proposto na literatura. O modelo OS2D apresenta limitações ao tentar detectar objetos de *aspect ratio* que não se encaixam nas âncoras predeterminadas. Além disso, ele também tem um mecanismo de inferência que o restringe a apenas uma imagem de referência para cada classe, dificultando a detecção de objetos mais complexo, que se apresentam diferentes para cada ângulo. Dessa forma, foi proposto o modelo OS2D aprimorado (OS2D+) incorporando camadas de distorção e correção e modificando sua estratégia de inferência para facilitar a utilização de múltiplas imagens referências por componente. Para que seja possível avaliar os resultados da solução OS2D+, desenvolveu-se também uma outra solução baseada em processamento de imagens (PIMG), para que os resultados das duas sejam comparados. Foram propostas as camadas de distorção e correção, que compõem à solução OS2D+, permitindo que a mesma possa detectar objetos cujo *aspect ratio* não se enquadre em nenhuma âncora de detecção do modelo OS2D. O mecanismo de inferência do modelo OS2D também foi modificado, buscando viabilizar a inferência de múltiplas imagens de referência, para cada componente de um kit. Por fim, realizou-se uma comparação entre os desempenhos dos dois modelos, na tentativa de analisar se as modificações realizadas no OS2D+ incrementaram a performance do modelo OS2D em detectar objetos. Após a análise, a solução OS2D+ proposta se mostrou mais robusta que a PIMG, detectando menos falsos positivos e negativos, além de apresentar um tempo de inferência menor. Entretanto, a solução PIMG foi capaz de fornecer melhores estimações de *bounding boxes* (BB), devido ao seu processo de proposição de localizações. Ainda assim, a solução OS2D+ apresenta potencial para ter estimações equivalentes à PIMG, sendo necessário um ajuste fino em seus parâmetros. Também foi construída uma base de dados composta de 111 fotos, que descrevem cinco kits diferentes e seus respectivos componentes. Essa base de dados foi anotada e utilizada para mensurar os resultados das duas soluções propostas.

Palavras-chave: *Few-shot Learning*, *Few-shot Object Detection*, Inspeção de kits de Componentes, Processamento de Imagens.

ABSTRACT

In many industries, assembling specific components to be inserted into a plastic container is a manual procedure. Each kit must be comprised of specific parts following a pre-defined recipe, which can be updated throughout time. Kits assembled inadequately cause rework, reducing production quality and time. Here we propose improvements in an object detection model, capable of performing a quality inspection, increasing the features of Few-Shot Object Detection (FSOD) based on the OS2D model, previously proposed in the literature. The OS2D model has limitations when trying to detect objects of aspect ratio that do not fit the predetermined anchors. In addition, it also has an inference mechanism that restricts it to only one reference image for each class, making it difficult to detect more complex objects, which are different for each angle. Bearing in mind this, the improved OS2D model (OS2D+) was proposed, incorporating layers of distortion and correction and modifying its inference strategy to facilitate the use of multiple reference images per component. In order to be able to evaluate the results of the OS2D+ solution, a image processing based solution (PIMG) was also developed, so the results of both solutions can be compared. The distortion and correction layers of the OS2D+ solution allowing it to detect objects whose aspect ratio does not fit into any OS2D detection anchor. The inference mechanism of the OS2D model has also been modified, seeking to enable the inference of multiple reference images to a kit component. Finally, the performances of both models were compared, in an attempt to evaluate whether the modifications proposed in the OS2D+ model improved the ability of the OS2D model of detecting objects. Finally, the proposed OS2D+ solution proved to be more robust than PIMG, detecting fewer false positives and negatives, in addition to presenting a shorter inference time. However, the PIMG solution was able to provide better bounding boxes (BB) estimations due to its process of proposing locations. Despite this, the OS2D+ solution has the potential to have equivalent estimations, requiring a fine adjustment in its parameters. A database composed of 111 photos was also built, describing five different kits and their respective components. This database was annotated and used to measure the results of the two proposed solutions.

Keywords: Components Kits Inspection, Image Processing, Few-shot Learning, Few-shot Object Detection.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
CV	*Computer Vision*
ML	*Machine Learning*
DL	*Deep Learning*
2SD	*Two-stage Detector*
1SD	*One-stage Detector*
FSL	*Few-shot Learning*
FSOD	*Few-shot Object Detection*
SSD	*Single-shot Multibox Detector*
CNN	*Convolutional Neural Network*
VGG	*Visual Geometry Group*
ILSVRC	*ImageNet Large-Scale Visual Recognition Challenge*
ResNet	*Residual Network*
BB	*Bounding Box*
R-CNN	*Regions with CNN features*
SVM	*Support Vector Machines*
RoI	*Region of Interest*
RPN	*Region Proposal Network*
NMS	*Non-maximum Suppression*
1SL	*One-shot Learning*
0SL	*Zero-shot Learning*
SNN	*Siamese Neural Network*
TPS	*Thin plate Spline*
OS2D	*One-stage One-shot Object Detection*

MR	Meta-Representação
PR	Coeficiente de Correlação Pearson
MPSR	*Multi-scale Positive Sample Refinement*
CG-BERT	*Conditional Text Generation with BERT*
GFSID	*Generalized Few-Shot Intent Detection*
IoU	*Intersection over Union*
PIMG	*Solução de Processamento de Imagem*
OS2D+	*OS2D Aprimorado*

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

A sociedade, atualmente, pode ser visualizada em diversas perspectivas e contextos. Dentre esses, muitos utilizam a computação como instrumento facilitador de atividades, seja um médico com técnicas de processamento de imagem para realçar tomografias computadorizadas ou um estudante com plataformas de cursos online ou ensino à distância para aprender.

A computação evolui de forma a tentar resolver ou facilitar a resolução dos problemas apresentados pela sociedade e indústria. Com as revoluções industriais, tornou-se necessária a demanda por computadores inteligentes capazes de tomar decisões e reconhecer padrões complexos, da mesma forma que os seres humanos fazem.

Com essa problemática, a inteligência artificial (IA) surge como uma tentativa da comunidade científica de mimetizar o funcionamento da inteligência humana em um computador. Graças a isso, hoje é possível consumir produtos com IA, como o buscador do Google, assistentes pessoais, *softwares* que traçam rotas de mapas e que buscam indicações de produtos, a partir de um histórico de escolhas.

Segundo Rich e Knight (1991), IA é a tentativa de fazer com que os computadores desempenhem ações que os seres humanos atualmente são melhores. Já para Winston (1992), a IA é a capacidade do computador perceber, raciocinar e agir perante a isso. Dentre as diversas aplicações da IA, uma que se faz muito presente nos produtos hoje em dia é a *computer vision* (CV, em português "visão computacional").

De acordo com Sonka, Hlavac e Boyle (2014), a visão computacional busca duplicar o efeito da visão humana ao perceber e entender eletronicamente uma imagem. Deste modo, é possível entender que visão computacional não se resume a uma câmera, pois a mesma além de capturar a imagem, busca interpretá-la e retirar informações disso. Para Shapiro e Stockman (2001), tende a infinito o número de possíveis aplicações de visão computacional para resolver problemas da sociedade e indústria.

Um segmento importante da IA, que se popularizou ao longo do tempo, é o *machine learning* (ML, em português "aprendizagem de máquina"). Para Mitchell (1997), o ML é o campo da IA que se preocupa em criar programas capazes de tornarem-se melhores com a experiência, automaticamente. Desta forma, com o ML é possível construir programas computacionais que, a partir de dados, são capazes de aprender ao longo do tempo.

Juntamente com a evolução da visão computacional, o algoritmo de ML que se tornou bastante popular foram as *neural networks* (NN, em português "redes neurais"). As NNs são subdivididas em camadas de neurônios, de modo que o número de camadas (profundidade) e de neurônios por camadas (largura) são parâmetros importantes na sintonização de uma rede

neural. A medida em que as NNs foram sendo aperfeiçoadas, as mesmas se tornaram cada vez mais profundas, surgindo dessa forma um subconjunto do ML denominado de *deep learning* (DL, em português "aprendizagem profunda").

Para Chollet (2017), o DL é um subcampo de ML que fornece uma nova visão sobre representações de aprendizado, de forma que as camadas mais profundas da NN apresentam o conhecimento mais complexo e as mais próximas da sua entrada, o conhecimento de estruturas mais básicas. Segundo Goodfellow, Bengio e Courville (2016), o DL permite que uma representação mais complexa fosse apresentada como uma combinação de elementos mais básicos.

Com o surgimento do DL, diversas arquiteturas de NNs foram sendo propostas para atacar diferentes problemas, como classificação de imagens, linguagem natural, sistema de sugestões, detecção de objetos, dentre outros. As redes neurais são fortemente utilizadas na visão computacional, como classificadores de imagens, detectores de objetos, geradores de imagens, transferência de estilo, entre outras aplicações.

As principais aplicações de redes neurais na visão computacional são a classificação de imagens e a detecção de objetos. Conceitualmente, elas apresentam uma diferença tênue, onde a primeira atribui uma classe a uma imagem e a segunda uma classe a cada objeto da imagem. Entende-se por objetos, elementos da imagem que se fazem importantes de serem detectados pela NN. Desta forma, uma das etapas da detecção de objetos é a classificação de cada objeto detectado.

Estruturalmente, a classificação de imagens e a detecção de objetos apresentam arquiteturas semelhantes, entretanto, no que diz respeito à base de dados, da construção à utilização, as duas técnicas são diferentes. Ambas podem utilizar da NN em uma abordagem supervisionada. Quando a rede neural é utilizada em uma abordagem supervisionada, ela tem como necessidade ser alimentada por uma base de dados anotada[1].

Apesar da NN ser um algoritmo muito eficaz na resolução de diversos problemas da sociedade, a mesma apresenta como fator complicante a necessidade de ser alimentada com um grande volume de dados anotados. A depender do problema, caso o mesmo seja algo muito específico, torna-se difícil conseguir tais dados. Em contrapartida, algumas vezes se tem acesso a um grande volume de dados não anotados, sendo necessário que a anotação dos dados seja feita pelo desenvolvedor ou pelo time responsável.

O processo de anotação dos dados se faz diferente para cada problema. Por exemplo, no caso da classificação de imagens, cada imagem presente na base de dados deverá ser atrelada a uma classe[2]. Desta forma, a complexidade da anotação dos dados depende diretamente do tamanho da base de dados, de modo que quanto maior a base, maior será o trabalho para anotá-la.

[1] Uma base de dados definida como anotada precisa apresentar suas informações previamente constatadas.

[2] A classe de uma imagem representa o que está descrito na mesma. Ex.: um animal, uma ação, um objeto, dentre outros

No que diz respeito à detecção de objetos, o processo de anotação se torna muito mais complexo, tendo em vista que em uma imagem podem existir diversos objetos da mesma classe ou de classes distintas, sendo necessário definir a posição, dimensão e o rótulo de cada um. Para definir a posição e a dimensão do objeto, geralmente desenha-se um retângulo envolvendo-o. Esse processo tem que ser feito para todos os objetos de todas as imagens da base de dados.

Uma arquitetura de rede neural para detecção de objetos geralmente apresenta dois componentes importantes, embora eles possam atuar de forma conjunta. O primeiro componente é o classificador, responsável por filtrar e classificar os objetos. O segundo é o detector, o qual tem como função propor a localização de cada objeto encontrado na imagem. A ordem de utilização desses componentes dependerá da arquitetura utilizada.

Até o momento, a maioria das arquiteturas de redes neurais utilizadas na detecção de objetos podem ser agrupadas em dois subgrupos: o *two-stage detector* (2SD, em português "detector de dois estágios") e o *single-stage detector* (1SD, em português "detector de um estágio"). No 2SD o estágio de detecção é executado em um momento separado do estágio de classificação. Já na 1SD, a localização e classificação são executadas simultaneamente, de modo que o detector e o classificador trabalham de forma conjunta.

Tendo em vista a dificuldade encontrada no processo de anotação das bases de dados, segmentos do DL como *transfer learning* e *few-shot learning* (FSL) começaram a se popularizar. No FSL existem modelos baseados em *metric-learning* e modelos baseados e *meta-learning*. O *metric-learning* consiste numa estratégia onde o modelo busca aprender a similaridade entre as suas entradas, na tentativa de atestar se as mesmas são iguais ou não. Já o *meta-learning* consiste no aprendizado de meta-características, na tentativa de extrair as *features* mais relevantes da entrada, buscando obter um retreinamento facilitado.

Com a popularização do FSL, alguns autores buscaram aplicá-lo na tarefa de detecção de objetos, a qual foi intitulada de *Few-Shot Object Detection* (FSOD, em português "detecção de objetos com poucos exemplos"). Autores como Michaelis, Bethge e Ecker (2018), Kang et al. (2019) e Fan et al. (2020) desenvolveram modelos FSOD utilizando de arquiteturas 2SD, em sua grande maioria inspirados no modelo proposto por Ren et al. (2015), a Faster R-CNN. Entretanto, autores como Fu et al. (2019), Bennequin (2019), Hsieh et al. (2019) e Osokin, Sumin e Lomakin (2020) propuseram modelos FSOD baseados em arquiteturas 1SD, inspirados em sua maioria pelo modelo proposto por Liu et al. (2016), a *Single Shot MultiBox Detector* (SSD).

1.2 Problemática

Alguns processos realizados internamente em uma fábrica apresentam um grande risco inerente, seja ele financeiro ou de algum desastre ambiental. Na tentativa de minimizar isso, diversos subprocessos são inspecionados, em busca de alguma falha. Tendo em vista isso, indústrias que manufaturam peças ou kits[3] acabam por ter que conferir diversos lotes de produtos fabricados, visando minimizar a expedição de um item defeituoso. Porém, essas verificações, em sua maioria, são realizadas por seres humanos, o que torna o processos lento e custoso.

Buscando reduzir os erros injetados nas linhas de montagem de kits em fábricas, *softwares* de visão computacional se tornam bastante atrativos para as empresas, tendo em vista que o custo de implantação e a taxa de erros são bem menores que a abordagem com supervisão humana. Entretanto, soluções tradicionais com processamento de imagens apresentam limitações referentes à variação de iluminação, oclusão de itens, desgastes materiais, entre outros. Já soluções dotadas de modelos de redes neurais convencionais, próprias para detecção de objetos, apresentam-se mais robustas que as soluções de processamento de imagem, porém não são flexíveis a mudanças de componentes dos kits, sendo necessário um retreinamento a cada adição ou remoção de componentes.

Tendo em vista as dificuldades encontradas nas soluções de processamento de imagem e de redes neurais para detecção de objetos, percebeu-se um grande potencial nas soluções de FSOD. O FSOD possibilita robustez a ruídos, inerente da solução com redes neurais, e a flexibilidade da solução com processamento de imagem, no que diz respeito a adição e remoção de novos componentes na linha de montagem.

1.3 Objetivos

1.3.1 Objetivo Geral

Este trabalho apresenta como objetivo geral o desenvolvimento de uma solução capaz de inspecionar kits de componentes. Esta solução deverá ser capaz de detectar, corretamente, todos os componentes presentes no kit. Para isso, serão construídas duas soluções, sendo a primeira baseada em técnicas de processamento de imagens (PIMG) e a segunda uma adaptação do modelo OS2D, proposto por Osokin, Sumin e Lomakin (2020), chamada de OS2D+. Também serão analisadas as duas soluções, sobre um conjunto de métricas, na tentativa de observar qual das duas apresenta-se mais capaz de solucionar a problemática.

1.3.2 Objetivos Específicos

Primeiramente, será desenvolvida uma base de dados, que conterá os kits e seus respectivos componentes. Essa base de dados deverá ser anotada, fornecendo dessa forma uma

[3] Um kit é definido como um conjunto de peças e ferramentas que serão utilizados para um fim específico. Ex.: Kit de parafusos para afixação de uma antena.

groundtruth para posterior comparação das duas soluções. Em seguida, será desenvolvida uma solução baseada em processamento de imagens (PIMG), a qual deverá cumprir a tarefa de detectar os componentes dispostos no kit e permitir a adição e remoção de novos componentes.

Posteriormente, será proposto o modelo OS2D+, o qual é uma adaptação do modelo OS2D desenvolvido por Osokin, Sumin e Lomakin (2020), de forma que ele será capaz de realizar a tarefa de detectar os componentes dispostos no kit e, da mesma forma que na solução PIMG, permitir a adição e remoção de novos componentes, sem a necessidade de retreinamento.

Ao longo do desenvolvimento do trabalho, foi percebida uma limitação no modelo OS2D proposto por Osokin, Sumin e Lomakin (2020) relacionada ao número de âncoras de detecção do mesmo. Devido a isso, são propostas duas camadas denominadas de distorção e correção, sendo a primeira de pré-processamento e a última de pós-processamento, que permitem que o modelo seja capaz de detectar objetos com *aspect ratios* diferentes de suas âncoras de treinamento. Também será ajustado o mecanismo de inferência do modelo OS2D, possibilitando o uso de múltiplas imagens de referência para cada componente de cada kit. Por fim, os desempenhos dos dois modelos serão comparados na tentativa de analisar se as modificações realizadas no OS2D+ incrementaram a performance do modelo OS2D em detectar objetos.

Para analisar as duas soluções desenvolvidas, será estabelecido um conjunto de métricas, que permitirá avaliar quantitativamente o desempenho individual de cada solução. Por fim, serão analisados os resultados obtidos pelas soluções em cada uma das métricas, a fim de apresentar qual das duas soluções se faz mais capaz de solucionar o problema de inspeção de kits de componentes.

2 REFERENCIAL TEÓRICO

Neste capítulo será descrito o embasamento teórico do presente trabalho. Inicialmente será descrita a parte de redes neurais convolucionais, no que tange a parte de extratores de características. Logo em seguida será mostrada a parte de arquiteturas de detecção de objetos, explanando de forma superficial o funcionamento delas. Também será apresentado um pouco da teoria de *few-shot learning*, mostrando alguns conceitos importantes da área e serão detalhados os modelos propostos por Rocco, Arandjelovic e Sivic (2017) e Osokin, Sumin e Lomakin (2020), os quais foram fundamentais para o desenvolvimento do presente trabalho. Por fim, serão mostrados alguns trabalhos de outros autores, os quais utilizaram FSL para resolver diversos problemas da área de detecção e interpretação de sons, visão computacional para classificação e detecção de objetos e interpretação de textos de linguagem natural.

2.1 Convolutional Neural Network (CNN)

Segundo Goodfellow, Bengio e Courville (2016), as redes convolucionais são redes neurais que realizam a convolução ao invés da multiplicação geral da matriz em pelo menos uma de suas camadas. Segundo o autor, as CNNs são especializadas em processar dados do tipo *grid*, por exemplo imagens. As mesmas podem ser vistas como camadas de pré-processamento, tendo em vista que elas irão filtrar a entrada para as camadas densas.

As CNNs apresentam como hiperparâmetro o número de filtros e a dimensão dos mesmos. Dessa forma, a CNN irá estimar, a partir do treinamento, os valores que melhor compõem os filtros convolucionais. A equação que calcula o número de parâmetros treináveis de uma camada convolucional pode ser vista na Equação 2.1, onde P representa o número de parâmetros treináveis da camada, A descreve a área dos filtros convolucionais da camada atual e N_{atual} e $N_{anterior}$ o número de filtros da camada atual e anterior, respectivamente.

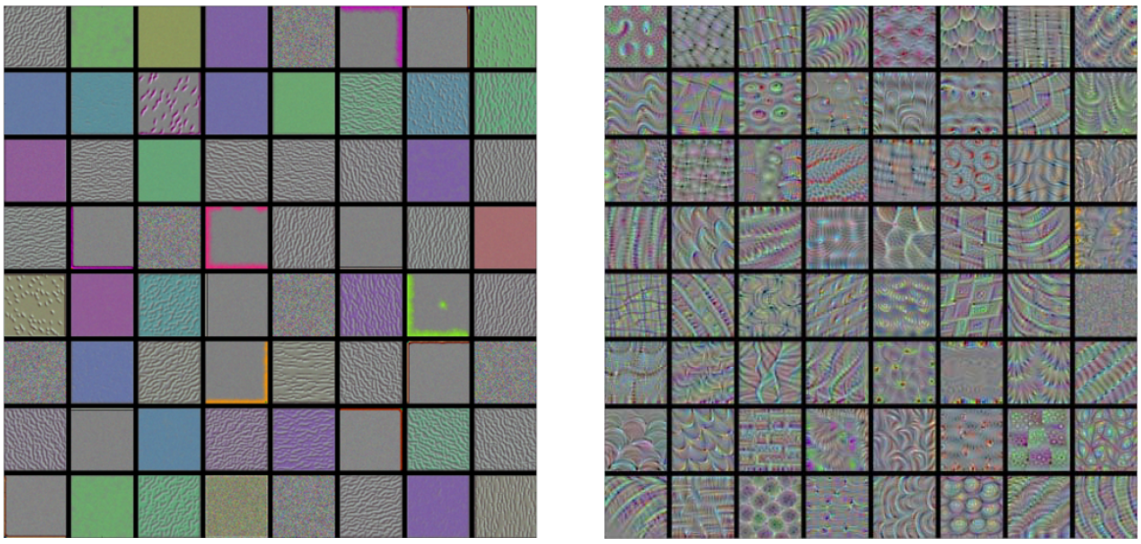
$$P = (A * N_{atual} * N_{anterior}) + N_{atual} \tag{2.1}$$

Fonte: Elaborado pelo autor.

Assim como as NNs convencionais, a CNN permite que representações mais complexas sejam combinações de elementos mais simples. Ao empilhar camadas convolucionais, desenvolveu-se o conceito de redes convolucionais. Tendo em vista isso, percebeu-se que os filtros presentes nas camadas convolucionais mais próximas da entrada da rede neural apresentavam padrões mais primitivos e simplificados (Figura 1a), em contrapartida os padrões encontrados nas camadas mais próximas do final da CNN eram mais complexos (Figura 1b), fruto da combinação dos padrões mais simples que os precediam. Tendo em vista que as camadas mais posteriores apresentam padrões mais complexos, os mesmos se fazem mais raros nas imagens, fazendo com

que o resultado das camadas finais da rede sejam mais esparsos, quando comparado com os das camadas iniciais, que apresentam filtros com padrões mais simples e, por isso, mais facilmente achados nas imagens.

Figura 1 – Exemplos de filtros convolucionais primitivos e complexos.



(a) Padrões mais primitivos (b) Padrões mais complexos

Fonte: Chollet (2017, p. 170 e 172).

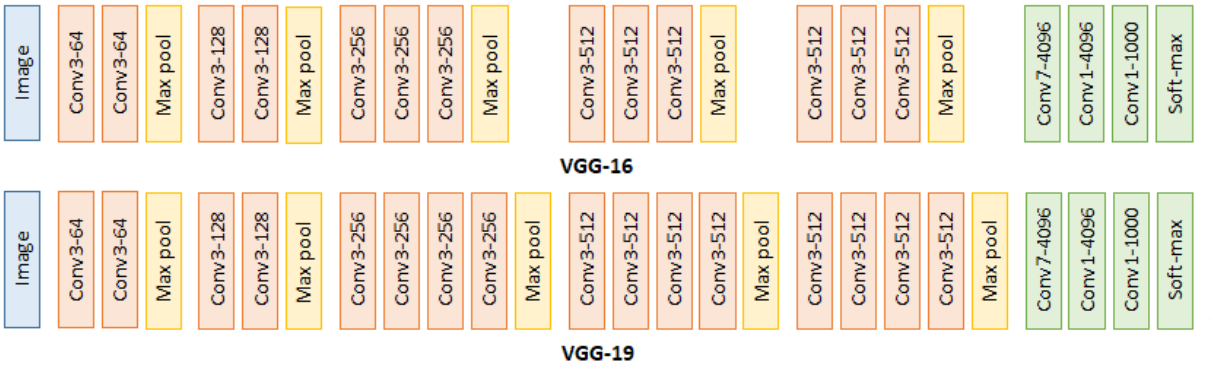
2.1.1 Modelos Convolucionais

Com o aprimoramento da teoria de redes neurais e o avanço da capacidade computacional e dos dados, fez-se possível o desenvolvimento de modelos distintos de CNNs. Alguns modelos buscavam ser mais performáticos no tempo de treinamento e execução, outros diminuir o *overfitting* ou até mesmo alcançarem novos recordes de acurácia nas bases de dados conhecidas. Desta forma, serão mostrados alguns modelos que apresentam características distintas e que, posteriormente, servirão como possíveis extratores de características dos modelos de detecção de objetos.

2.1.1.1 VGG16 e VGG19

Propostos por Simonyan e Zisserman (2014), que pertenciam ao grupo *Visual Geometry Group* (VGG), os modelos VGG16 e VGG19 introduziram a utilização de filtros convolucionais pequenos (3x3), o que permitiu a construção de redes neurais mais profundas, com 16 e até 19 camadas. Por isso, o nome VGG16 e VGG19, sendo a primeira com 16 camadas e a última com 19. Esses modelos foram submetidos à competição *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC) de 2014, onde conquistaram o primeiro e o segundo lugar nas faixas de localização e classificação, respectivamente. As topologias dos dois modelos citados anteriormente podem ser vistas na Figura 2.

Figura 2 – Topologia dos modelos VGG16 e VGG19.

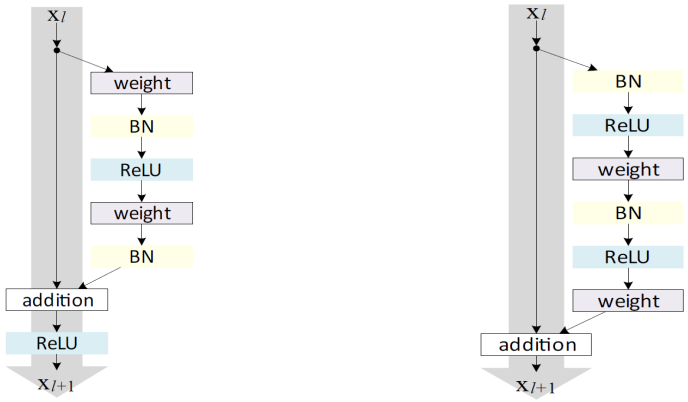


Fonte: Adaptada de Tsang (2018).

2.1.1.2 *Residual Network* (ResNet)

A *Residual Network* (ResNet) proposta por He et al. (2016a) permitiu a criação de redes neurais com mais de 1000 camadas, o que anteriormente não seria possível devido ao *overfitting* causado pelo grande número de camadas. Isso se deu pela criação do bloco Residual, ilustrado na Figura 3a, que busca apresentar-se diferente de um bloco neural convencional. O bloco Residual apresenta duas ramificações por onde o sinal da entrada passa, onde a primeira apresenta um modo de operação semelhante com um bloco de rede neural convencional e a segunda permite a passagem direta do sinal da entrada para a saída. Devido a isso, os autores denominam a segunda ramificação como identidade.

Figura 3 – Módulos residuais das ResNet V1 e V2.



(a) Bloco V1 (b) Bloco V2

Fonte: He et al. (2016b, p. 2).

Após o sinal da entrada passar pelas duas ramificações, suas saídas são somadas e o sinal gerado passa por uma camada de ativação e em seguida para o próximo bloco. A arquitetura de um bloco residual permite que o sinal de entrada seja refletido na saída sem ser alterado,

de modo que se uma rede neural tiver muitas camadas, os blocos desnecessários não injetarão ruídos no sinal, tendo em vista que os pesos do componente residual serão próximos a zero. Em contrapartida, a arquitetura de uma rede neural convencional iria alterar o valor do sinal da entrada multiplicando os pesos e somando os vieses de blocos desnecessários, o que injetaria ruídos no sinal.

A formalização matemática que descreve a relação da saída de uma camada com o sinal de entrada da mesma, nas arquiteturas de redes neurais não residual e residual, a formalização pode ser vistas nas Equações 2.2 e 2.3, respectivamente. O elemento x_{l+1} representa a saída e o x_l a entrada da camada atual, o W_l é a matriz de pesos e vieses da ramificação não residual, f é a função de ativação e F é a função que define a computação da ramificação não residual.

$$x_{l+1} = f(W_l \cdot x) \tag{2.2}$$

$$x_{l+1} = f(x_l + F(x_l, \{W_l\})) \tag{2.3}$$

Fonte: He et al. (2016b, p. 1).

O bloco residual proposto por He et al. (2016a) permite que o sinal da entrada seja refletido na saída, passando apenas por uma função de ativação. Essa função de ativação, acaba por distorcer levemente o sinal de entrada, não permitindo que o mesmo passe de forma inalterada pelo bloco. Devido a isso, He et al. (2016b) propôs uma melhoria no módulo original da ResNet, como pode ser visto na Figura 3b. Agora a ativação é realizada apenas na ramificação não residual, como pode ser visto na Equação 2.4, permitindo que a entrada seja propagada para a saída sem nenhuma modificação. Isso se mostrou interessante em redes neurais muito profundas (> 1000 camadas), entretanto o resultado para as redes mais superficiais (aproximadamente 100 camadas) não apresentou grandes mudanças.

$$x_{l+1} = x_l + F(f(x_l), \{W_l\}) \tag{2.4}$$

Fonte: He et al. (2016b, p. 3).

2.2 *Object Detection*

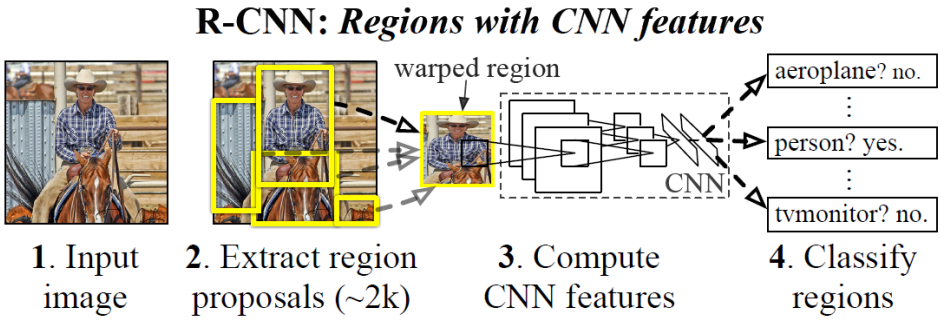
Em geral, os modelos de detecção de objetos existentes podem ser classificados em dois subgrupos, os *two-stage detectors* (2SD) e os *single-stage detectors* (1SD). Os detectores de dois estágios, geralmente, apresentam um estágio para detecção de possíveis *bounding boxes* (BB) e um estágio para classificação dos mesmos. Já os detectores de um estágio, em sua maioria, realizam a atividade de localização e classificação simultaneamente. Exemplos de modelos conhecidos para os dois grupos são *Faster R-CNN*, para o 2SD, e *Single Shot Multibox Detector* (SSD), para o 1SD.

2.2.1 *Two-Stage Detectors* (2SD)

2.2.1.1 R-CNN

O modelo *Regions with CNN features* (R-CNN) proposto por Girshick et al. (2014), aplica o algoritmo *Selective Search* na imagem, extraindo possíveis objetos de interesse. Após isso, esses objetos são inseridos em uma CNN, que irá filtrá-los, tendo sua saída conectada a um classificador *Support Vector Machine* (SVM) e um regressor, que irá estimar as coordenadas x, y e a largura e altura dos BBs dos objetos. É possível perceber que o R-CNN apresenta um *pipeline* de operação complexo, sendo isso um dos fatores que o tornam lento e difícil de ser treinado. Os autores relataram que uma entrada demora cerca de 47 segundos para ser processada, ou seja, retornar os BBs e classes encontrados na imagem. Um esquema do *pipeline* da R-CNN pode ser visto na Figura 4.

Figura 4 – Esquema do *pipeline* de execução da R-CNN.



Fonte: Girshick et al. (2014, p. 1).

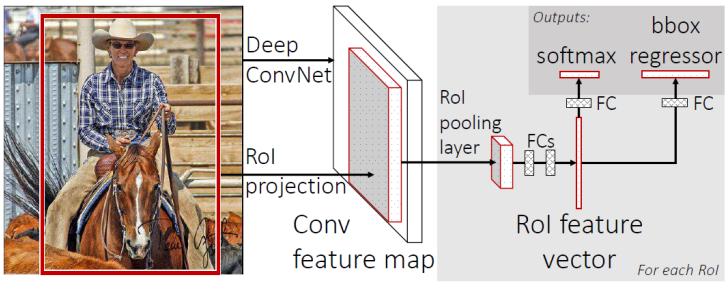
O método de treinamento de uma R-CNN se faz complexo, pois o *Selective Search*, a CNN e o SVM são sintonizados separadamente, o que dificulta o processo de otimização do modelo. Um problema relatado pelos autores consiste no armazenamento das informações entre as etapas do *pipeline*, resultando em centenas de gigabytes de espaço ocupado com etapas intermediárias. Devido ao fato do *Selective Search* fornecer 2000 BBs de possíveis objetos, muitas das detecções realizadas pela R-CNN são descartadas, pois o mesmo não sofre influência da rotina de treinamento dos outros componentes da arquitetura. Desta forma, muitas das detecções fornecidas pelo *Selective Search* são de trechos irrelevantes.

2.2.1.2 Fast R-CNN

Devido aos problemas encontrados na R-CNN, Girshick (2015) propôs uma nova versão da mesma intitulada de Fast R-CNN. Essa nova versão visa reduzir a complexidade do *pipeline* de treinamento e inferência, tornando-os mais ágeis. O novo fluxo consiste na inserção da imagem de entrada e de um conjunto de regiões de interesse diretamente na CNN. As regiões de interesse inseridas na CNN, são propostas pelo algoritmo *Selective Search*, assim como na

R-CNN. Em seguida, essas regiões passarão pela camada *Region of Interest* (RoI) *pooling layer*, que irá extrair um conjunto fixo de vetores de *features*, que posteriormente serão inseridos em uma camada *fully connected* (FC). A FC irá estimar as coordenadas do BB e a classe do objeto contido na região de interesse. O *pipeline* da Fast R-CNN pode ser visto na Figura 5.

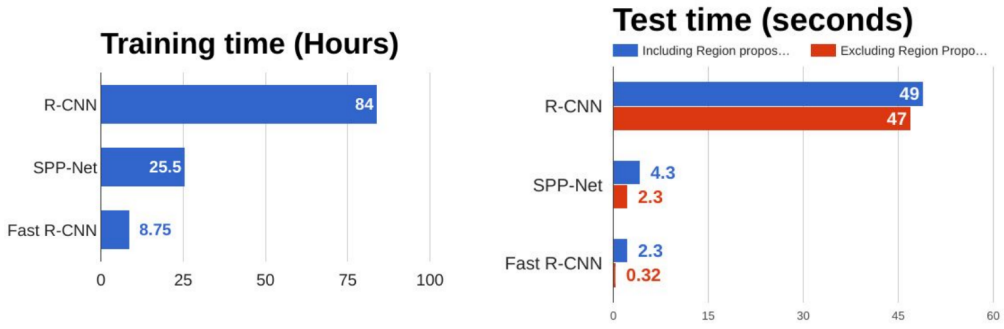
Figura 5 – Esquema do *pipeline* de execução da Fast R-CNN.



Fonte: Girshick (2015, p. 2).

A modificação do *pipeline* de execução da Fast R-CNN permitiu uma melhoria na sua acurácia e velocidade de inferência. Segundo os autores, os ajustes realizados no modo de operação do modelo tornou-o, aproximadamente, 147 vezes mais rápido que a R-CNN, em tempo de inferência. Essa melhora, fez com que o modelo que antes levava 47 segundos para realizar a inferência de uma imagem, agora precise de apenas 0,32 segundo para cumprir a mesma tarefa. Apesar disso, a Fast R-CNN manteve o mesmo propositor de BBs que a R-CNN, sendo necessários 2 segundos para que o *Selective Search* possa propor as regiões de interesse. Ao comparar o tempo de treinamento, a Fast R-CNN apresenta-se 10 vezes mais rápida que a R-CNN, como pode ser nos gráficos contidos na Figura 6.

Figura 6 – Comparação da Fast R-CNN, R-CNN e SPP-Net em tempo de execução e treino.



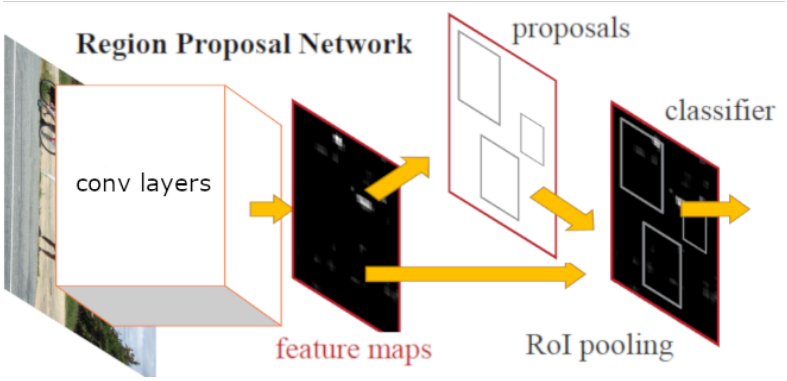
Fonte: CS231n (2017, p. 79).

2.2.1.3 Faster R-CNN

Na tentativa de eliminar os 2 segundos necessários para o *Selective Search* localizar as BBs, Ren et al. (2015) propôs a Faster R-CNN, que é uma versão aprimorada da Fast

R-CNN. Nesse novo modelo, os BBs são propostos por uma rede neural chamada de *Region Proposal Network* (RPN), a qual irá aprender como detectar as classes e realizar a regressão das coordenadas das localizações. Desta forma, o novo *pipeline* proposto na Faster R-CNN consiste na inserção da imagem de entrada em uma CNN, que irá gerar os *feature maps*. Em seguida os *feature maps* passarão pela RPN, a qual irá propor os BBs. Por fim, os BBs e os *feature maps* serão inseridos na ROI *pooling layer*, que irá classificá-los. O esquema do *pipeline* de execução da Faster R-CNN pode ser visto na Figura 7.

Figura 7 – Esquema do *pipeline* de execução da Faster R-CNN.



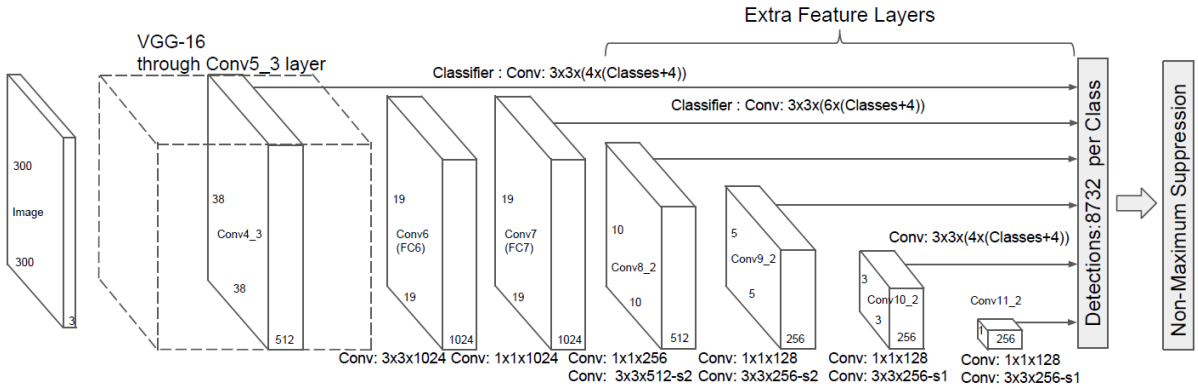
Fonte: Adaptado de Ren et al. (2015, p. 3).

2.2.2 Single-Stage Detectors

2.2.2.1 SSD

Um modelo de detecção de objetos 1SD é a *Single Shot MultiBox Detector* (SSD), proposta por Liu et al. (2016). Ela apresenta dois componentes principais: o *backbone* e a *SSD head*. Geralmente, o *backbone* é uma CNN, que irá filtrar a imagem de entrada, preservando o contexto dos objetos e a distribuição espacial dos mesmos, além de reduzir a dimensão da entrada da rede. Já a *SSD head* são camadas convolucionais adicionadas no fim do *backbone*, que irão estimar as coordenadas e as classes dos BBs dos objetos encontrados na imagem. A arquitetura da SSD pode ser vista na Figura 8.

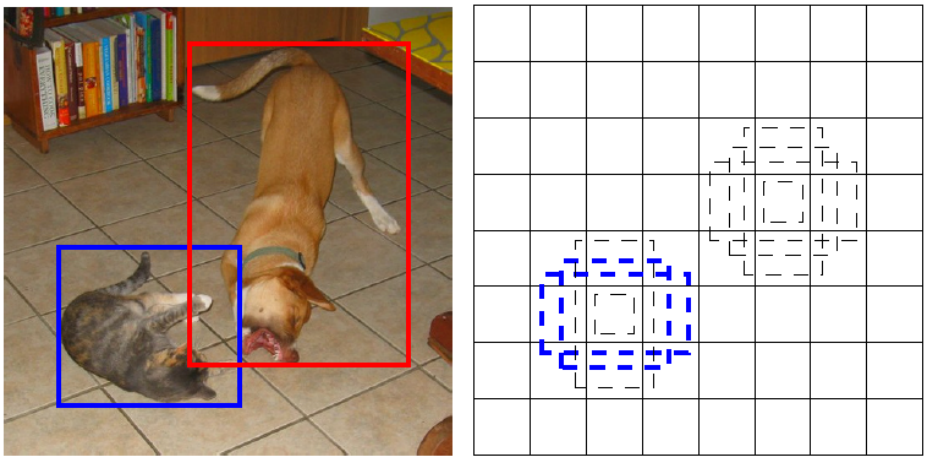
Figura 8 – Esquema que ilustra a arquitetura da SSD.



Fonte: Liu et al. (2016, p. 4).

A SSD divide a imagem em células e analisa individualmente cada uma. Ao analisar uma célula, a SSD avalia se há objetos na mesma. Caso seja percebido que não há nenhum objeto na célula, a mesma é descartada. Para resolver o problema de escala e proporção dos objetos, a SSD opera com diversas âncoras e níveis de zoom. Entretanto, essas diversas âncoras e níveis de zoom acabam por permitir múltiplas detecções de um mesmo objeto. Desta forma, é aplicado o algoritmo *Non-maximum Suppression* (NMS) na saída da SSD, visando excluir essas múltiplas detecções de um mesmo objeto. Um exemplo de como são as âncoras e o mecanismo de divisão das células da imagem pela SSD pode ser visto na Figura 9.

Figura 9 – Exemplo da repartição da imagem em células e das âncoras.



Fonte: Liu et al. (2016, p. 3).

2.3 Few-Shot Learning (FSL)

O *few-shot learning* (FSL) é uma área do DL que busca permitir que modelos sejam capazes de classificar ou detectar objetos com um número pequeno de dados. Dentro da área do

FSL apresentam-se diversos conceitos como *one-shot learning* (1SL) e *zero-shot learning* (0SL). Segundo Wang et al. (2020c), o 1SL consiste na capacidade do modelo de aprender sobre uma classe a partir de um exemplo supervisionado da mesma e o 0SL a partir de nenhum. Segundo o autor, quando não se tem exemplos do objeto, faz-se necessário ter informações que o descrevam, de forma a permitir que o modelo 0SL possa aprender.

2.3.1 *Metric-learning* e *Meta-learning*

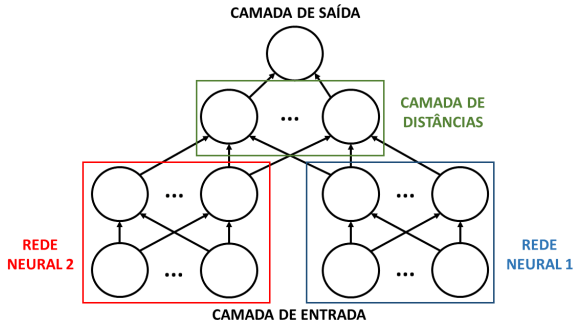
Outros conceitos relevantes na área de FSL são *meta-learning* e *metric-learning*. Segundo Hospedales et al. (2020), o *meta-learning* ou meta-aprendizagem, também conhecida como *learning to learn*, é uma estratégia de treinamento de um algoritmo através de algumas etapas de aprendizado. Durante o treinamento convencional de um modelo, o mesmo é treinado para resolver um problema pontual utilizando uma base de dados, por exemplo classificação de imagens ou detecção de objetos. Já durante o *meta-learning*, um algoritmo mais externo gerencia o aprendizado de um outro algoritmo mais interno, na tentativa de treinar um modelo capaz de resolver um problema. Desta forma, o *meta-learning* é utilizado no FSL para desenvolver modelos capazes de serem retreinados, permitindo com que eles detectem ou classifiquem novas classes, com poucos dados.

No caso do *metric-learning* ou *non-parametric learning*, ainda de acordo com Hospedales et al. (2020), uma comparação é realizada entre os elementos de treinos e os elementos que representam as classes. O modelo aprende a avaliar, a partir do resultado da comparação, se o elemento de treino apresenta a mesma classe do elemento de referência. Com isso, ele aprende a analisar a semelhança ou a diferença entre os dois elementos.

2.3.2 *Siamese Neural Network* (SNN)

A *Siamese Neural Network* (SNN), proposta por Koch, Zemel e Salakhutdinov (2015), é capaz de analisar a semelhança entre duas imagens, sendo possível considerá-la como um algoritmo de *metric-learning*. Originalmente, as entradas da SNN são filtradas por duas CNNs idênticas, que compartilham os mesmos parâmetros. As entradas são filtradas igualmente, para que seja possível compará-las posteriormente. A saída das CNN, que consistem nos *feature maps* das entradas da SNN, passam por uma camada que estima a semelhança entre as duas. Segundo os autores, foi utilizada a função de distância L1 ponderada para calcular a semelhança entre os mapas de características. A saída da camada de distâncias passa pela função *sigmoid*, a qual limita os seus valores entre 0 e 1. É possível ver a arquitetura proposta por Koch, Zemel e Salakhutdinov (2015) da SNN na Figura 10.

Figura 10 – Exemplo da arquitetura de uma SNN.

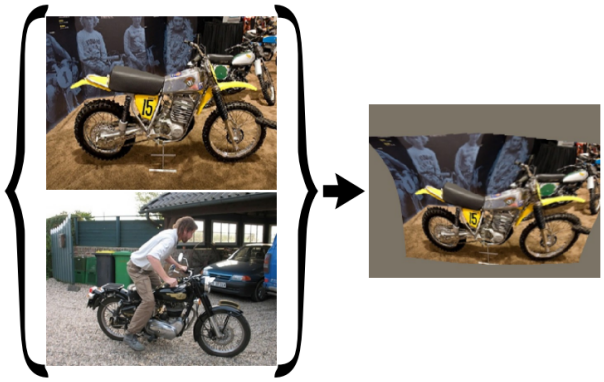


Fonte: Adaptado de Koch, Zemel e Salakhutdinov (2015, p. 3).

2.3.3 Alinhamento Geométrico com *Few-Shot Learning*

Os autores Rocco, Arandjelovic e Sivic (2017) desenvolveram uma arquitetura capaz de analisar o objeto principal de duas imagens, estabelecer pares de correspondências entre os dois objetos e por fim fazer o alinhamento geométrico do objeto da primeira imagem com o da segunda. O alinhamento geométrico é feito a partir da deformação de uma imagem com relação à outra, na tentativa de aproximar geometricamente o objeto contido na primeira com o da segunda. É possível ver na Figura 11 um exemplo dos resultados obtidos pela arquitetura proposta pelos autores, ao realizar o alinhamento geométrico em um par de imagens.

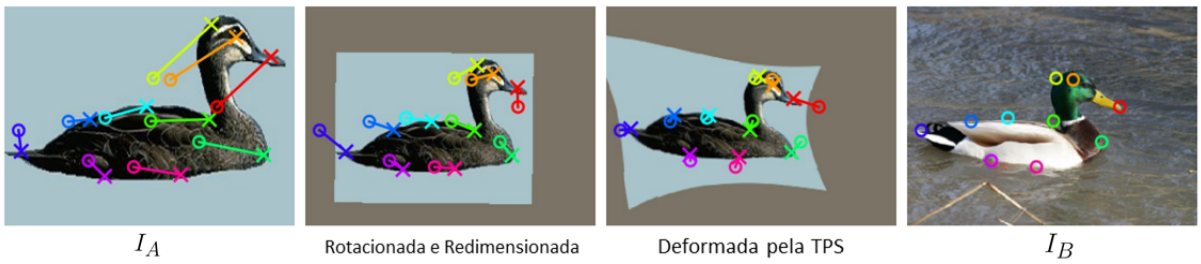
Figura 11 – Exemplo de alinhamento geométrico em um par de imagens.



Fonte: Rocco, Arandjelovic e Sivic (2017, p. 1).

A arquitetura proposta por Rocco, Arandjelovic e Sivic (2017) recebe duas imagens de entrada, a I_A e I_B, e é executada em dois estágios, os quais apresentam três etapas principais: a extração de características, a análise de correlação e a estimação dos parâmetros. O primeiro estágio é o responsável por estimar seis parâmetros que irão redimensionar e rotacionar a I_A com relação a I_B. Já o segundo estágio irá estimar 18 parâmetros para realizar uma transformação *thin-plate spline* (TPS), que irá deformar a I_A, que já foi redimensionada e rotacionada, com relação a I_B. É possível visualizar cada um dos estágios do alinhamento geométrico na Figura 12.

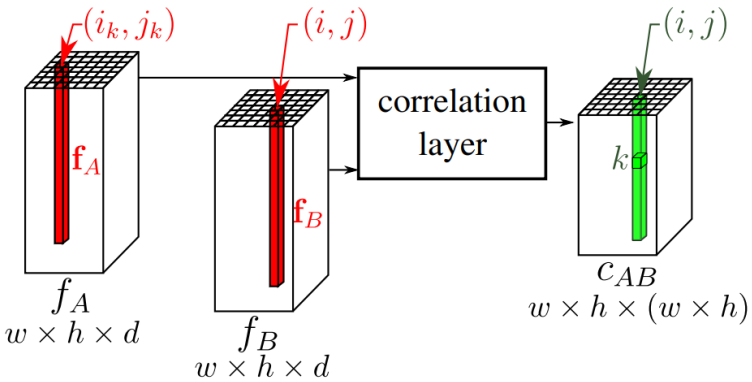
Figura 12 – Estágios da arquitetura de alinhamento geométrico.



Fonte: Adaptado de Rocco, Arandjelovic e Sivic (2017, p. 7).

A primeira etapa, que é a de extração de características, é a mesma para os dois estágios, onde é utilizada a VGG16 combinada com uma arquitetura siamesa, buscando filtrar igualmente ambas as imagens de entrada. Após a etapa de extração de características, as saídas são normalizadas[4] e em seguida é executada a etapa de correlação, a qual irá verificar quais partes de I_A se correlacionam mais com as partes de I_B. A etapa de correlação funciona da mesma forma para os dois estágios. Na Figura 13, é possível ver como é feita a etapa de correlação entre os mapas de características das duas imagens.

Figura 13 – Diagrama de funcionamento da etapa de correlação.

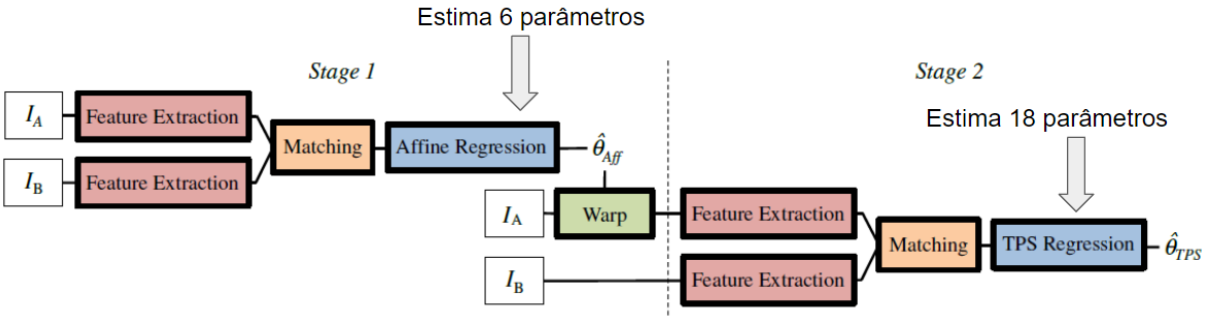


Fonte: Rocco, Arandjelovic e Sivic (2017, p. 3).

A terceira e última etapa é a de estimação dos parâmetros da transformação de cada estágio. Ela é diferente para cada um dos estágios, de forma que no primeiro são estimados seis parâmetros e no segundo 18. Para o primeiro estágio serão estimados seis parâmetros que irão compor a matriz de transformação linear, a qual irá realizar um redimensionamento, rotação e translação da imagem I_A com relação a I_B. Em seguida, a I_A transformada será inserida em outro modelo, que estimará os 18 parâmetros que irão compor a TPS. A TPS será utilizada para deformação a I_A transformada, alinhando geometricamente o objeto de I_A com relação ao de I_B. O fluxo de execução da arquitetura e o diagrama do regressor de parâmetros propostos por Rocco, Arandjelovic e Sivic (2017) podem ser vistos nas Figuras 14 e 15, respectivamente.

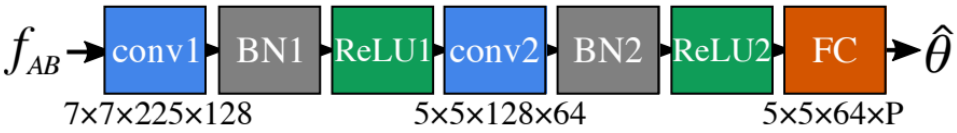
[4] Utiliza-se a normalização por *feature* nos mapas de características.

Figura 14 – Diagrama da arquitetura de Alinhamento Geométrico.



Fonte: Adaptado de Rocco, Arandjelovic e Sivic (2017, p. 5).

Figura 15 – Diagrama do regressor de parâmetros.

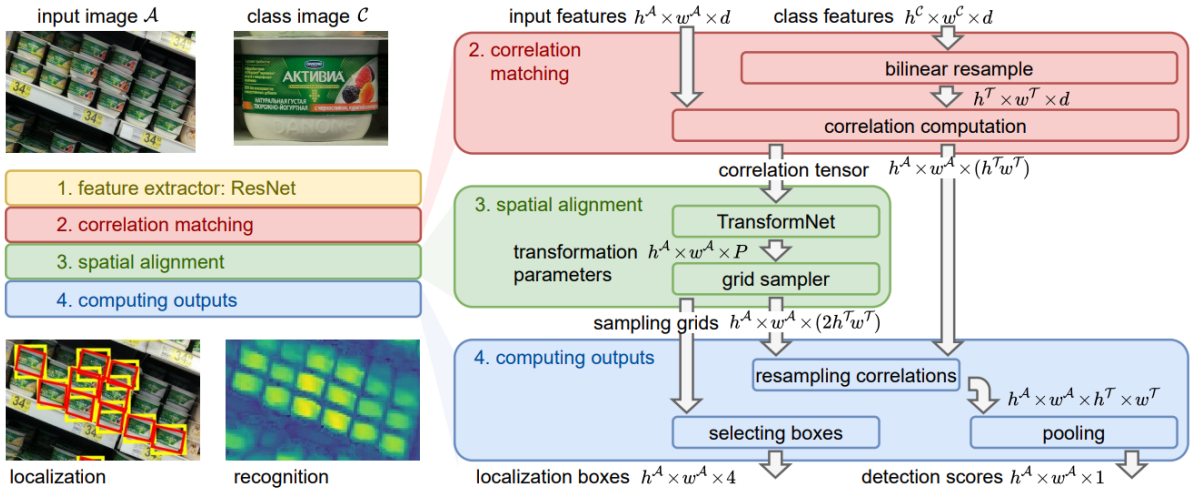


Fonte: Rocco, Arandjelovic e Sivic (2017, p. 4).

2.3.4 *One-stage One-shot Object Detection* (OS2D)

O modelo proposto por Osokin, Sumin e Lomakin (2020), chamado de *One-stage One-shot Object Detector* (OS2D), é capaz de detectar objetos novos a partir de exemplos fornecidos dos mesmos. O modelo tem como entrada duas imagens, sendo uma a imagem de busca e a outra de referência. A imagem de busca poderá ou não conter o objeto procurado. Caso ela contenha uma ou mais instâncias desse objeto, então o modelo irá retornar o BB e o *score* de cada detecção. Com relação a imagem de referência, a mesma irá conter um exemplo do objeto que será procurado na imagem de busca. O modelo OS2D apresenta quatro estágio principais: a extração de características, a análise de correlação, o alinhamento espacial e a computação das saídas. Um diagrama da arquitetura OS2D pode ser visto na Figura 16.

Figura 16 – Diagrama da arquitetura OS2D.



Fonte: Osokin, Sumin e Lomakin (2020, p. 4).

O primeiro estágio da arquitetura OS2D, extração de características, consiste da inserção de duas imagens em uma SNN, cujo extrator de características é uma ResNet. Os dois ramos da SNN apresentam os mesmos parâmetros. No segundo estágio, de análise de correlação, é utilizado o mesmo esquema de correlação proposto por Rocco, Arandjelovic e Sivic (2017), onde é computado um tensor, ou matriz, de quarta dimensão, armazenando a correlação de cada parte da imagem de referência com a de busca. Após a computação desse tensor de quarta dimensão, o mesmo é reorganizado em um tensor de terceira dimensão.

O terceiro estágio consiste na inserção do tensor de terceira dimensão na TransformNet[5] que irá fornecer os parâmetros necessários para realizar a rotação apropriada para cada detecção. Por fim, todas as transformações estimadas são convertidas por um *grid sampler* para a notação em malha, permitindo o alinhamento das classes com as localizações estimadas. A TransformNet apresenta a mesma arquitetura que o regressor de parâmetros proposto por Rocco, Arandjelovic e Sivic (2017), exibido na Figura 15.

O quarto e último estágio é responsável por fornecer os BBs e *scores* de cada detecção. Entretanto, é necessário redimensionar as correlações obtidas, pois segundo os autores o tensor apresenta uma dimensão 255 vezes maior que qualquer tensor criado até o momento, apresentando dificuldades para ser computado. Desta forma, o tensor é reamostrado na tentativa de reduzí-lo a um tamanho computável. Por fim, é aplicado o *average pooling* para a obtenção dos *scores*. Com relação a obtenção dos BBs, as localizações das detecções são estimadas a partir da extração dos mínimos e máximos do tensor em formato de malha, retornado pelo *grid sampler*.

[5] A TransformNet é um estimador de parâmetros para calcular a rotação necessária em cada BB de cada detecção, semelhante ao proposto por Rocco, Arandjelovic e Sivic (2017).

2.4 Trabalhos Relacionados

Nesta subseção serão mostrados alguns trabalhos, onde foi utilizado o FSL na tentativa de solucionar diversos problemas existentes no âmbito da ciência e indústria. Alguns autores visaram atacar problemas que envolvem a detecção e classificação de sons e outros propuseram modelos para classificação de imagens e detecção de objetos com FSL. Alguns trabalhos buscaram resolver problemas de detecção e classificação de texto de linguagem natural.

O primeiro trabalho, proposto por Droghini et al. (2018), descreve a utilização de uma rede neural siamesa para análise de áudio, na tentativa de detectar se a pessoa caiu. Segundo os autores, a utilização de uma rede neural siamesa se justifica pelo fato de ser difícil ter um conjunto de dados que descrevam inteiramente a ação de cair, sendo necessário incrementar esse conjunto à medida que o tempo passa. Para isso, estratégias de FSL são úteis, tendo em vista que elas permitem o tipo de estratégia citado anteriormente.

Michaelis, Bethge e Ecker (2018) propuseram uma forma de segmentar um objeto, em um cenário com bastante oclusão, utilizando apenas de um exemplo desse objeto. Para isso, eles combinaram uma rede neural siamesa, utilizada para comparação, e uma U-net, responsável pela segmentação do objeto. Além da arquitetura citada anteriormente, foi também utilizada uma MaskNet, a qual extraia os objetos do *background* e comparava com o exemplo fornecido, permitindo achar múltiplos candidatos do mesmo objeto na imagem. Também foi proposto um *dataset* chamado de *cluttered Omniglot*.

No trabalho de Ustyuzhaninov et al. (2018) é proposto um modelo capaz de segmentar partes da imagem a partir da textura fornecida. O modelo foi analisado em dois *datasets*: o CollTex e o Omniglot. A arquitetura utilizada consiste da inserção de duas imagens no modelo, uma sendo a textura de referência e a outra a imagem de entrada. Em seguida as entradas passam pelos estágios de codificação, correlação das partes e por fim decodificação, gerando, desta forma, uma máscara de predição.

O trabalho desenvolvido por Fu et al. (2019) propõe uma arquitetura FSD *single-stage*, chamada de Meta-SSD, capaz de detectar novos objetos a partir de um *fine-tuning* realizado com poucos exemplos anotados dos objetos de interesse. Esse modelo baseia-se na ideia de *meta-learning*, diferentemente dos modelos que utilizam estruturas semelhantes a SNN, a qual baseia-se na ideia de *metric-learning*. Apesar de ser capaz de detectar classes que não estavam presentes no conjunto de treinamento, o modelo proposto pelos autores exige que um retreinamento de algumas camadas seja feito, de modo a condicionar que o modelo detecte aquelas classes.

É desenvolvido no trabalho de Hsieh et al. (2019) um modelo que visa lidar com o desafio de detectar objetos a partir de um exemplo dos mesmos. Desta forma, são dadas imagens de referência, que descrevem os objetos que devem ser encontrados, e uma imagem de busca,

na qual o modelo deverá buscar os objetos das imagens de referência. O modelo é de uma rede neural siamesa, seguido de um módulo chamado de *Non-local features*, o qual é capaz de propor regiões de interesse na imagem para cada par de referência e busca. Em seguida, tem o módulo *squeeze-and-co-excitation*, o qual buscará enfatizar quais regiões são as mais relevantes e por fim o módulo de comparação, que irá avaliar a similaridade entre os objetos de referência e as regiões encontradas pelo modelo.

Segundo Kang et al. (2019), foi desenvolvido um FSD *single-stage* capaz de detectar objetos de classes não vistas a partir do retreinamento do modelo, com poucos exemplos anotados das classes. O modelo proposto é capaz de se adaptar rapidamente a novas classes utilizando de um *meta feature learner* e um *reweighting module*. O *meta feature learner* extrai as meta características que generalizam os objetos da classe nova e o *reweighting module* reavalia a relevância de cada *feature* extraída para detectar objetos da nova classe.

De acordo com Michaelis et al. (2018), foi desenvolvida uma arquitetura, chamada de Siamese Mask R-CNN, a qual estendia a arquitetura padrão da Mask R-CNN, proposta por He et al. (2017), de forma que a mesma fosse capaz de realizar a segmentação de objetos utilizando de FSL. A modificação na arquitetura da Mask R-CNN consistiu na alteração do *backbone* da rede, no que antes era uma ResNet 50, para uma SNN e foi acrescentada a distância L1 entre a imagem de busca e a imagem de referência na RPN. O modelo proposto recebe de entrada imagens de referência, que descrevem os objetos, e uma imagem de busca, a qual os objetos serão buscados. Por fim, o modelo irá fornecer uma máscara que irá segmentar as instâncias dos objetos encontrados na imagem de busca.

No trabalho de Wang, Ramanan e Hebert (2019), é proposto um *framework* de *meta-learning* que permite, simultaneamente, classificar e detectar objetos utilizando FSL. O *framework* permite que dados provenientes de uma extensa base de dados sejam reaproveitados para aprender novas classes. Os parâmetros do modelo são divididos em dois conjuntos: o conjunto de parâmetros agnósticos a classes e o de parâmetros específicos das classes. Desta forma, o modelo é capaz de prever os pesos dos parâmetros necessários para construir os detectores das novas classes a partir de poucos exemplos da mesma.

Li et al. (2020b) propôs um *framework* que mescla o *metric-learning* com o *meta-learning* chamado de MM-FSOD. A estratégia utilizada permite que o modelo aprenda características intra-classe com o módulo meta-representação (MR), sem a necessidade de um processo de *fine-tuning*. Desta forma, o módulo MR é treinado para ser capaz de reconstruir características de alto nível dos objetos, enquanto isso, é analisada a similaridade entre os objetos utilizando o coeficiente de correlação Pearson (PR). Os autores afirmam que o PR se apresenta mais eficiente que a distância do cosseno para análise de similaridade entre os objetos.

De acordo com Li et al. (2020a), é proposto um modelo de detecção de objetos *two-stage*, que não necessita de retreinamento. O primeiro estágio do modelo é chamado de *Matching-FCOS network* e o segundo de *Structure-Aware Relation module*. A combinação dos

dois módulos integra a arquitetura livre de âncoras, baseada em *metric-learning*, com o *pipeline* de detecção proposto na R-CNN.

Neste trabalho, Michaelis, Bethge e Ecker (2020) adapta o modelo proposto em seu último artigo, *Siamese Mask-RCNN*, para uma estratégia de detecção de objetos. Desta forma, o modelo proposto não precisa de retreinamento, por se tratar de um modelo baseado em *metric-learning*. O modelo consiste em uma rede neural siamesa, seguida de um propositor de regiões. Esse tipo de modelo de detecção de objetos *two-stage* é muito utilizado em FSOD pelo fato do mesmo apresentar o estágio de classificação e detecção separados.

Perez-Rua et al. (2020) adaptou a arquitetura CenterNet, *single-stage* livre de âncoras, para uma abordagem FSL com *meta-learning*. Essa arquitetura exige retreinamento, sempre que for adicionar uma nova classe, apesar de necessitar de poucos exemplos. O modelo proposto apresenta um módulo chamado de *Class Code Generator*, o qual irá aprender os parâmetros de cada classe e injetá-los no *Object Locator*[6].

No trabalho proposto por Rahman et al. (2020), foi criado o termo *Any-shot* para descrever modelos que são capazes de detectar objetos que estavam e não estavam presentes no seu conjunto de treinamento. Essa estratégia define que o detector de objetos será capaz de achar os objetos para o qual foi treinado, de forma convencional, e também os objetos o qual o modelo é ajustado, a partir de estratégias de FSL. Para isso, foi proposto um modelo capaz de aprender a detectar objetos a partir de estratégias de 0SL e FSL.

Shi et al. (2020) buscaram desenvolver um modelo de FSL capaz de detectar a ocorrência de eventos sonoros. O modelo desenvolvido utiliza de *meta-learning* para aprender, a partir de poucas amostras, novos eventos sonoros a serem detectados. Diferentemente da visão computacional, o FSL é pouco estudado na área de detecção acústica.

No trabalho proposto por Wang et al. (2020b), foram adaptados alguns modelos de FSL baseados em *metric-learning* para construir um novo modelo capaz de detectar sons similares, a partir de alguns exemplos fornecidos. Os modelos testados foram: SNN, *Matching Networks*, *Prototypical Networks* e *Relation Networks*. O modelo de melhor resultado foram as *Prototypical Networks*.

Wang et al. (2020a) propuseram uma estratégia de *fine-tuning* para os modelos de detecção de objetos, mostrando que ao alterar apenas as cabeças de regressão e classificação são suficientes para se obter bons resultados em uma abordagem de FSL baseada em *meta-learning*. Os autores utilizaram um modelo *two-stage* baseado na R-CNN.

No trabalho proposto por Wu, Sahoo e Hoi (2020), foi desenvolvida uma nova arquitetura chamada de Meta-RCNN, a qual aplica na arquitetura Faster RCNN uma abordagem de FSL baseada em *meta-learning*. O módulo RPN, meta-treinado, propõe regiões específicas para cada classe, enquanto o classificador de objetos realiza uma classificação com apenas

[6] Módulo da CenterNet responsável por detectar os objetos.

poucos exemplos.

Wu et al. (2020) desenvolveu uma estratégia de processamento dos dados de entrada que permite uma melhor detecção dos objetos em diferentes escalas, utilizando modelos de FSL. A estratégia proposta foi intitulada de *Multi-scale Positive Sample Refinement* (MPSR). O MPSR gera amostras positivas em várias escalas como pirâmides de objetos, permitindo uma predição refinada em múltiplas escalas. A estratégia desenvolvida é aplicada na Faster R-CNN.

Por fim, foi construída, no trabalho de Xia et al. (2020), uma arquitetura FSL para detectar a intenção em alguns textos de linguagem natural. Para isso, foi proposto o modelo *Conditional Text Generation with BERT* (CG-BERT), capaz de gerar novos dados textuais a partir de uma base de dados. Também foi proposto um modelo capaz de detectar a intenção do texto utilizando de FSL chamado de *Generalized Few-Shot Intent Detection* (GFSID).

3 METODOLOGIA

Nesta seção serão descritas as etapas que serão realizadas para a obtenção dos resultados. A primeira etapa será definida pela construção de uma base de dados com as fotos dos kits e seus respectivos componentes. Essa base de dados será utilizada para avaliar a performance das soluções PIMG e OS2D+. Após a obtenção das imagens dos kits, as mesmas serão anotadas. A anotação consiste uma etapa relevante, pois permitirá que testes automatizados sejam desempenhados nas duas soluções propostas. Tendo a base de dados criada, será iniciado o desenvolvimento de uma rotina de processamento de imagem capaz de inspecionar os kits.

Com a solução PIMG desenvolvida, o modelo proposto por Osokin, Sumin e Lomakin (2020) será adaptado, visando solucionar o problema da inspeção dos kits. A ideia de fornecer os dois tipos de soluções permitirá compará-las, salientando seus prós e contras. Em seguida, serão definidas as métricas calculadas a partir dos resultados das duas soluções. Por fim, serão analisados os resultados obtidos pelas duas soluções, buscando julgar qual das duas apresenta maior capacidade de solucionar a problemática.

3.1 Obtenção da base de dados

As fotos dos kits e seus respectivos componentes foram obtidas utilizando uma câmera de resolução 3482x2844 píxels. A partir da necessidade de se obter exemplos de alta qualidade de cada componente, foi utilizada uma câmera de uma alta resolução, com mais de 1920x1080 píxels. Após a obtenção das fotos, exemplos de cada componente foram separados em pastas, para posteriormente serem utilizados. A definição dos kits pode ter o número de fotos e componentes diferentes.

Para o processo de obtenção das fotos, foram estabelecidas algumas restrições. A primeira restrição consistiu na definição de que o tabuleiro, no qual os componentes de cada kit foram posicionados, era da cor verde. Essa restrição se justifica pelo fato de que, para a solução PIMG, era necessário segmentar os componentes que foram posicionados sobre o tabuleiro utilizando da segmentação por cor. A segunda restrição consistiu na limitação do posicionamento dos componentes, pois para que os modelos fossem capazes de detectar o mesmo componente em múltiplas posições, seriam necessários exemplos de cada componente em cada posição.

O processo de obtenção das fotos dos kits se deu em algumas etapas. Na primeira etapa o tabuleiro vazio foi posto sobre a câmera e em seguida, para cada componente posicionado em cima do tabuleiro, uma foto foi obtida. Após todos os componentes de um kit estarem posicionados sob o tabuleiro, todos os componentes tinham sua posição alterada e uma nova foto era obtida. Após capturar algumas fotos dos componentes em posições diferentes, fixava-se a posição de todos eles e variava a posição do ponto de iluminação. Para cada nova posição do ponto de iluminação, uma nova foto era registrada. Por fim, a medida em que cada componente

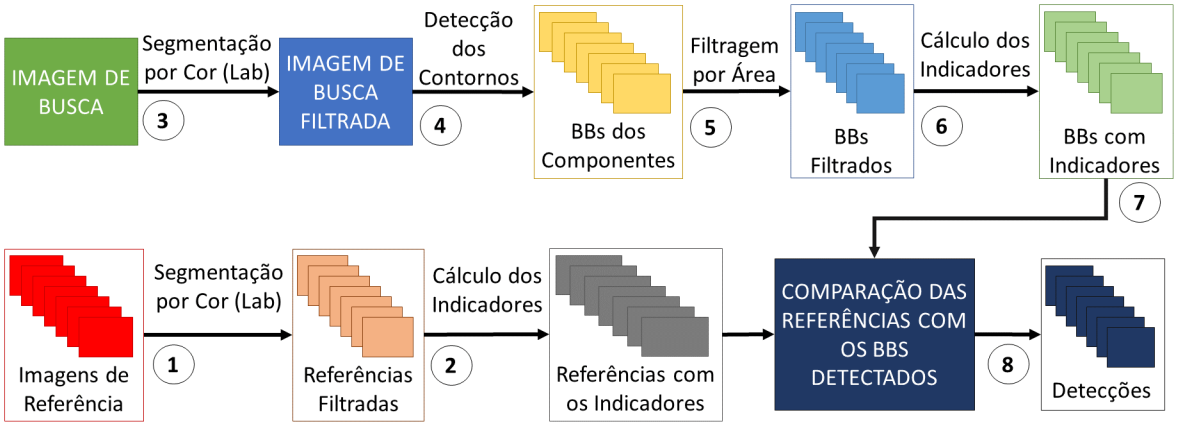
fosse removido da superfície do tabuleiro, uma nova foto era obtida, até o momento em que não restasse mais nenhum componente. Nesse momento, iniciava-se o processo de obtenção das fotos de outro kit.

Com as fotos dos kits e exemplos dos componentes separados em pastas, iniciou-se o processo de anotação da base de dados. O processo de anotação foi realizado para cada imagem de cada kit separadamente. Primeiramente era desenhado um retângulo em volta de cada componente da imagem. Em seguida, era atribuído um rótulo a cada retângulo. O retângulo desenhado em volta do componente fornece três informações relevantes: qual a posição posição (x e y) do componente, qual a dimensão (altura e largura) do componente e a qual classe do componente pertencia.

3.2 Solução com Processamento de Imagem (PIMG)

Para construir a solução PIMG foi necessário definir a cor do tabuleiro dos componentes. Para isso, buscou-se uma cor que não fosse comum entre os componentes. Devido a isso, foi escolhida a cor verde, sendo possível segmentá-la utilizando o espaço de cores Lab. O fluxo de execução da solução PIMG é descrito em oito etapas, como pode ser visto na Figura 17.

Figura 17 – Esquema do *pipeline* de execução da solução PIMG.



Fonte: Elaborado pelo autor.

3.2.1 Etapa 1: Segmentação das imagens de referência

A primeira etapa consistiu em aplicar a segmentação no espaço de cores Lab[7] nas imagens de referência[8]. Após a remoção do *background*, a imagem era binarizada, de modo a criar uma máscara. Essa etapa fez-se necessária para o cálculo dos indicadores, que serão utilizados para a comparação dos componentes.

[7] A segmentação utilizando o espaço de cores Lab foi escolhida por apresentar o melhor resultado, quando comparado com os outros espaços de cores (HSV e RGB).

[8] Imagens de todos os exemplos dos componentes relacionados ao kit que está sendo processado.

3.2.2 Etapa 2: Cálculo dos indicadores das imagens de referência

Tendo as imagens de referência filtradas, sem o *background*, iniciou-se a segunda etapa que consistiu no cálculo de três indicadores relativos a cada imagem de referência. O primeiro indicador era o de cor, ele era constituído de nove histogramas de cores, pois ele armazenava o histograma referente aos três canais dos espaços de cores RGB, HSV e Lab. Logo, eram os histogramas dos canais R, G, B, H, S, V, L, a e b. Cada histograma era composto por 256 números de 0 a 1. Portanto, o indicador de cor era descrito por 2304 números flutuantes.

O segundo indicador foi o de contorno, onde as bordas das imagens de referência foram extraídas utilizando o algoritmo Canny, proposto por Canny (1986), e armazenadas em uma matriz bidimensional. O terceiro indicador era o de área, o qual consistia no número de pixels diferentes de zero, calculados sobre cada imagem de referência binarizada. O indicador de área se fez importante para restringir que os componentes, com os indicadores de cor e contorno semelhantes, fossem confundidos.

3.2.3 Etapa 3: Segmentação da imagem de busca

A terceira etapa consistiu na segmentação no espaço de cores Lab da imagem de busca[9]. Da mesma forma que nas imagens de referência, após a remoção do *background*, a imagem foi binarizada. A etapa de segmentação do *background* da imagem de busca, diferentemente das imagens de referência, se fez necessária para poder extrair os candidatos a componentes.

3.2.4 Etapa 4: Detecção de contornos na imagem de busca filtrada

Foi utilizada uma rotina de detecção de contornos sobre a imagem de busca filtrada, na tentativa de encontrar os candidatos a componentes. Caso a rotina encontrasse dois contornos, um contido no outro, o mais interno era descartado. Desta forma, a imagem de busca se tornou uma lista de BBs.

3.2.5 Etapa 5: Filtragem por área dos BBs dos componentes

Após a detecção dos BBs, os mesmos foram submetidos a um filtro de área, na tentativa de excluir BBs muito pequenos, cuja área fosse menor que 10000 pixels. Essa filtragem era realizada na tentativa de excluir ruídos provocados pela variação da iluminação sobre o tabuleiro. O filtro por área também serviu para remover pequenos pedaços dos componentes que foram separados do restante do corpo da peça por uma falha na segmentação.

3.2.6 Etapa 6: Cálculo dos indicadores dos BBs dos componentes

Tendo os BBs dos componentes filtrados, iniciou-se o cálculo dos mesmos três indicadores utilizados nas imagens de referências. Esses indicadores foram utilizados para

[9] Imagem cujos componentes serão inspecionados.

comparar cada imagem de referência com cada BB de componente.

3.2.7 Etapa 7: Comparação dos BBs dos componentes com as imagens de referência

Nesta etapa foi feita uma comparação de todos os BBs dos componentes com todas as imagens de referência. Os indicadores de cor foram comparados calculando a média da distância do cosseno sobre os nove histogramas calculados, com isso obteve-se a distância por cor. Em seguida foram comparados os indicadores de contorno, obtendo-se a distância por contorno. A distância por contorno era calculada a partir da comparação dos momentos de Hu[10] de cada um dos contornos.

Logo após, utilizou-se da equação 3.1, onde a a_r representa o indicador de área de uma imagem de referência, a a_c é o indicador de área do BB de um componente e D_{area} é a distância por área. Por fim, foi calculada a distância composta, onde foram somadas as três distâncias. A distância composta representou o quão diferente o BB de um componente era de uma imagem de referência. Com isso já era possível determinar qual a classe de cada BB dos componentes.

$$D_{area} = |(a_r/a_c) - 1| \tag{3.1}$$

Fonte: Elaborado pelo autor.

3.2.8 Etapa 8: Definição das classes das detecções

Após o cálculo da distância composta para cada BB dos componentes, foi analisada qual imagem de referência apresentou a menor distância com relação aos indicadores do BB do componente. Após encontrar qual foi a menor distância, atribuiu-se a classe da imagem de referência para o BB do componente. Os BBs dos componentes com suas classes definidas eram chamados de detecções.

3.3 Solução OS2D+

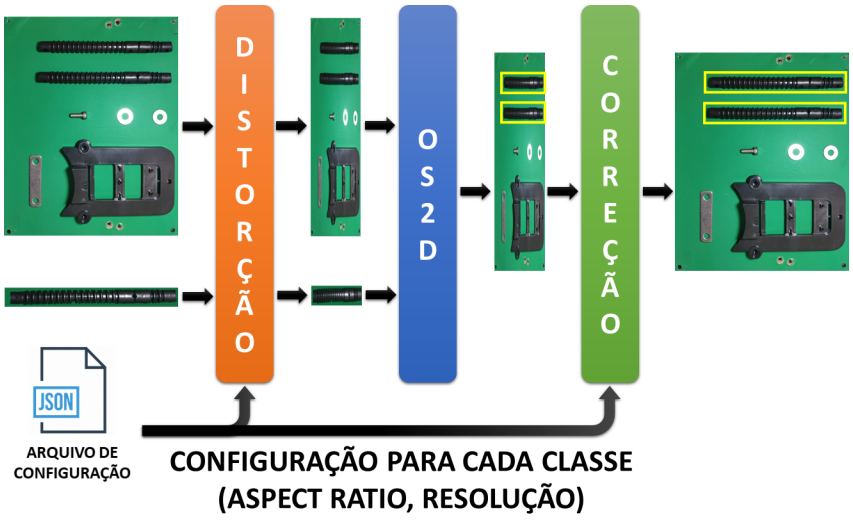
A solução OS2D, consistiu na adaptação do OS2D, modelo proposto por Osokin, Sumin e Lomakin (2020). O OS2D foi escolhido devido ao mesmo ser um modelo FSOD 1SD, capaz de detectar novas classes sem a necessidade de retreinamento. O modelo utiliza de uma abordagem de *metric-learning*, utilizando das redes neurais siamesas. O OS2D apresentou algumas limitações arquiteturais, sendo necessário realizar algumas modificações para que o mesmo pudesse resolver o problema de inspeções dos kits.

O OS2D tem um mecanismo semelhante ao da SSD, proposta por Liu et al. (2016), a qual utiliza de âncoras para detectar objetos de diferentes *aspect ratios* e escalas. O OS2D

[10] Segundo OpenCV (2021), os momentos Hu são sete momentos invariantes à translação, rotação e escala.

utiliza de nove âncoras, sendo três *aspect ratios* e três escalas. Essa quantidade pequena de âncoras permitia que o modelo fosse capaz de detectar a grande maioria dos objetos. Entretanto, no que diz respeito a inspeção de kits, existem componentes dos mais variados formatos e *aspect ratios*, sendo necessário a adição de mais âncoras ao modelo.

Figura 18 – Funcionamento das camadas de distorção e correção.



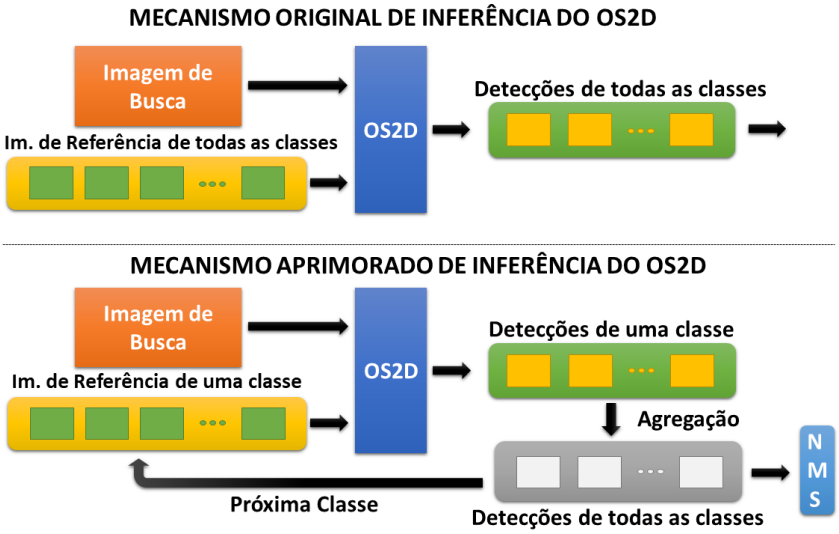
Fonte: Elaborado pelo autor.

A adição de mais âncoras ao modelo exigia o retreinamento do mesmo. Na tentativa de evitar isso foi proposta uma camada de preprocessamento que distorce a imagem de busca e as imagens de referência, de forma individual para cada componente. Essa camada de preprocessamento foi nomeada de camada de distorção. Logo, componentes compridos[11] foram achatados na tentativa de aproximá-los de uma das âncoras do modelo. Por fim, as predições do modelo eram inseridas em uma camada que as corrigia para o formato original. Essa camada foi chamada de camada de correção. O *pipeline* de execução descrito para as duas novas camadas pode ser visto na Figura 18.

A camada de distorção recebia quatro parâmetros: a imagem de busca, as imagens de referência, a resolução final da imagem de busca e a proporção largura e altura. O parâmetro da resolução final da imagem de busca foi utilizado para diminuir o tempo de processamento do modelo, tendo em vista que componentes grandes precisariam de resoluções menores e componentes pequenos de resoluções maiores. Já o parâmetro referente à proporção largura e altura definia o quanto as imagens de referência e a imagem de busca deveriam ser deformadas, em termos de largura e altura. Por fim, as imagens de referência e de busca eram distorcidas, ao seu formato original, apropriadamente.

[11] Com a largura ou a altura mais que três vezes maior que a outra.

Figura 19 – Funcionamento das camadas de distorção e correção.

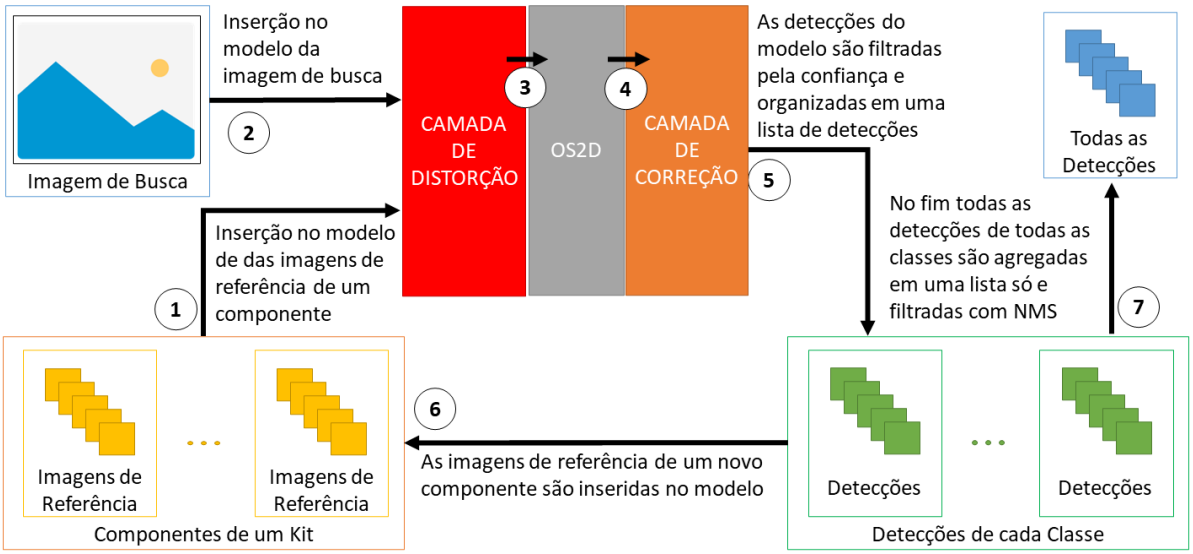


Fonte: Elaborado pelo autor.

Além da camada de distorção que foi proposta, para sanar os problemas das âncoras, foi realizada uma modificação no mecanismo de inferência da OS2D. Anteriormente, todas as imagens de referência dos componentes eram inseridas em batelada, juntamente com a imagem de busca, no OS2D. Isso causava um problema de estouro de memória, tendo em vista que um kit poderia ter muitos componentes e que cada componente poderia ter mais do que uma imagem de referência do mesmo. Tendo em vista isso, modificou-se a forma de se realizar a inferência no modelo OS2D, de modo que agora, para um kit, foram realizadas inferências para cada componente, onde eram inseridas apenas as imagens de referência relacionadas a um componente. É possível visualizar o funcionamento dos dois mecanismos, original e aprimorado, na Figura 19.

Após a inferência de todos os componentes no modelo OS2D, foram combinadas as saídas de forma a agregar todas as detecções realizadas pelo modelo. Por fim, essa lista de detecção era filtrada pela confiança do modelo sobre cada predição, buscando remover detecções equivocadas. Quando as detecções de todas as classes foram agregadas na mesma lista, realizou-se a filtragem com NMS, evitando múltiplas detecções na mesma localização na imagem. A modificação no modo de inferência da OS2D causou um aumento no tempo de inferência para cada kit, entretanto, viabilizou a detecção de mais componentes e mais imagens de referência para cada componente. O fluxo de execução de uma inferência na solução OS2D+ foi descrito em sete etapas, como pode ser visto na Figura 20.

Figura 20 – Esquema do *pipeline* de execução da Solução OS2D+.



Fonte: Elaborado pelo autor.

3.3.1 Etapa 1: Processamento das imagens de referências

Para cada kit era definido um arquivo de configuração no formato JSON, o qual especificava o caminho das imagens de referência para cada componente, bem como também a resolução da imagem de busca, a proporção largura e altura, a quantidade de componentes dessa classe e o limiar de confiança. Desta forma, as imagens de referência de cada componente foram lidas, distorcidas e redimensionadas de acordo com as especificações de cada classe. Por fim, era aguardada a deformação da imagem de busca, para que a inferência no modelo pudesse ser executada.

3.3.2 Etapa 2: Processamento da imagem de busca

O modelo OS2D+ proposto executou uma inferência por cada classe de componente. Devido a isso, a imagem de busca pôde ser deformada de acordo com cada classe de componente, seguindo as especificações que foram definidas no arquivo de configuração do kit. A deformação realizada nas imagens de referência de um componente precisava ser a mesma realizada na imagem de busca, caso contrário seria difícil encontrar o objetivo descrito na referência, na imagem de busca.

3.3.3 Etapa 3: Inferência no modelo OS2D+

Com o resultado fornecido pela camada de distorção, realizou-se a inferência de cada um dos componentes. As detecções encontradas pelo modelo serão posteriormente inseridas na camada de correção, que irá corrigi-las para o formato original, definido antes da camada de

distorção. O tempo de inferência no modelo OS2D+ variou com relação a resolução definida para cada classe e também para o número de imagens de referência para cada componente.

3.3.4 Etapa 4: Correção das detecções do modelo OS2D+

Devido à distorção aplicada pela camada de distorção nas imagens de busca e de referência, realizou-se o procedimento inverso após a saída do modelo, buscando corrigir as detecções que estavam distorcidas. Desta forma, utilizou-se a mesma proporção largura e altura, especificada no arquivo de configuração de kit, para a realização da distorção reversa. Apesar das coordenadas do BB, que saem do modelo, sejam dadas em porcentagens, foi necessário convertê-las para valores absolutos.

3.3.5 Etapa 5: Armazenamento e Filtragem das detecções

Após as detecções do modelo terem sido corrigidas, pela camada de correção, as mesmas foram filtradas pelo limiar de confiança, conforme estabelecido no arquivo de configuração do kit. Com as detecções filtradas, as mesmas foram adicionadas a uma lista de detecções, que por fim foram agregadas em uma lista única.

3.3.6 Etapa 6: Inferência de uma nova classe de componente

As cinco primeiras etapas representaram a inferência de uma classe de um componente do kit. A sexta etapa descreveu a inferência da próxima classe do kit. Quando todas as classes foram propriamente inferidas, o algoritmo partiu para a próxima etapa de execução.

3.3.7 Etapa 7: Agregação e filtragem de todas as detecções em uma lista

Após todas as classes de componentes do kit serem propriamente inferidas, elas foram filtradas utilizando do NMS. Com a realização de diversas inferências no mesmo modelo, com diferentes imagens de referência, o modelo propôs algumas detecções para componentes muito semelhantes, equivocadamente. Desta forma, o NMS removeu essas múltiplas detecções no mesmo lugar, preservando a que tinha a maior confiança. Portanto, mesmo que o modelo propusesse que dois componentes semelhantes ocupassem a mesma posição da imagem, apenas a que teve maior confiança foi mantida.

Após o desenvolvimento da solução OS2D+, foi realizada uma comparação dos dois modelos (OS2D e OS2D+) buscando analisar se as modificações realizadas no modelo original realmente melhoraram sua performance em detectar objetos. Além disso, buscou-se também analisar como seria a performance do modelo original sobre uma base de dados de componentes com diversos *aspect ratios* diferentes, visando elucidar ainda mais a fragilidade do modelo original e mostrar a correção no modelo aprimorado.

3.4 Métricas estabelecidas

Foram utilizados três conjuntos de métricas: as capazes de avaliar a qualidade do BB predito pelo modelo, as que buscaram determinar quanto o modelo acertava e as que analisavam o tempo gasto para realizar a tarefa. No conjunto das métricas capazes de avaliar a qualidade do BB, utilizou-se a análise da IoU média para cada kit e classe de componente. Já nas métricas que buscavam determinar quanto o modelo acertava, foi utilizado o *precision* e o *recall* médios para cada kit e classe de componente. Tanto o *precision* quanto o *recall* foram analisados com 50% e 75% de IoU. Por fim, foi registrado o tempo necessário para processar cada kit utilizando as duas soluções.

Na análise da IoU média foi possível visualizar a qualidade do BB predito por cada solução. Quando se analisou a IoU média por kit, buscou-se entender se a organização dos componentes sobre o tabuleiro ou se a iluminação acabou por dificultar a percepção da verdadeira dimensão do componente. Ao analisar a IoU média por cada classe de componente, buscou-se entender se existiu algum componente que dificultou a detecção do BB para cada uma das duas técnicas. Isso podia acontecer devido ao formato do componente, ou até mesmo se ele apresentava uma parte de seu corpo que refletia a luz ou o entorno.

A utilização do *precision* médio permitiu entender o quanto das detecções realizadas pelas soluções realmente eram relevantes de serem feitas. O *precision* médio foi analisado com detecções com IoU maior que 50% e com IoU maior que 75%. Essa análise em múltiplas IoU buscou entender a qualidade das detecções, pois caso a solução apresentasse um resultado melhor para uma IoU menor, significava que apesar da solução ter acertado as classes dos componentes, a mesma não conseguiu detectar bem o contorno dos mesmos. O *precision* médio também foi analisado para cada kit individualmente e para cada classe de componente.

A análise do *recall* médio buscou calcular se as soluções foram capazes de detectar o que era relevante. Para isso, foi feita a análise do *recall* médio sobre 50% e 75% de IoU. Da mesma forma que o *precision*, a análise com múltiplas IoU permitiu uma análise da qualidade das detecções. O *recall* médio foi analisado para cada kit individualmente e para cada classe de componente.

Além de verificar a qualidade das detecções fornecidas por cada solução, também foi analisado em quanto tempo cada solução conseguiu desempenhar a mesma tarefa. Para isso, foram medidos o tempo necessário que cada uma levou para fornecer as detecções de cada kit. Entretanto, cada solução funcionou de uma forma e, devido a isso, o modo como elas realizaram a inferência mudou o tempo necessário para completar a tarefa. O tempo gasto pela solução PIMG estava mais relacionado com o número de BBs dos componentes encontrados na imagem, logo, quanto mais BBs ela encontrava, mais ela demorava para processar tudo. Já o tempo demandado pela solução OS2D+ dependeu mais do número de imagens de referência utilizado, portanto, quanto mais exemplos de componentes ela teve, mais tempo ela demorou pra processar tudo.

3.5 Comparação dos resultados obtidos

Após todas métricas serem calculadas corretamente, foi realizada a comparação dos resultados obtidos entre as duas soluções, visando entender em que momentos uma se fez melhor que a outra e qual, no fim, apresentou-se mais capaz de resolver o problema de inspeção de kits. Foram plotados também alguns gráficos, os quais forneceram um entendimento mais visual à resposta. Por fim, foi realizada uma análise dos gráficos plotados, buscando explicar os resultados obtidos.

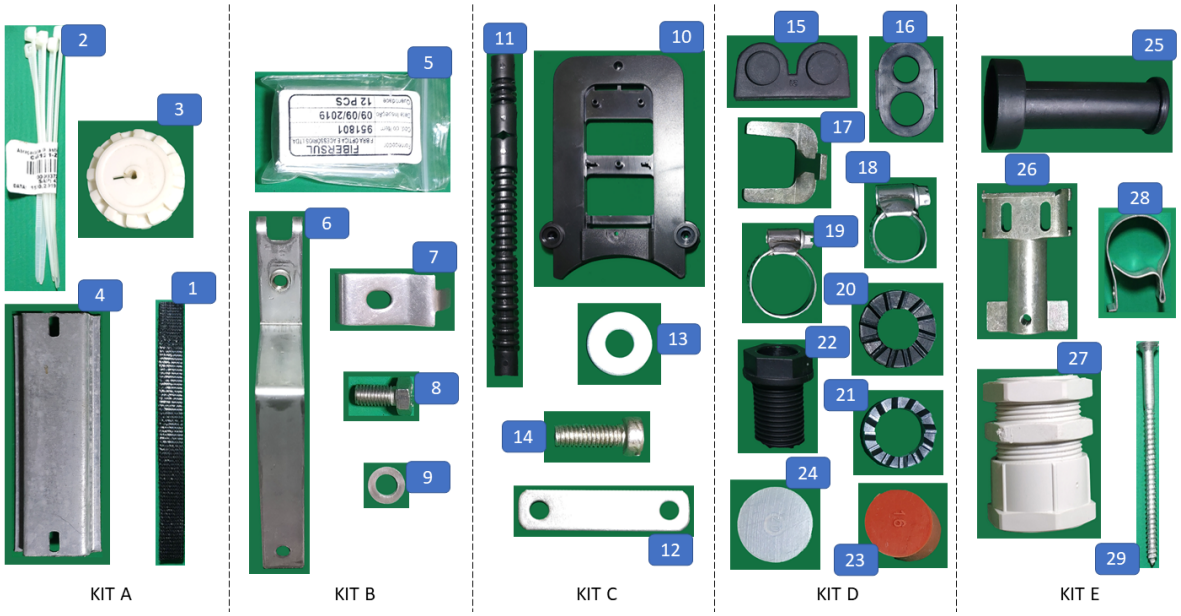
4 RESULTADOS E DISCUSSÕES

Nesta seção serão apresentados os resultados obtidos nas etapas descritas na metodologia. Primeiramente, serão mostrados os resultados obtidos no processo de criação de uma base de dados. Em seguida, serão exibidos e comentados os gráficos com os resultados das métricas calculadas para as duas soluções. Por fim, serão comparados os resultados das mesmas soluções, analisando o desempenho das duas ao tentar resolver o problema de inspeção de kits.

4.1 Base de dados criada

Para desenvolver a base de dados, foram selecionados componentes e peças obtidos em uma linha de montagem de kits industrial real, visando aproximar mais ainda o problema replicado de uma situação existente. No total, foram tiradas 111 fotos, as quais foram divididas em cinco kits. Em relação à quantidade de fotos e classes de componentes, os kits ficaram da seguinte forma: A - 18 fotos e quatro classes; B - 21 fotos e cinco classes; C - 21 fotos e cinco classes; D - 29 fotos e 10 classes; e E - 22 fotos e cinco classes. É possível ver os kits e seus componentes, juntamente com o número identificador de cada classe, na Figura 21.

Figura 21 – Fotos dos kits e seus componentes.



Fonte: Elaborado pelo autor.

Cada componente de cada kit pode ter mais de uma imagem de referência e cada kit pode conter 2 ou mais elementos de uma mesma classe. Desta forma, cada componente apresenta o número de imagens de referência que serão usadas na detecção e quantos dele devem ser achados no kit, para que o mesmo seja julgado como correto. Portanto, a Tabela 1. o número

de imagens de referência e a quantidade de cada componente.

Tabela 1 – Relação dos componentes com o número de imagens de referência e quantidade.

Componente	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Nº de Exemplos	2	2	2	1	2	3	2	1	1	1	1	2	2	2	1
Quantidade do componente presente no kit	2	1	1	1	1	1	2	1	1	1	2	1	2	1	2
Componente	16	17	18	19	20	21	22	23	24	25	26	27	28	29	
Nº de Exemplos	2	4	3	3	3	3	2	2	3	2	2	2	2	3	
Quantidade do componente presente no kit	1	1	1	1	1	1	1	1	1	2	2	1	1	1	

Fonte: Elaborado pelo autor.

Alguns componentes não apresentam notória diferença quando observados em ângulos distintos. Entretanto, outros são bem diferentes dependendo do ângulo que de visualização. Desta forma, o número de imagens de referência para cada componente depende do quão diferente o mesmo se faz com relação à câmera, ao ter sua posição (x e y) modificada sob o tabuleiro. Nesse sentido, um esforço foi feito na tentativa de minimizar o número de imagens de referência, visando deixar apenas aquelas necessárias para se obter uma boa detecção. A Figura 22 mostra um exemplo de um componente sobre diferentes ângulos.

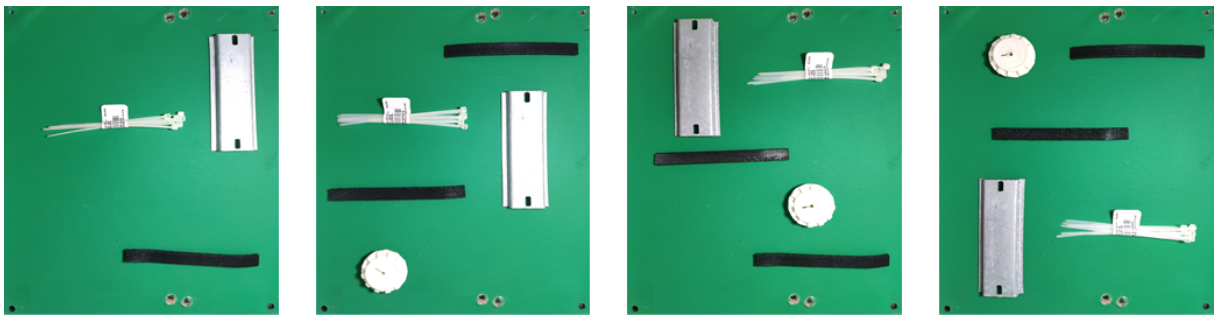
Figura 22 – Exemplo de um componente sobre diferentes ângulos.

Fonte: Elaborado pelo autor.

Foram obtidas diversas fotos de cada kit onde cada foto contém uma distribuição aleatória dos componentes na mesa, para simular o que ocorre no ambiente fabril. Nenhuma foto com componentes errados ou com uma quantidade maior que a correta de cada componente foi

obtida, porém foram capturadas algumas fotos com componentes faltando. Essa estratégia foi adotada para que não fosse necessário misturar os componentes de cada kit[12], de forma que cada um apresentasse seu conjunto de componentes isoladamente. Alguns exemplos do kit A estão disponíveis na Figura 23, onde pode ser visto que alguns componentes apresentaram um brilho diferente, que dependeu da posição do mesmo sobre a câmera. Isso se em virtude do número limitado de pontos de luz.

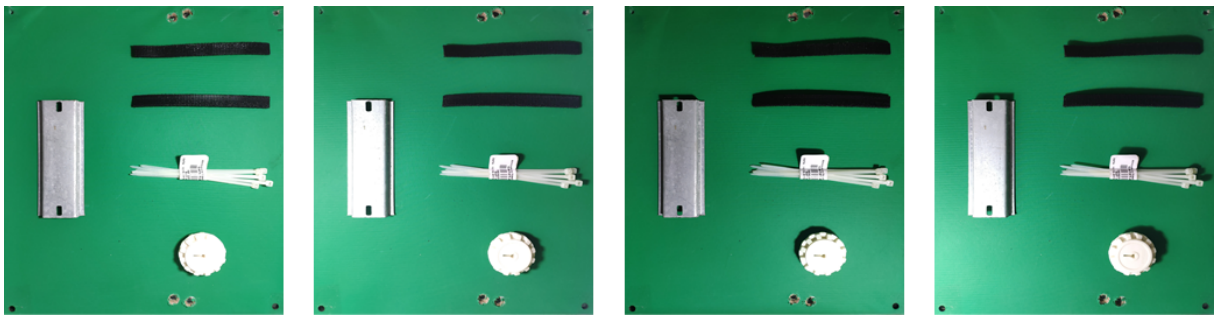
Figura 23 – Alguns exemplos do kit A.



Fonte: Elaborado pelo autor.

Além das fotos de cada kit, a fim de avaliar se o sistema sofre com variações de iluminação, foram também capturadas imagens submetidas a esse tipo de distúrbio, como ilustrado na Figura 24. Essas fotos foram realizadas buscando analisar, posteriormente, se ambas as soluções apresentariam complicações no momento de detectar os componentes para padrões de luz distintos.

Figura 24 – O kit A com o ponto de luz em diferentes posições.



Fonte: Elaborado pelo autor.

Por fim, foi realizada a anotação de cada exemplo de cada kit. O padrão de anotação adotado foi o YOLO, proposto por Redmon et al. (2015). O padrão de anotação YOLO descreve

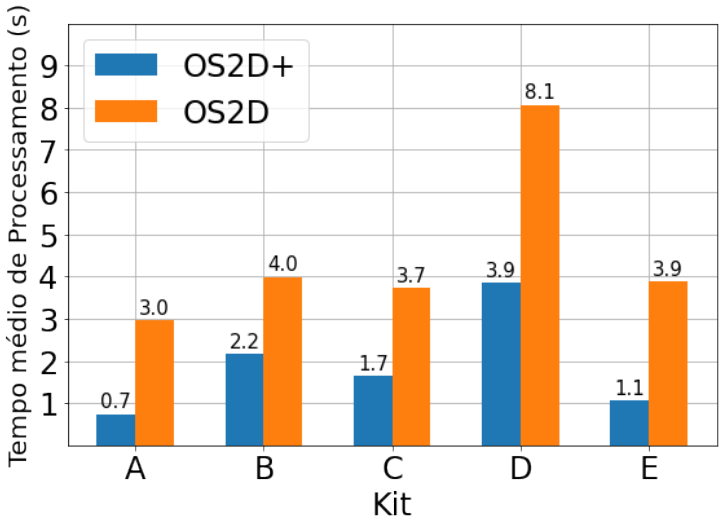
[12] Não é comum em ambientes fabris a mistura de componentes de kits diferentes, tendo em vista que os operadores somente tem contato com os componentes de um kit separadamente. O maior problema da montagem dos kits é a falta ou excesso de algum componente do kit sendo avaliado.

a classe e o BB de cada detecção por linha, separados por um espaço. A primeira informação apresentada é a classe, em seguida as coordenadas x e y do ponto superior esquerdo da detecção e depois a largura e altura da detecção. A classe é um número inteiro positivo maior que zero e as demais informações são números flutuantes entre zero e um.

4.2 Comparação dos modelos OS2D e OS2D+

Primeiramente, serão comparados os modelos OS2D+ e OS2D, afim de analisar se as modificações realizadas no mecanismo de inferência e as duas camadas adicionadas fizeram com que o modelo OS2D+ detectasse os componentes de forma mais precisa. Desta forma, os dois modelos serão submetidos às métricas de tempo médio de processamento e IoU médio, buscando analisar o tempo necessário que cada modelo leva para processar um *frame* e também a qualidade dos BBs detectados por cada modelo. Essa comparação se faz importante para avaliar se as modificações implementadas no modelo OS2D aprimoram o desempenho do mesmo em detectar novos objetos.

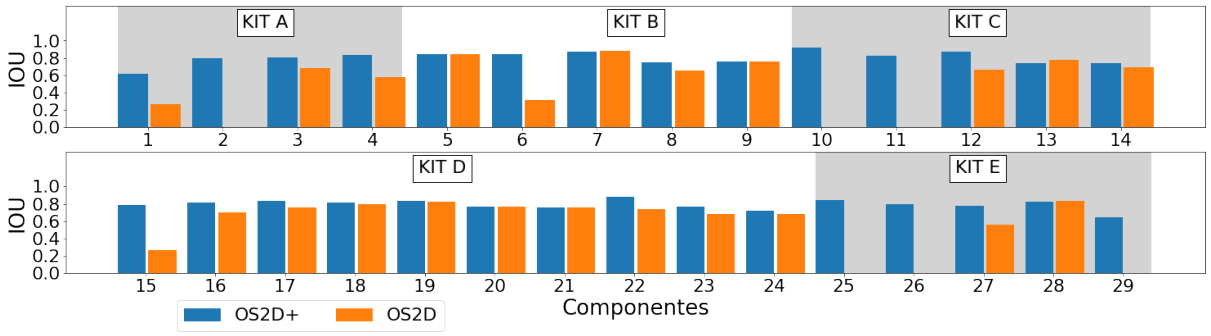
Figura 25 – Tempo médio de processamento de cada modelo em cada kit.



Fonte: Elaborado pelo autor.

Ao analisar o tempo médio de processamento dos dois modelos para cadas kit, percebe-se que o modelo OS2D+ obteve um resultado melhor em todos os kits, quando comparado com o modelo OS2D. Essa característica se deu devido ao fato da camada de distorção permitir a customização das resoluções de entrada dos componentes, otimizando o modelo para cada componente. Percebe-se que em kits com componentes maiores, como o kit A, o modelo OS2D+ desempenhou as detecções em um tempo médio menor que em kits com componentes menores, como o kit D. Os tempos médios alcançados pelos dois modelos para cada kit podem ser vistos na Figura 25.

Figura 26 – IoU médio de cada modelo para cada classe de componente.



Fonte: Elaborado pelo autor.

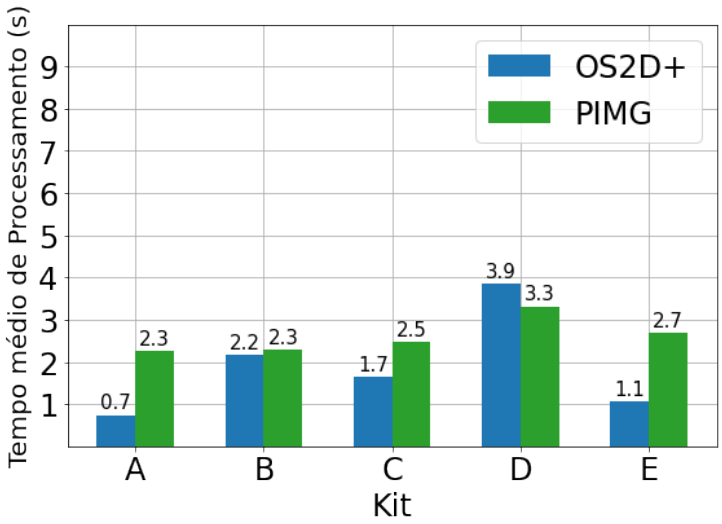
Outra métrica analisada foi o IoU médio para os modelos OS2D e OS2D+, como pode ser visto na Figura 26. O modelos OS2D+, mais uma vez, apresentou-se superior ao modelo OS2D. É possível perceber que nos componentes de *aspect ratio* mais diferentes foram os que o modelo OS2D não detectou corretamente, como os componentes 1, 10, 11, 25, 26 e 29. Isso se deu devido a limitação imposta pelas âncoras treinadas no modelo OS2D. As novas camadas adicionadas no modelo OS2D+ conseguiram melhorar o desempenho do modelo original.

A partir da análise do tempo médio necessário para processar um *frame* e o IoU médio obtido pelos modelos OS2D e OS2D+, percebe-se que as modificações implementadas no modelo OS2D+ permitiram que o mesmo alcançasse resultados melhores que o modelo original (OS2D). Isso se deu pela capacidade de customização fornecida pelas camadas de distorção e correção, que permitem uma detecção melhorada do objeto de interesse e, possivelmente, uma melhora no tempo de processamento, caso o componente em questão necessite de menos resolução de entrada para ser detectado.

4.3 Comparação dos resultados das soluções PIMG e OS2D+

Com a execução das duas soluções sobre a base de dados construída, foram obtidos os arquivos referentes a cada kit, descrevendo as detecções encontradas. Para facilitar o processo, cada objeto detectado foi descrito utilizando do padrão YOLO, o mesmo utilizado na anotação da base de dados. A primeira métrica analisada é o tempo médio de processamento de cada modelo em cada kit.

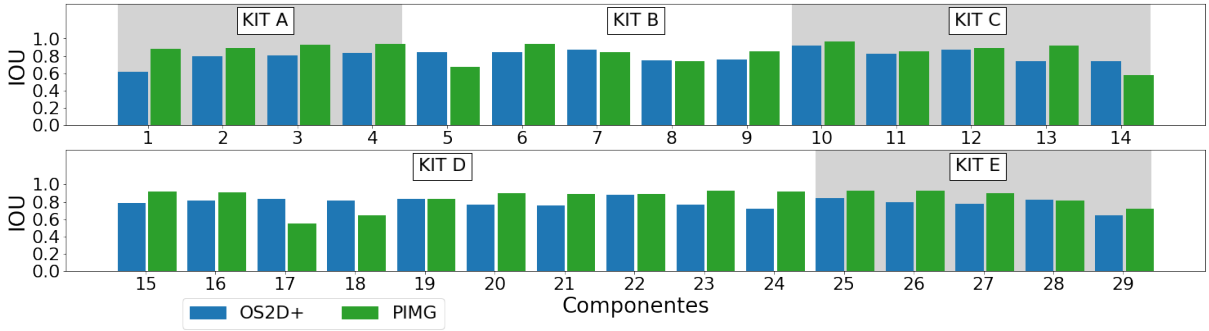
Figura 27 – Tempo médio de processamento de cada modelo em cada kit.



Fonte: Elaborado pelo autor.

Ao analisar o tempo desempenhado pelas soluções OS2D+ e PIMG nos cinco kits, percebe-se que a solução OS2D+ se apresenta mais rápida nos kits A, B, C e E, perdendo para a PIMG apenas no kit D. A solução OS2D+ não apresentou um tempo médio menor que a solução PIMG no kit D devido ao fato dos componentes do mesmo terem um tamanho médio pequeno, necessitando dessa forma de uma resolução maior para detectá-los. É possível observar o tempo médio desempenhado pelas duas soluções na Figura 27.

Figura 28 – IoU médio de cada modelo para cada classe de componente.

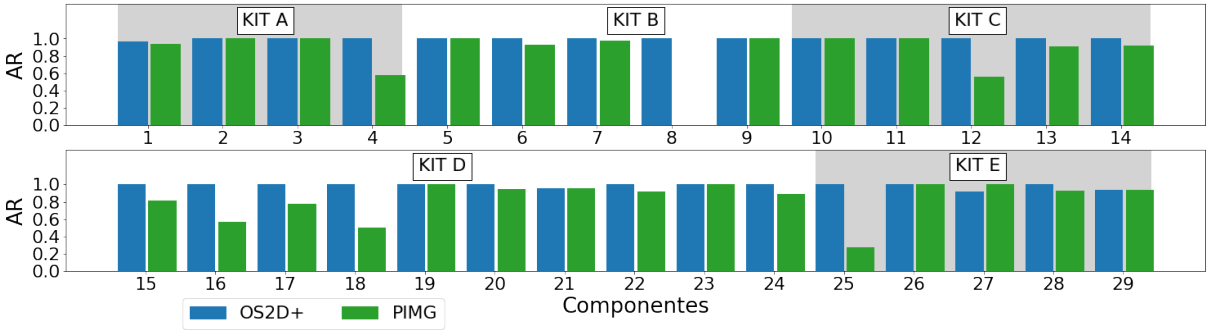


Fonte: Elaborado pelo autor.

Ao analisar o IoU médio de cada modelo para cada classe de componente, que pode ser visto na Figura 28, é possível observar que a solução PIMG foi capaz de captar melhor os BBs dos componentes quando comparada com a solução OS2D+. Uma característica que permitiu essa melhor detecção por parte da solução PIMG foi o fato da mesma utilizar de um mecanismo de segmentação por cor para extrair os contornos dos componentes. Esse mecanismo consegue, de forma precisa, capturar os formatos dos objetos, entretanto, acaba permitindo

também a detecção de falsos positivos devido a mudanças na iluminação ou avarias no tabuleiro. Apesar das fragilidades impostas pelo mecanismo de segmentação por cor presente no modelo PIMG, o mesmo alcançou um IoU médio na maioria dos componentes dos cinco kits.

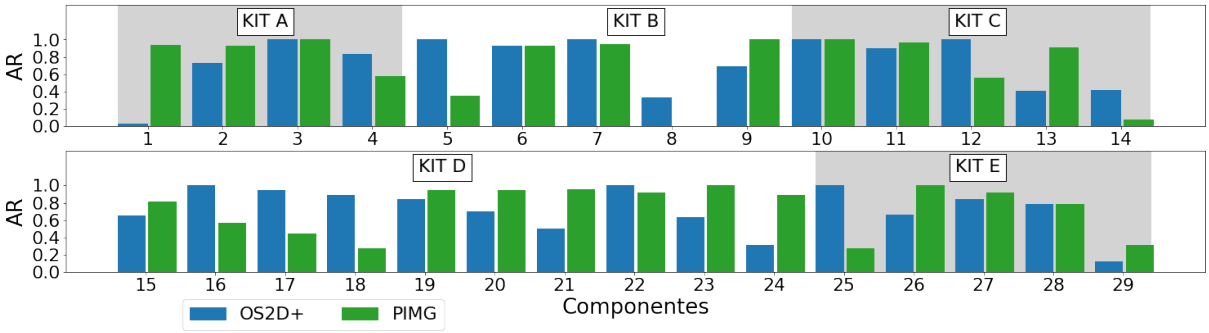
Figura 29 – AR@0.5 médio de cada modelo para cada classe de componente.



Fonte: Elaborado pelo autor.

Na Figura 29 estão dispostos os cálculos dos AR utilizando o filtro de IoU de 50% para as soluções PIMG e OS2D+. Para todos os componentes de todos os kits a solução OS2D+ obteve um AR médio superior ou equivalente ao obtido pela solução PIMG. Isso descreve a robustez do modelo OS2D+ a falsos negativos, de forma que o mesmo consegue recuperar apenas os objetos corretos na imagem, na grande maioria das vezes. Em contrapartida, o modelo PIMG cometeu muitos falsos negativos, inclusive deixando de detectar o componente de classe 8.

Figura 30 – AR@0.75 médio de cada modelo para cada classe de componente.

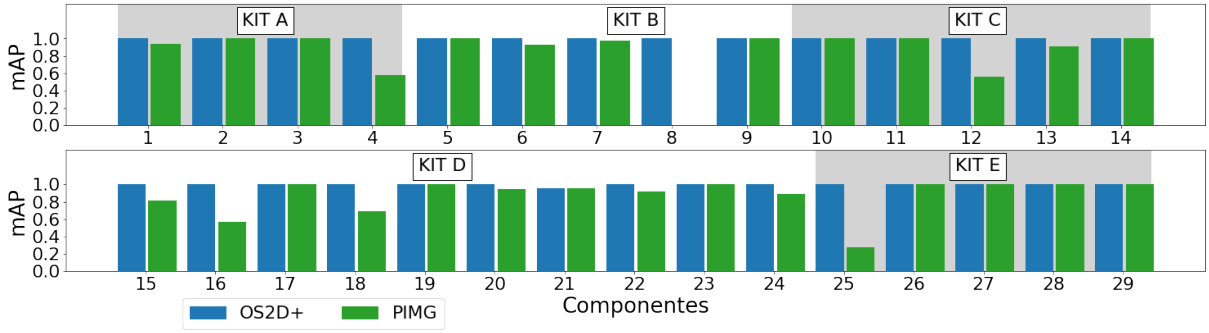


Fonte: Elaborado pelo autor.

Além de analisar o AR médio utilizando o filtro de IoU de 50%, também analisou-se o AR médio para todos os componentes de todos os kits utilizando um filtro de IoU de 75%, como pode ser visto na Figura 30. É possível perceber que o desempenho do modelo OS2D+ sobre a métrica foi prejudicado devido ao filtro de IoU de 75%, isso se deu devido ao fato do mesmo não conseguir estimar um BB tão próximo do *groundtruth* quanto o modelo PIMG, que

não teve seu desempenho tão alterado pelo filtro.

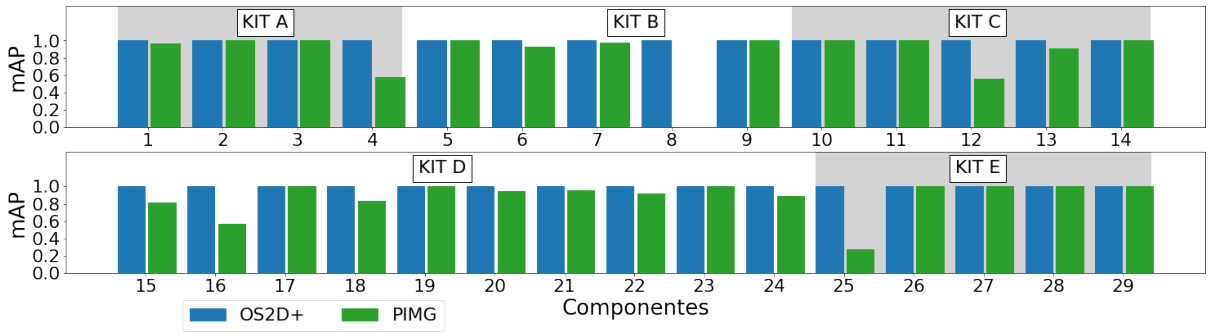
Figura 31 – mAP@0.5 médio de cada modelo para cada classe de componente.



Fonte: Elaborado pelo autor.

É possível observar na Figura 31 o cálculo da métrica mAP médio para todos os componentes de todos os kits, utilizando um filtro de IoU de 50%. É possível perceber que o modelo OS2D+ apresenta um mAP alto para todos os componentes, o que descreve uma robustez a falsos positivos inerente do modelo. Já o modelo PIMG apresenta, em alguns componentes, a detecção de alguns falsos positivos, tornando-o menos robusto que o modelo OS2D+.

Figura 32 – mAP@0.75 médio de cada modelo para cada classe de componente.



Fonte: Elaborado pelo autor.

Ao analisar o mAP filtrado com 75% para todos os componentes de todos os kits para as soluções OS2D+ e PIMG, como ilustrado na Figura 32, percebe-se que os resultados para o modelo OS2D+ sofreram poucas alterações, tendo em vista que a métrica mAP analisa apenas a existência de falsos positivos. Já o filtro de 75% não provocou falsos positivos no modelo, pelo contrário, provocou falsos negativos. O resultado do modelo PIMG também sofreu poucas alterações, quando comparado com os resultados obtidos com o filtro de 50% de IoU.

Tabela 2 – Média e desvio padrão das métricas calculadas para os modelos PIMG e OS2D+.

Kit	A	B	C	D	E
IOU : OS2D+	0,70 ± 0,06	0,83 ± 0,04	0,82 ± 0,06	0,79 ± 0,03	0,78 ± 0,07
IOU : PIMG	0,90 ± 0,04	0,81 ± 0,07	0,88 ± 0,04	0,85 ± 0,08	0,87 ± 0,08
AR@0.5 : OS2D+	0,99 ± 0,05	1,00 ± 0,00	1,00 ± 0,00	1,00 ± 0,02	0,95 ± 0,21
AR@0.5 : PIMG	0,91 ± 0,13	0,85 ± 0,11	0,92 ± 0,11	0,87 ± 0,13	0,72 ± 0,29
AR@0.75 : OS2D+	0,39 ± 0,24	0,87 ± 0,16	0,74 ± 0,27	0,73 ± 0,23	0,72 ± 0,25
AR@0.75 : PIMG	0,89 ± 0,16	0,68 ± 0,21	0,84 ± 0,18	0,81 ± 0,15	0,57 ± 0,29
mAP@0.5 : OS2D+	1,00 ± 0,00	1,00 ± 0,00	1,00 ± 0,00	1,00 ± 0,02	0,95 ± 0,21
mAP@0.5 : PIMG	0,91 ± 0,13	0,97 ± 0,08	0,92 ± 0,11	0,90 ± 0,14	0,73 ± 0,29
mAP@0.75 : OS2D+	0,78 ± 0,42	1,00 ± 0,00	0,95 ± 0,21	0,97 ± 0,18	0,95 ± 0,21
mAP@0.75 : PIMG	0,92 ± 0,11	0,93 ± 0,22	0,91 ± 0,13	0,91 ± 0,13	0,67 ± 0,32

Fonte: Elaborado pelo autor.

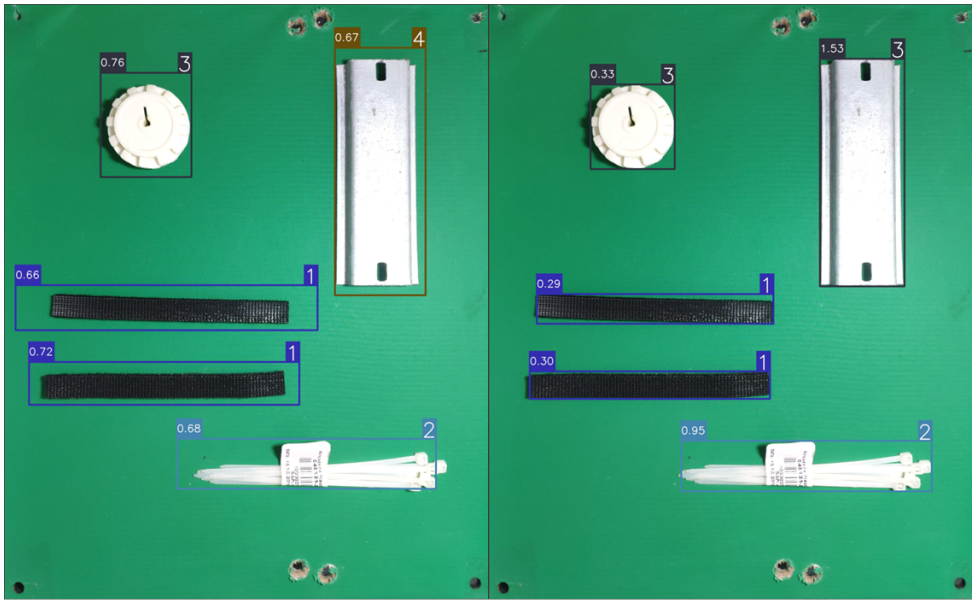
Primeiramente, foram analisados os resultados das métricas para cada classe de componente individualmente. Entretanto, é possível realizar a mesma análise para cada kit de componente, como pode ser visto na Tabela 2. É possível perceber que os resultados para a média das métricas para cada kit foi semelhante ao obtido para cada classe de componente, de forma que a solução PIMG superou a solução OS2D+ nas métricas IoU médio e AR@0.75 médio, entretanto, em todas as outras métricas a solução OS2D+ se mostrou superior a PIMG. A solução PIMG foi capaz de estimar melhores BB que a solução OS2D+, quando comparadas com o *groundtruth* anotado, permitindo que a mesma alcançasse melhores resultados na métrica IoU. O filtro de 75% de IoU presente na métrica AR@0.75 levou a solução OS2D+ a cometer alguns falsos negativos, o que acabou por diminuir a performance alcançada pela mesma na métrica AR@0.5. Apesar disso, a solução OS2D+ se mostrou mais robusta, cometendo menos falsos positivos que a PIMG, sendo esse um critério decisivo em uma solução industrial.

4.3.1 Análise visual entre os resultados das soluções para alguns exemplos

Nesta seção serão apresentados alguns exemplos com todos os seus componentes detectados pelas duas soluções, na tentativa de evidenciar ainda mais os pontos em que as mesmas se diferenciam.

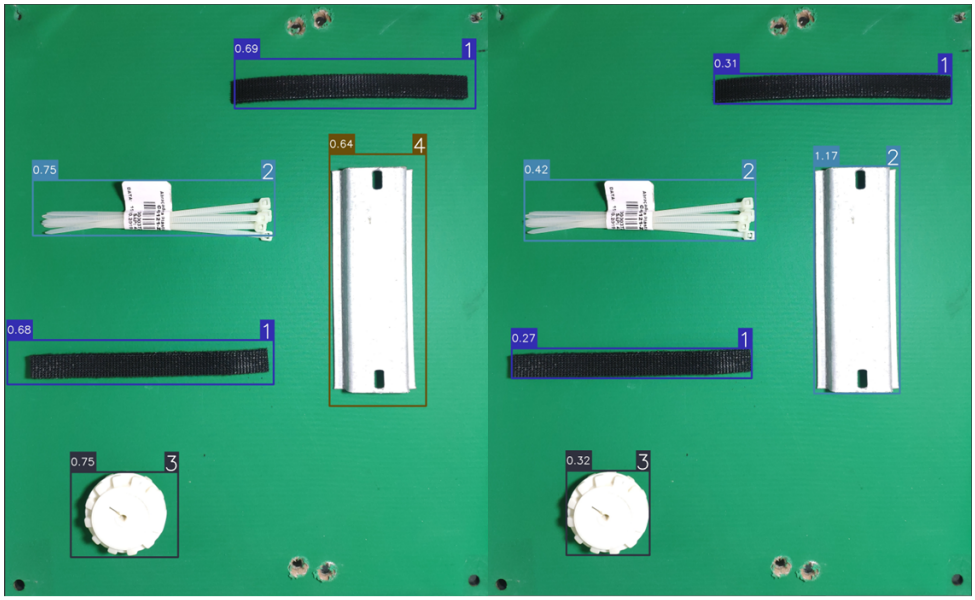
As Figuras 33, 34 e 35 exibem os resultados das duas soluções para os exemplos 6, 7 e 12 do kit A, respectivamente. Nas três figuras, percebe-se que a solução de OS2D+ não cometeu nenhum falso positivo ou negativo, diferentemente da solução PIMG que acabou por confundir o componente de classe quatro pelas classes três, dois e três, respectivamente. Entretanto, percebe-se também que os BBs da solução PIMG são bem mais ajustados aos componentes que a solução OS2D+.

Figura 33 – Comparação das detecções das duas soluções para o exemplo 6 do kit A.



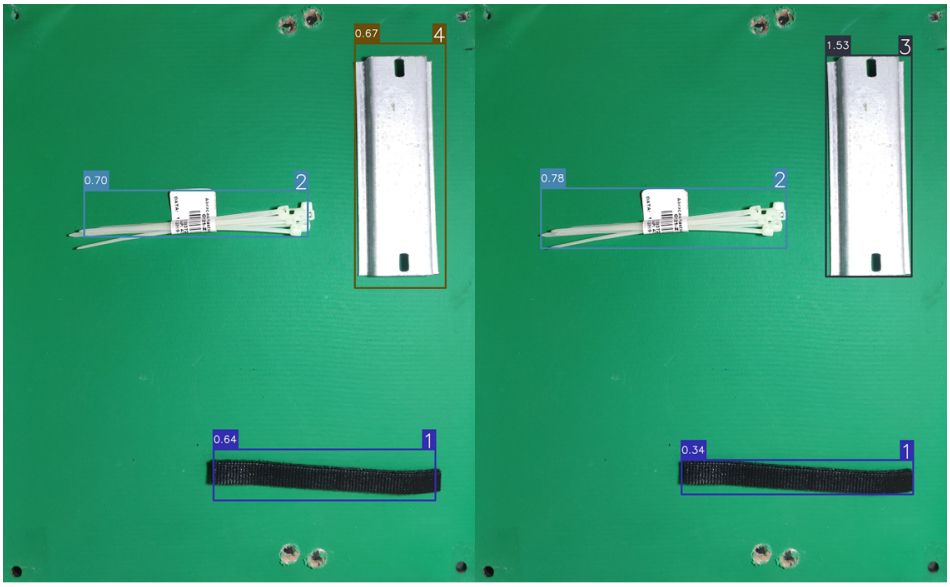
Fonte: Elaborado pelo autor.

Figura 34 – Comparação das detecções das duas soluções para o exemplo 7 do kit A.



Fonte: Elaborado pelo autor.

Figura 35 – Comparação das detecções das duas soluções para o exemplo 12 do kit A.

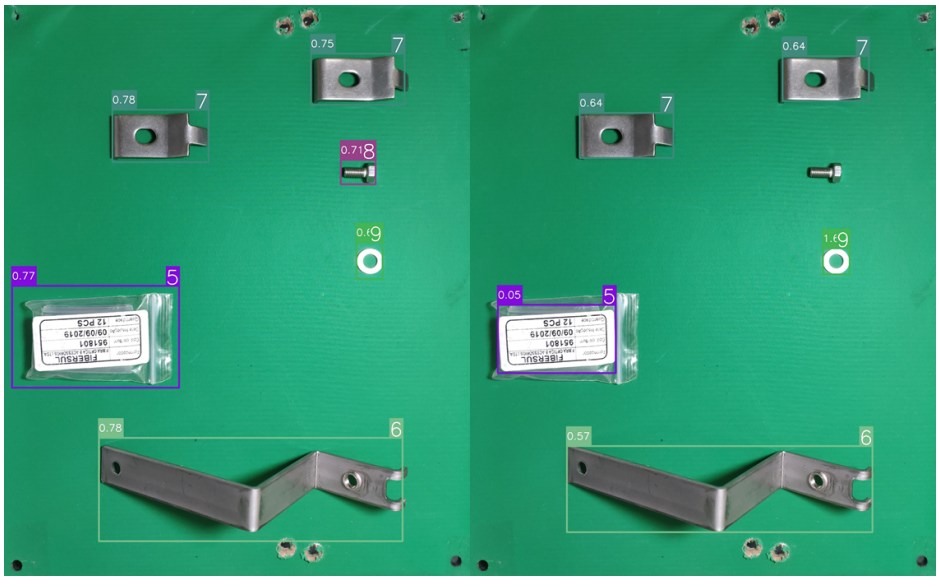


OS2D+ PIMG

Fonte: Elaborado pelo autor.

As Figuras 36, 37 e 38 contém as detecções das duas soluções para os exemplos 7, 8 e 19 do kit B, respectivamente. Novamente, a solução OS2D+ acertou as classes de todos os componentes corretamente e apresentou uma estimação de BBs mais precisa que nos exemplos exibidos do kit A. Já a solução PIMG não foi capaz de detectar o componente de classe 8 em nenhum dos exemplos do kit B e também se equivocou ao atribuir a classe de um componente na Figura 38, a qual deveria ser a classe seis.

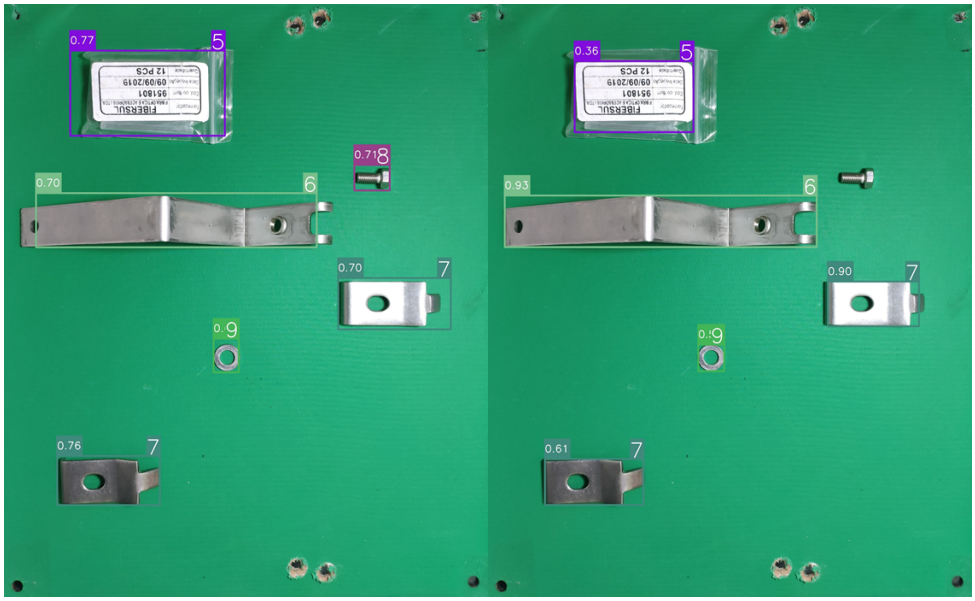
Figura 36 – Comparação das detecções das duas soluções para o exemplo 7 do kit B.



OS2D+ PIMG

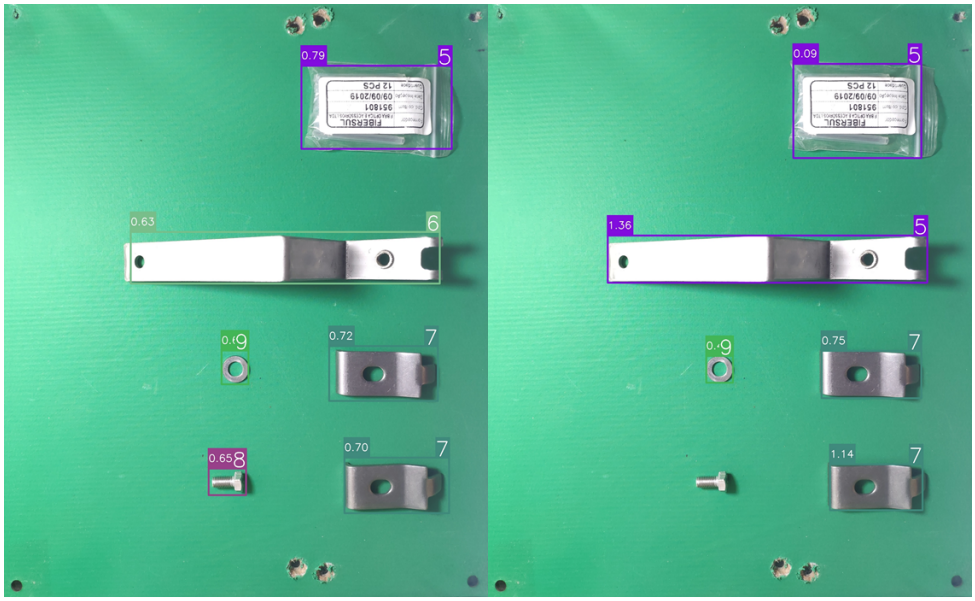
Fonte: Elaborado pelo autor.

Figura 37 – Comparação das detecções das duas soluções para o exemplo 8 do kit B.



Fonte: Elaborado pelo autor.

Figura 38 – Comparação das detecções das duas soluções para o exemplo 19 do kit B.

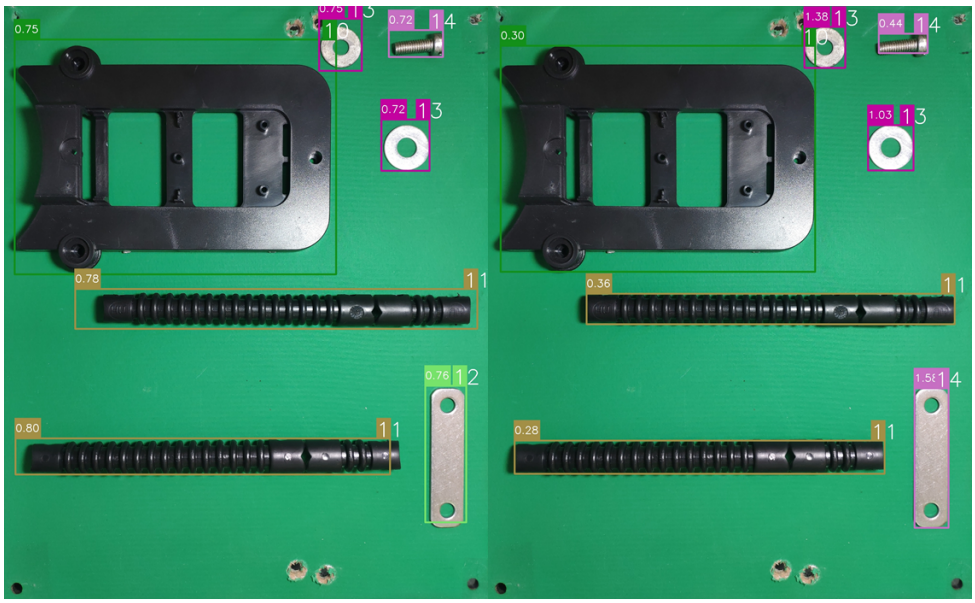


Fonte: Elaborado pelo autor.

As Figuras 39, 40 e 41 contém as detecções das duas soluções para os exemplos 10, 11 e 18 do kit C, respectivamente. Comparando com a performance nos exemplos do kit A, a solução OS2D+ foi capaz de resgatar corretamente a classe de todos os componentes dos exemplos do kit C e também apresentou BBs mais precisos. Já a solução PIMG se equivocou ao definir a classe do componente de classe 12, a qual ela atribuiu como 14 e 13 nas Figuras 39 e 40,

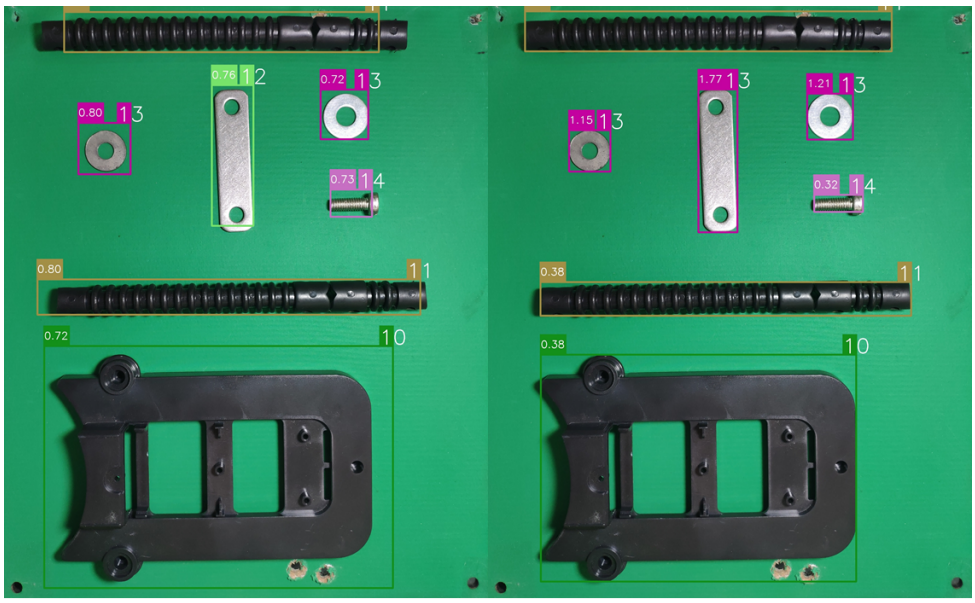
respectivamente. Além disso, ela também atribuiu incorretamente a classe 14 a um componente de classe 13, na Figura 41.

Figura 39 – Comparação das detecções das duas soluções para o exemplo 10 do kit C.



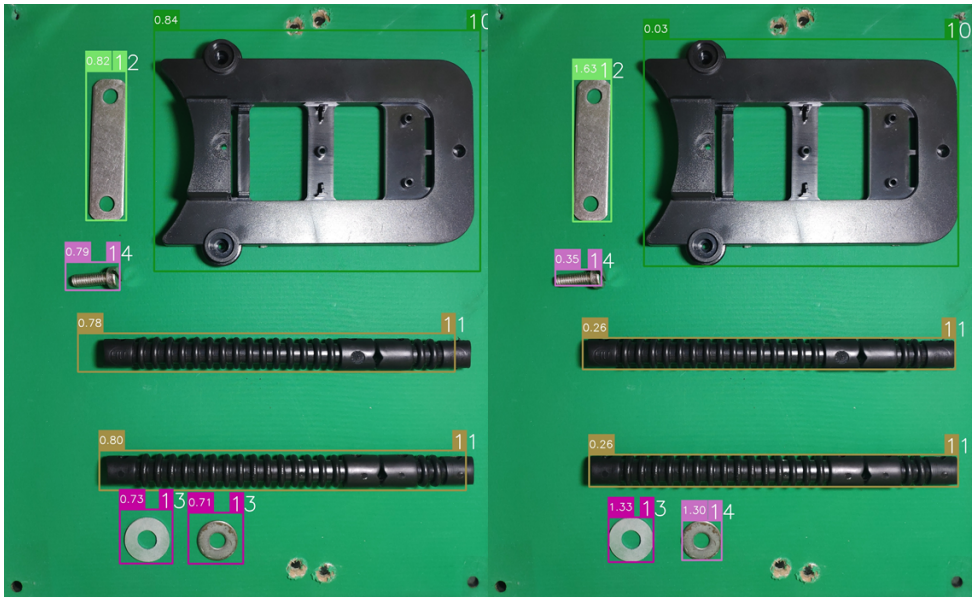
Fonte: Elaborado pelo autor.

Figura 40 – Comparação das detecções das duas soluções para o exemplo 11 do kit C.



Fonte: Elaborado pelo autor.

Figura 41 – Comparação das detecções das duas soluções para o exemplo 18 do kit C.

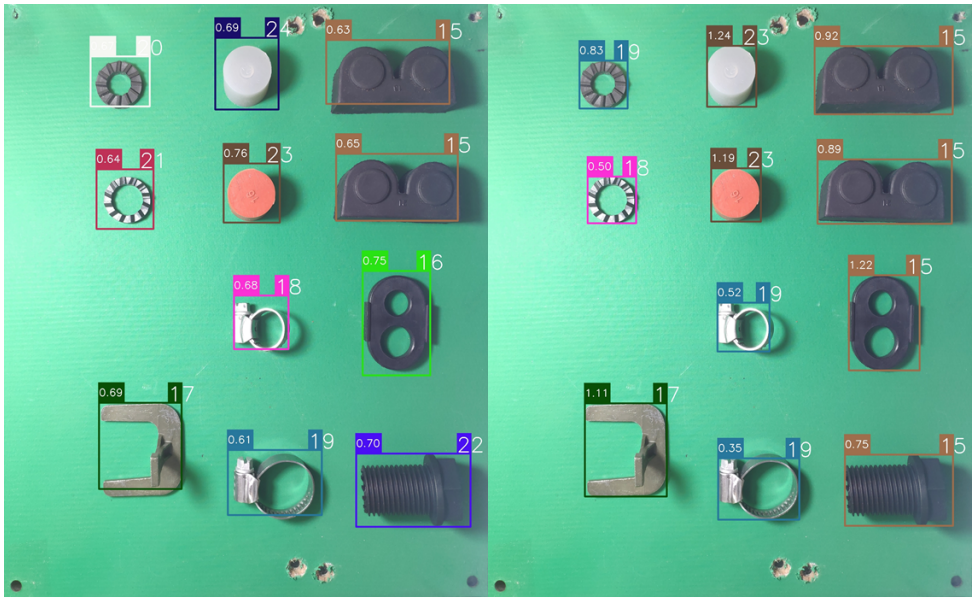


OS2D+ PIMG

Fonte: Elaborado pelo autor.

As Figuras 42, 43 e 44 mostram os resultados das duas soluções para os exemplos 27, 17 e 16 do kit D, respectivamente. A solução PIMG comete alguns erros nas atribuições das classes corretas nas três figuras, apesar de conseguir estimar bem os BBs. Já a solução OS2D+ acertou todas as classes dos componentes para os exemplos expostos do kit D.

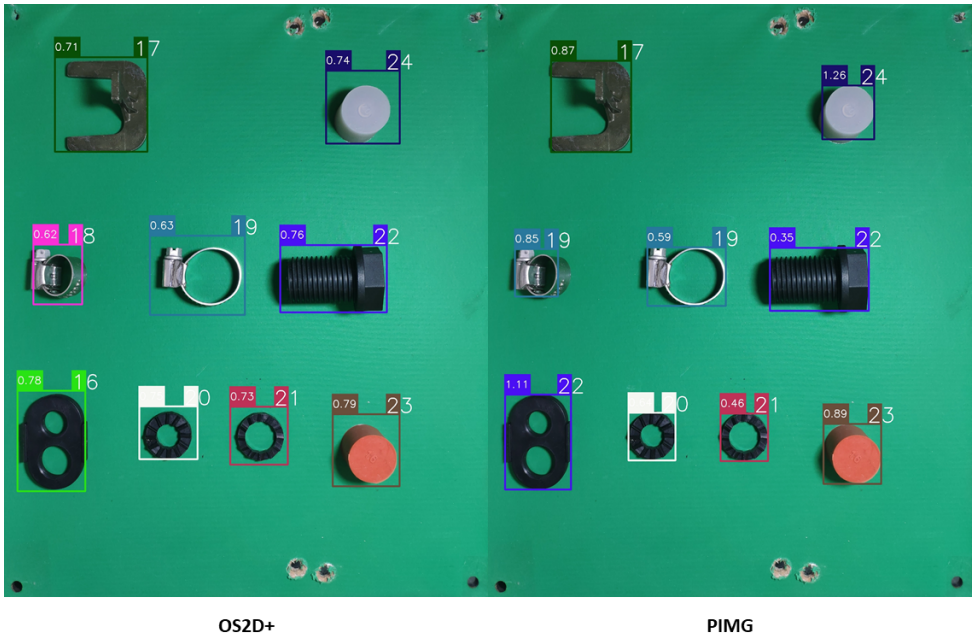
Figura 42 – Comparação das detecções das duas soluções para o exemplo 27 do kit D.



OS2D+ PIMG

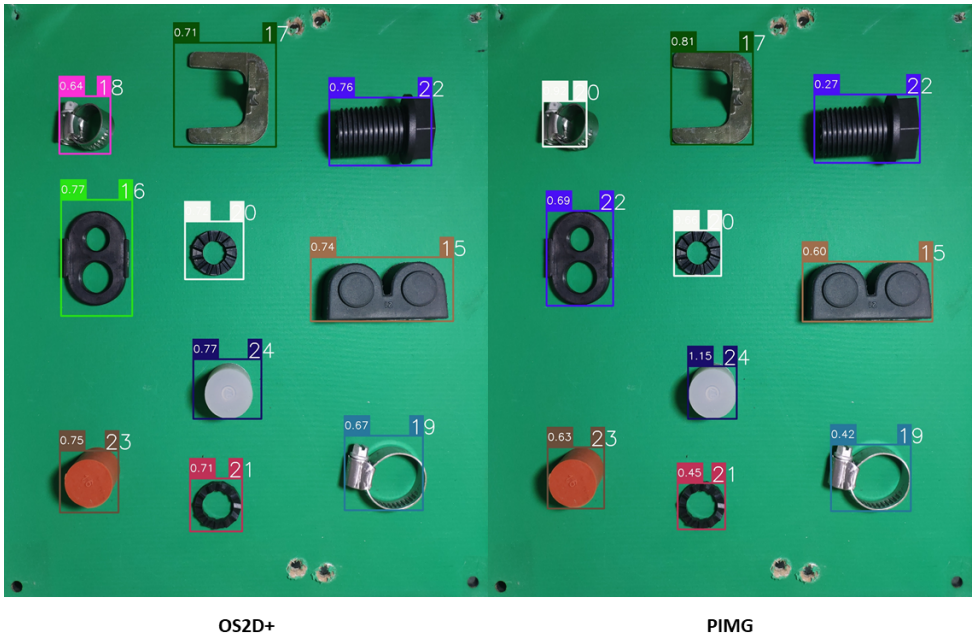
Fonte: Elaborado pelo autor.

Figura 43 – Comparação das detecções das duas soluções para o exemplo 17 do kit D.



Fonte: Elaborado pelo autor.

Figura 44 – Comparação das detecções das duas soluções para o exemplo 16 do kit D.

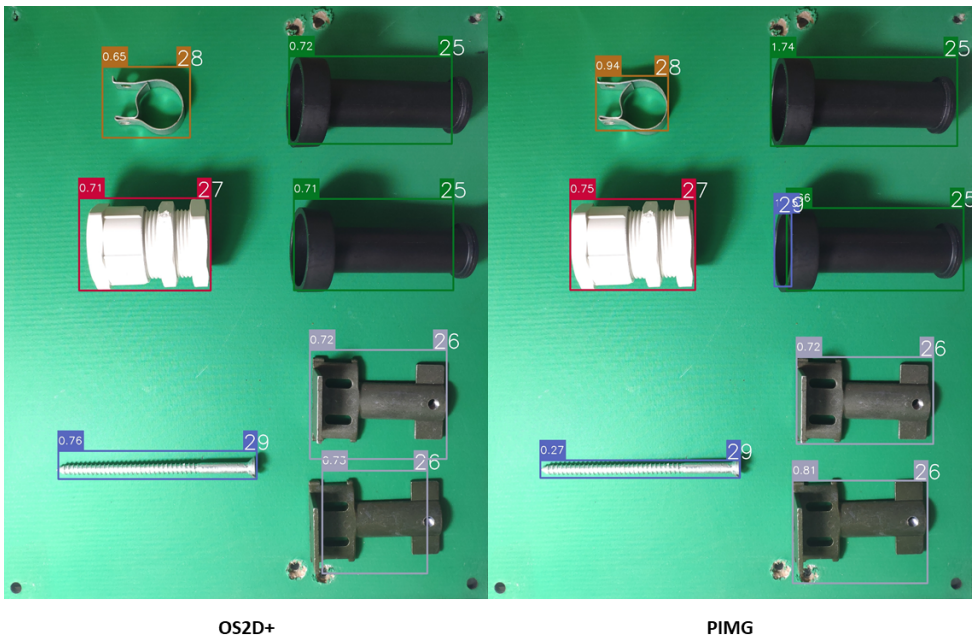


Fonte: Elaborado pelo autor.

As Figuras 45, 46 e 47 mostram os resultados das duas soluções para os exemplos 22, 21 e 19 do kit E, respectivamente. No kit E alguns objetos que não existiam foram detectados, como é possível de se ver nas Figuras 45 e 46. Isso se dá devido ao processo de segmentação por cor. Além de detectar equivocadamente alguns componentes inexistentes, a solução PIMG também errou a classe de alguns componentes. A solução OS2D+, da mesma forma que nos

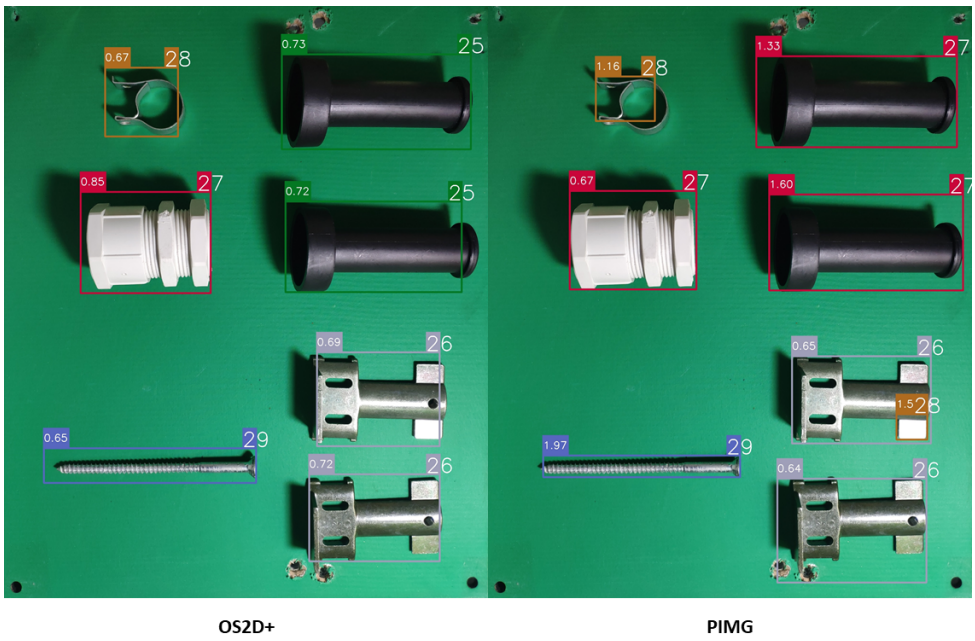
outros exemplos dos outros kits, acertou corretamente as classes dos objetos dispostos nos exemplos exibidos.

Figura 45 – Comparação das detecções das duas soluções para o exemplo 22 do kit E.



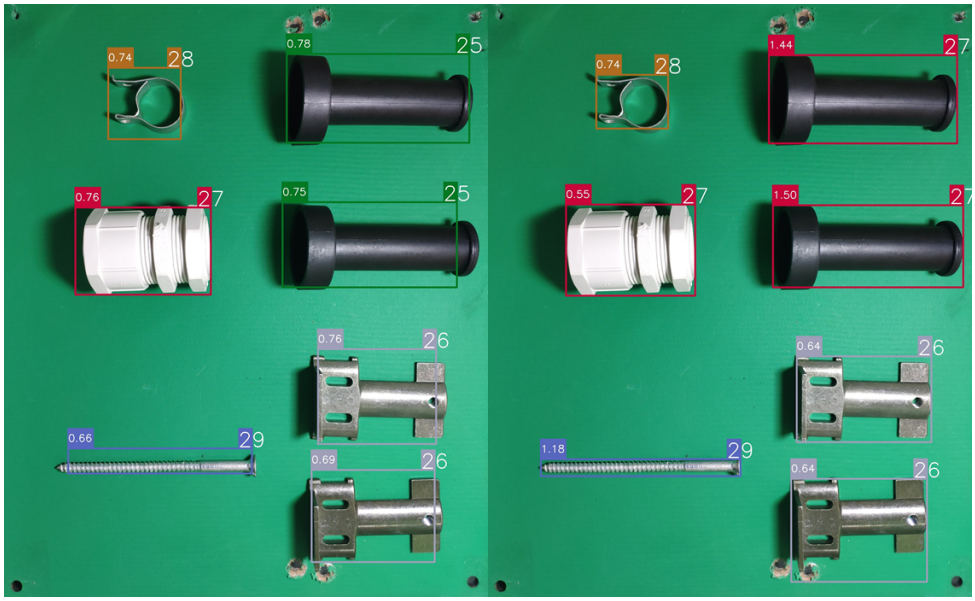
Fonte: Elaborado pelo autor.

Figura 46 – Comparação das detecções das duas soluções para o exemplo 21 do kit E.



Fonte: Elaborado pelo autor.

Figura 47 – Comparação das detecções das duas soluções para o exemplo 19 do kit E.



Fonte: Elaborado pelo autor.

Após observar as detecções das duas soluções para alguns exemplos dos cinco kits, obtém-se um compromisso (trade-off) entre as duas soluções. A solução OS2D+ fornece robustez na detecção das classes, porém apresenta BBs menos precisos. Já a solução PIMG apresenta muitos equívocos no momento de detectar as classes dos componentes, uma vez que, algumas vezes encontrou componentes inexistentes. Entretanto, foi capaz de estimar BBs mais bem ajustados. Apesar da segmentação fornecer BBs bem precisos, ela também acaba por injetar muitas fragilidades à solução PIMG, tendo em vista que quaisquer avarias no tabuleiro verde poderão ser detectados como possíveis componentes.

5 CONCLUSÃO

Após obter os resultados para as duas soluções propostas, foi possível perceber que ambas têm seus pontos fortes e fracos. A solução PIMG apresentou uma boa capacidade de estimar os BBs das detecções, entretanto, foi mais suscetível a cometer falsos positivos. Em contrapartida, a solução OS2D+ apresentou-se mais robusta, com poucos falsos positivos e falsos negativos, embora não tenha sido capaz de estimar um BB tão próximo do *groundtruth*, como a solução PIMG.

Apesar dos pontos fortes e fracos de cada solução, ao analisar o modo de construção e operação da solução OS2D+, percebeu-se que a mesma tem parâmetros que podem ser ajustados de forma a melhorar os resultados obtidos. Para este caso, é possível aumentar a resolução de entrada das imagens, na tentativa de obter BBs mais bem ajustados, ou aumentar o número de exemplos de componentes, entre outras possíveis ideias para melhorar a performance da solução OS2D+. Entretanto, a solução PIMG apresenta fragilidades em sua construção, que podem vir a piorar sua performance posteriormente, como um possível desgaste da superfície do tabuleiro, o que ocasionaria a detecção de falsos positivos. Outro possível problema seria a variação da iluminação, como pode ser visto nas Figuras 33, 34 e 35, que provocaria uma atribuição errada, por parte da solução PIMG, para o componente detectado.

Portanto, é possível observar que as fragilidades da solução PIMG fazem com que a mesma apresente-se pouco robusta para ser um solução industrial, em contrapartida a solução OS2D+ apresentou-se mais robusta, sendo uma forte candidata a resolver o problema de inspeção de kits de componentes. Isso, levando em consideração que a precisão de localização dos BBs das detecções não é tão necessária para um solução industrial quanto à capacidade de evitar a detecção incorreta de componentes.

Ao longo do desenvolvimento do presente trabalho, além da aplicação do modelo OS2D proposto por Osokin, Sumin e Lomakin (2020) para a resolução do problema de inspeção de kits industriais, foram desenvolvidas as camadas de distorção e correção, uma de pré-processamento e a outra de pós-processamento. Elas foram elaboradas visando tornar o modelo OS2D menos dependente de âncoras, permitindo assim a detecção de objetos de diferentes *aspect ratios*, o que antes era limitado a um número pré-definido de âncoras, que não se faziam capazes de detectar todos os tipos de objetos. Essa contribuição se faz muito relevante no âmbito do FSOD, tendo em vista que os modelos que independem de retreinamento se propõem a serem capazes de detectar todos os objetos, cujos exemplos forem fornecidos.

Outra modificação realizada no modelo OS2D, proposto por Osokin, Sumin e Lomakin (2020), foi uma reformulação do mecanismo de inferência do modelo. Essa modificação removeu uma limitação do modelo OS2D, que não permitia um número maior de imagens de referência para cada componente, acabando por estourar a memória disponível na placa

gráfica. Com o novo mecanismo de inferência, o modelo OS2D realiza a inferência de cada componente do kit separadamente, sendo no final os resultados agregados em uma só lista. Devido a modificação no mecanismo de inferências e as camadas de distorção e correção que o modelo OS2D se fez capaz de resolver o problema de inspeção de kits de componentes.

REFERÊNCIAS

BENNEQUIN, E. Meta-learning algorithms for few-shot computer vision. *arXiv preprint arXiv:1909.13579*, 2019. Citado na página 3.

CANNY, J. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, Ieee, n. 6, p. 679–698, 1986. Citado na página 25.

CHOLLET, F. *Deep Learning with Python*. [S.l.]: Manning Publications Company, 2017. ISBN 9781617294433. Citado 2 vezes nas páginas 2 e 7.

CS231N, S. Convolutional neural networks for visual recognition. *URL: http://cs231n. stanford. edu*, 2017. Citado na página 11.

DROGHINI, D. et al. Few-shot siamese neural networks employing audio features for human-fall detection. In: *Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence*. [S.l.: s.n.], 2018. p. 63–69. Citado na página 19.

FAN, Q. et al. Few-shot object detection with attention-rpn and multi-relation detector. In: *CVPR*. [S.l.: s.n.], 2020. Citado na página 3.

FU, K. et al. Meta-ssd: Towards fast adaptation for few-shot object detection with meta-learning. *IEEE Access*, IEEE, v. 7, p. 77597–77606, 2019. Citado 2 vezes nas páginas 3 e 19.

GIRSHICK, R. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 1440–1448. Citado 2 vezes nas páginas 10 e 11.

GIRSHICK, R. et al. 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. Citado na página 10.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. (Adaptive Computation and Machine Learning series). ISBN 9780262035613. Citado 2 vezes nas páginas 2 e 6.

HE, K. et al. *Mask R-CNN*. 2017. Citado na página 20.

HE, K. et al. 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. Citado 2 vezes nas páginas 8 e 9.

HE, K. et al. Identity mappings in deep residual networks. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 630–645. Citado 2 vezes nas páginas 8 e 9.

HOSPEDALES, T. et al. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020. Citado na página 14.

HSIEH, T.-I. et al. One-shot object detection with co-attention and co-excitation. *arXiv preprint arXiv:1911.12529*, 2019. Citado 2 vezes nas páginas 3 e 19.

KANG, B. et al. Few-shot object detection via feature reweighting. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 8420–8429. Citado 2 vezes nas páginas 3 e 20.

KOCH, G.; ZEMEL, R.; SALAKHUTDINOV, R. Siamese neural networks for one-shot image recognition. In: LILLE. *ICML deep learning workshop*. [S.l.], 2015. v. 2. Citado 2 vezes nas páginas 14 e 15.

LI, X. et al. One-shot object detection without fine-tuning. *arXiv preprint arXiv:2005.03819*, 2020. Citado na página 20.

LI, Y. et al. Mm-fsod: Meta and metric integrated few-shot object detection. *arXiv preprint arXiv:2012.15159*, 2020. Citado na página 20.

LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 21–37. Citado 4 vezes nas páginas 3, 12, 13 e 26.

MICHAELIS, C.; BETHGE, M.; ECKER, A. One-shot segmentation in clutter. In: PMLR. *International Conference on Machine Learning*. [S.l.], 2018. p. 3549–3558. Citado 2 vezes nas páginas 3 e 19.

MICHAELIS, C.; BETHGE, M.; ECKER, A. S. Closing the generalization gap in one-shot object detection. *arXiv preprint arXiv:2011.04267*, 2020. Citado na página 21.

MICHAELIS, C. et al. One-shot instance segmentation. *arXiv preprint arXiv:1811.11507*, 2018. Citado na página 20.

MITCHELL, T. *Machine Learning*. [S.l.]: McGraw-Hill, 1997. (McGraw-Hill International Editions). ISBN 9780071154673. Citado na página 1.

OPENCV. *Contours : More Functions*. 2021. Disponível em <https://docs.opencv.org/master/d5/d45/tutorial_py_contours_more_functions.html>. Acesso em Março/2021. Citado na página 26.

OSOKIN, A.; SUMIN, D.; LOMAKIN, V. Os2d: One-stage one-shot object detection by matching anchor features. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2020. p. 635–652. Citado 9 vezes nas páginas 3, 4, 5, 6, 17, 18, 23, 26 e 50.

PEREZ-RUA, J.-M. et al. Incremental few-shot object detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2020. p. 13846–13855. Citado na página 21.

RAHMAN, S. et al. Any-shot object detection. In: *Proceedings of the Asian Conference on Computer Vision*. [S.l.: s.n.], 2020. Citado na página 21.

REDMON, J. et al. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. Disponível em: <http://arxiv.org/abs/1506.02640>. Citado na página 35.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 91–99. Citado 3 vezes nas páginas 3, 11 e 12.

RICH, E.; KNIGHT, K. *Artificial Intelligence*. [S.l.]: McGraw-Hill, 1991. (Artificial Intelligence Series). ISBN 9780071008945. Citado na página 1.

ROCCO, I.; ARANDJELOVIC, R.; SIVIC, J. Convolutional neural network architecture for geometric matching. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 6148–6157. Citado 5 vezes nas páginas 6, 15, 16, 17 e 18.

SHAPIRO, L.; STOCKMAN, G. *Computer Vision*. [S.l.]: Prentice Hall, 2001. ISBN 9780130307965. Citado na página 1.

SHI, B. et al. Few-shot acoustic event detection via meta learning. In: IEEE. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2020. p. 76–80. Citado na página 21.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado na página 7.

SONKA, M.; HLAVAC, V.; BOYLE, R. *Image Processing, Analysis, and Machine Vision*. [S.l.]: Cengage Learning, 2014. ISBN 9781133593607. Citado na página 1.

TSANG, S.-H. *Review: VGGNet — 1st Runner-Up (Image Classification), Winner (Localization) in ILSVRC 2014*. 2018. Disponível em <https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11>. Acesso em Junho/2020. Citado na página 8.

USTYUZHANINOV, I. et al. One-shot texture segmentation. *arXiv preprint arXiv:1807.02654*, 2018. Citado na página 19.

WANG, X. et al. Frustratingly simple few-shot object detection. *arXiv preprint arXiv:2003.06957*, 2020. Citado na página 21.

WANG, Y. et al. Few-shot sound event detection. In: IEEE. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. [S.l.], 2020. p. 81–85. Citado na página 21.

WANG, Y. et al. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, ACM New York, NY, USA, v. 53, n. 3, p. 1–34, 2020. Citado na página 14.

WANG, Y.-X.; RAMANAN, D.; HEBERT, M. Meta-learning to detect rare objects. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. [S.l.: s.n.], 2019. p. 9925–9934. Citado na página 20.

WINSTON, P. H. *Artificial Intelligence (3rd Ed.)*. USA: Addison-Wesley Longman Publishing Co., Inc., 1992. ISBN 0201533774. Citado na página 1.

WU, J. et al. Multi-scale positive sample refinement for few-shot object detection. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2020. p. 456–472. Citado na página 22.

WU, X.; SAHOO, D.; HOI, S. Meta-rcnn: Meta learning for few-shot object detection. In: *Proceedings of the 28th ACM International Conference on Multimedia*. [S.l.: s.n.], 2020. p. 1679–1687. Citado na página 21.

XIA, C. et al. Cg-bert: Conditional text generation with bert for generalized few-shot intent detection. *arXiv preprint arXiv:2004.01881*, 2020. Citado na página 22.