UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

VITOR BISI VIEIRA

DETECÇÃO DE BLOCOS DE ROCHAS NATURAIS EM AMBIENTE NÃO CONTROLADO

VITÓRIA
2022

VITOR BISI VIEIRA

DETECÇÃO DE BLOCOS DE ROCHAS NATURAIS EM AMBIENTE NÃO CONTROLADO

Parte manuscrita do Projeto de Graduação do aluno **Vitor Bisi Vieira**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Patrick Marques Ciarelli

VITÓRIA
2022

VITOR BISI VIEIRA

DETECÇÃO DE BLOCOS DE ROCHAS NATURAIS EM AMBIENTE NÃO CONTROLADO

Parte manuscrita do Projeto de Graduação do aluno **Vitor Bisi Vieira**, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovada em 16 de agosto de 2022.

COMISSÃO EXAMINADORA:

Prof. Dr. Patrick Marques Ciarelli
Universidade Federal do Espírito Santo
Orientador

Prof. Dra. Thaís Pedruzzi do Nascimento
Universidade Federal do Espírito Santo
Examinadora

Prof. Dr. Klaus Fabian Côco
Universidade Federal do Espírito Santo
Examinador

AGRADECIMENTOS

Agradeço aos meus pais, irmãos e família, que sempre me apoiaram, sustentaram e incentivaram durante todos estes anos curso. Agradeço a sua paciência e amor que nunca falharam, principalmente nos momentos mais difíceis e de maior necessidade.

Aos meus amigos e colegas de curso, pelo companheirismo, força e apoio. Às pessoas que estiveram ao meu lado não só nos momentos de descontração e riso, mas também nos dias duros e cansativos. Sem vocês essa jornada não teria sido tão especial e marcante.

Ao meu orientador Patrick Marques Ciarelli, por ter despertado meu interesse por esse tema fascinante e desafiador, tendo sido um grande guia e mentor durante todo o desenvolvimento deste trabalho. Agradeço também todo o apoio durante o curso de Engenharia Elétrica, eu não teria chegado até o fim deste curso sem a sua ajuda e aconselhamento.

À banca examinadora pela aceitação do convite e pelo tempo que investiram na leitura e avaliação deste trabalho.

Agradeço à Universidade Federal do Espírito Santo pela minha formação.

RESUMO

Sendo um produto de origem natural, o granito e o mármore se manifestam em cores e tipos variados, mostrando-se uma tarefa complexa o discernimento dos seus variados tipos na comercialização das rochas. Na indústria de granito, em específico, a falta de exatidão na especificação da aparência visual das rochas de granito leva a desentendimentos entre fornecedores e clientes, acarretando perdas econômicas consideráveis. A identificação dos diferentes tipos de granito é feita em grande parte por meio de inspeção visual por especialistas na área, sendo este um processo lento e subjetivo. Com a meta de aprimorar este processo, a indústria de rochas naturais tem demonstrado um interesse crescente em utilizar sistemas inteligentes para automatizar processos, tais como de classificação e de reidentificação dos tipos de granito e mármore. Uma das etapas fundamentais para cumprir este objetivo é a detecção de blocos de rochas naturais em ambientes não controlados, foco deste trabalho. Uma abordagem promissora no campo de detecção de objetos em imagens é a utilização de modelos de Redes Neurais Convolucionais (CNN, do inglês *Convolucional Neural Networks*). Algumas das soluções mais recentes utilizando redes neurais têm superado os métodos tradicionais em desempenho, com o adicional de terem menor sensibilidade a variações no ambiente, revelando um maior potencial para utilização em aplicações no mundo real. O objetivo deste trabalho é o desenvolvimento de uma aplicação baseada em CNN que efetue a detecção de blocos de rochas naturais em imagens obtidas em ambientes não controlados. Foram estudados detectores de objetos nas arquiteturas *You Only Look Once* (*YOLOv5*) e *Single Shot Multibox Detector* (SSD), que são baseados em modelos de CNN. Escolheu-se o modelo mais adequado através da análise e comparação das métricas padrões de avaliação do *dataset Common Objects in Context* (COCO), já bem estabelecidas no campo da visão computacional. Foi comprovado que ambos os modelos são capazes de realizar a detecção dos blocos com boa confiabilidade e precisão em ambientes não controlados, tendo sido alcançados valores de precisão média iguais a 0,9604 para a *SSD MobileNet* e 0,9693 para a *YOLOv5*. Tendo em vista o desempenho destes modelos, abre-se a possibilidade de desenvolvimento de aplicações futuras que realizem a classificação e reidentificação das rochas naturais em ambientes não controlados, empregando técnicas de extração de características nas imagens com menor interferência do fundo.

Palavras-chave: Detecção de rochas naturais. Rede Neural Convolucional. Ambiente não controlado.

ABSTRACT

Being products of natural origin, granite and marble come in a variety of colors and types, making it a complex task to discern the various types in the marketing of stone. In the granite industry in particular, inaccuracy in specifying the visual appearance of granite rocks leads to misunderstandings between suppliers and customers, resulting in considerable economic losses. Identification of the different types of granite is largely done by visual inspection by granite experts, being a slow and subjective process. In order to improve this process, the natural stone industry has shown a growing interest in using intelligent systems to automate processes such as grading and re-identification of granite and marble types. One of the key steps to accomplish this goal is the detection of natural rock blocks in uncontrolled environments, the focus of this work. A promising approach in the field of object detection in images is the use of Convolutional Neural Networks (CNN) models. Some of the most recent solutions using neural networks have outperformed traditional methods in performance, with the added bonus of having less sensitivity to variations in the environment, revealing a greater potential for use in real-world applications. The objective of this work is to develop a CNN-based application that performs detection of natural rock blocks in images taken in uncontrolled environments. Object detectors in the architectures You Only Look Once (YOLOv5) and Single Shot Multibox Detector (SSD), which are based on CNN models, were studied. The most suitable model was chosen by analyzing and comparing standard Common Objects in Context (COCO) dataset evaluation metrics, which are already well established in the computer vision field. It was proven that both models are able to perform the detection of blocks with good reliability and accuracy in uncontrolled environments, reaching average precision values equal to 0.9604 for SSD MobileNet and 0.9693 for YOLOv5. Given the performance of these models, it opens up the possibility of developing future applications that perform the classification and re-identification of natural rocks in uncontrolled environments, employing feature extraction techniques on images with less background interference.

Keywords: Natural rock detection. Convolutional Neural Network. Uncontrolled environment.

LISTA DE FIGURAS

Figura 1 – Redes neurais “não-profundas” (à esquerda) e redes profundas (à direita) 21
Figura 2 - Arquitetura de uma Rede Neural Convolucional.....22
Figura 3 - Geração de um mapa de características em camadas de convolução 23
Figura 4 - Comparativo entre *max pooling* e *average pooling*.....24
Figura 5 - Exemplo de imagem com *bounding boxes* 25
Figura 6 - Exemplo de algoritmo de dois estágios 26
Figura 7 - Células e *bounding boxes*.....29
Figura 8 - Exemplo de parâmetros dimensionais de uma *bounding box*.....29
Figura 9 - Arquitetura da rede *YOLOv5* 32
Figura 10 - Detecção com mapas de características de diferentes dimensões.....34
Figura 11 - Arquitetura de camadas da rede SSD.....34
Figura 12 - Ilustração de uma *bounding box* com *scores* de classe.....36
Figura 13 - Definição da IOU.....40
Figura 14 - Determinação das *bounding boxes*.....45
Figura 15 - Exemplos de aplicação de operações de aumento de dados 47
Figura 16 - Congelamento de camadas e *fine-tuning* 52
Figura 17 - *Bounding boxes* de *ground truth* (em vermelho) e detecção (em verde).....54
Figura 18 - Exemplo de detecção do modelo *YOLOv5* 63
Figura 19 - Exemplo de detecção do modelo *SSD MobileNet*.....64

LISTA DE GRÁFICOS

Gráfico 1 - Degraus da curva de Precisão x *Recall.* 41
Gráfico 2 - Comparação de desempenho das redes *YOLOv5* de diferentes tamanhos 48
Gráfico 3 - Curva de Precisão x *Recall* modelo *YOLOv5* 61
Gráfico 4 – Curva de Precisão x *Recall SSD MobileNet* 61

LISTA DE QUADROS

Quadro 1 – Transformações de aumento de dados aplicadas 47
Quadro 2 - Hiperparâmetros utilizados nos treinamentos dos modelos, com parte dos pesos congelados 56
Quadro 3 -- Hiperparâmetros utilizados no *fine-tuning* dos modelos..... 58
Quadro 4 - Métricas de desempenho obtidas com o *toolkit Object Detection Metrics* 59

LISTA DE TABELAS

Tabela 1 - Comparação entre as redes SSD300, SSD512, YOLO e Faster R-CNN. 33
Tabela 2 - Desempenho dos modelos SSD.....49

LISTA DE ABREVIATURAS E SIGLAS

AP	*Average Precision*
AR	*Average Recall*
API	*Application Programming Interface*
AUC	*Area Under Curve*
CCR	*Coordinated Clusters Representation*
CNN	*Convolutional Neural Network*
COCO	*Common Objects in Context*
GPU	*Graphics Processing Unit*
ILBP	*Improved Local Binary Patterns*
IOU	*Intersection Over Union*
LBP	*Local Binary Patterns*
mAP	*Mean Average Precision*
PANet	*Path Aggregation Network*
ReLU	*Rectified Linear Activation Function*
RNC	Rede Neural Convolucional
RoI	*Region of Interest*
SSD	*Single Shot Multibox Detector*
SVM	*Support Vector Machine*
VOC	*Visual Object Classes*
YOLO	*You Only Look Once*

LISTA DE SÍMBOLOS

AP	Precisão média
$AP@.5$	Precisão média no limiar de IOU de 0,5
$AP@.75$	Precisão média no limiar de IOU de 0,75
AP_S	*AP Across Scales* para objetos pequenos
AP_M	*AP Across Scales* para objetos médios
AP_L	*AP Across Scales* para objetos grandes
AR	*Recall* médio
AR_S	*AR Across Scales* para objetos pequenos
AR_M	*AR Across Scales* para objetos médios
AR_L	*AR Across Scales* para objetos grandes
B_{gt}	Área da *bounding box* de *ground truth*
B_p	Área da *bounding box* prevista
FP	Falso Positivo
FN	Falso Negativo
IOU	Interseção sobre União
L_{conf}	Perda de confiança
L_{loc}	Perda de localização
mAP	Média da precisão média
P	Precisão
$Pr(Objeto)$	Probabilidade de existência de um objeto na célula
R	*Recall*
TP	Verdadeiro Positivo
$\rho(r)$	Precisão ρ no *recall* r

SUMÁRIO

1 INTRODUÇÃO **14**
1.1 Apresentação e Objeto da Pesquisa **14**
1.2 Justificativa **15**
1.3 Objetivos **18**
1.3.1 Objetivo Geral 18
1.3.2 Objetivos Específicos 19
1.4 Estrutura do texto **19**
2 REFERENCIAL TEÓRICO **20**
2.1 Deep Learning **20**
2.2 Redes Neurais Convolucionais **21**
2.2.1 Arquitetura básica das Redes Neurais Convolucionais 21
2.2.2 Camadas Convolucionais 22
2.2.3 Camadas de Pooling 23
2.2.4 Camadas totalmente conectadas 24
2.3 Detecção de objetos **25**
2.4 You Only Look Once (YOLO) **27**
2.4.1 Metodologia de detecção 27
2.4.2 Arquitetura da rede 30
2.5 Single Shot Multibox Detector (SSD) **32**
2.6 Transfer Learning **37**
2.6.1 Fine-tuning 38
2.7 Métricas de desempenho **39**
2.7.1 Interseção Sobre União 39
2.7.2 Curva Precisão x Recall 41
2.7.3 Average Precision e Mean Average Precision 42
2.7.3.1 Interpolação de 11 pontos 42
2.7.3.2 Interpolação utilizando todos os pontos 43
2.7.3.3 *Mean Average Precision* (mAP) 43
2.7.4 Average Recall 44
3 METODOLOGIA **45**
3.1 Base de Dados **45**

3.1.1 Aumento de dados 46
3.2 Seleção dos modelos 48
3.2.1 YOLOv5m-P6 48
3.2.2 SSD MobileNet V1 FPN 640x640 49
3.3 Ambiente de desenvolvimento e frameworks 49
3.4 Treinamento dos modelos 50
3.4.1 Transfer Learning 51
3.5 Avaliação de desempenho 53
4 EXPERIMENTOS E RESULTADOS 56
4.1 Primeira etapa de treinamento 56
4.2 Fine-tuning 57
4.3 Avaliação do desempenho 59
4.3.1 Métricas 59
4.3.2 Gráficos 60
4.4 Resultados 62
5 CONCLUSÕES 66
REFERÊNCIAS BIBLIOGRÁFICAS 69

1 INTRODUÇÃO

1.1 Apresentação e Objeto da Pesquisa

Desde a Antiguidade, as rochas naturais têm sido utilizadas pela humanidade na construção de estruturas, continuando a ser materiais ainda bastante presentes na modernidade. Dentre os tipos de rochas encontradas na natureza, o granito e o mármore se destacam como materiais de notável durabilidade e valor ornamental, utilizado extensivamente em construções, pontes, pavimentação e monumentos.

Segundo dados da Associação Brasileira da Indústria de Rochas Ornamentais, em 2019 as exportações de granito geraram cerca 187 milhões de dólares no Brasil, correspondendo a um volume físico de 4,2 milhões de toneladas de granito. Adicionalmente, as exportações de mármore somaram mais de 14,9 milhões de dólares, movimentando uma quantidade equivalente a 2,3 milhões de toneladas de mármore. Dentre os estados brasileiros, o Espírito Santo vem recebendo grande destaque na indústria de rochas ornamentais, posicionando-se em 2019 como principal produtor de granito e mármore no Brasil e contribuindo com uma parcela de cerca de 3 milhões de toneladas das exportações nacionais de granito e mármore, tendo contribuído com 82% do total do faturamento e 77% do total do volume físico das exportações brasileiras de rochas ornamentais.

Devido à sua origem natural, o granito e o mármore se manifestam em cores e tipos variados, mostrando-se uma tarefa complexa o discernimento dos seus tipos na comercialização. São bastante comuns as situações em que alguns exemplares com mesmas propriedades mineralógicas tenham aparência visual levemente diferente, tornando complexa a precificação e rotulação das rochas. Tendo em vista melhorias no controle de qualidade, na rastreabilidade do produto e no gerenciamento de estoque, as indústrias de rochas naturais têm considerado uma nova abordagem que possa mitigar as faltas de clareza e consistência na classificação dos blocos: a implementação de sistemas automatizados.

Um procedimento com potencial para automatização é a reidentificaçãoreidentificação, que é a tarefa de identificar novamente um ou mais blocos individuais de rochas naturais. Este procedimento é importante no processo de transporte da mercadoria e evitar fraudes quando um

bloco é entregue a um cliente. Sendo normalmente realizada de forma subjetiva por agentes humanos, a reidentificação das rochas está sujeita a erros e pode prejudicar a confiabilidade e rastreabilidade do transporte da mercadoria. Nesse contexto, o trabalho presente busca uma solução computacional para uma parte essencial do problema de reidentificação: a detecção de blocos de rochas naturais em ambientes não controlados.

Propõe-se a investigação das soluções computacionais baseadas em métodos de aprendizado de máquina e visão computacional. Mais especificamente, serão utilizados dois modelos baseados em Redes Neurais Convolucionais para detecção de objetos em imagens: a *You Only Look Once* (*YOLOv5*) (JOCHER, 2020) e o *Single Shot Multibox Detector* (SSD) (LIU et al., 2015). Com a aplicação destes algoritmos detectores de objetos, busca-se uma abordagem que possa ser útil para separar os blocos do fundo, evitando a necessidade de um processo manual de marcação das rochas na imagem em um nível muito detalhado e complexo, como aquele necessário para tarefas de segmentação semântica. É necessário observar que esta pode ser uma etapa útil de pré-processamento antes da classificação e reidentificação da rocha, pois com uma menor presença do fundo, existe uma maior possibilidade na correta identificação do tipo de rocha na imagem.

Para tornar o método proposto mais útil para uma aplicação prática, a abordagem deve permitir a detecção de blocos em imagens obtidas em ambientes não controlados, que possam ter a presença de variação de iluminação, sombras, fundo não homogêneo e com aspecto semelhante às rochas, variações do ângulo das imagens, etc. Assim, dentro do contexto de identificação e reidentificação do tipo de bloco de granito e mármore, o escopo deste trabalho é limitado ao processo de detecção dos blocos de rochas naturais na imagem.

1.2 Justificativa

Nos anos recentes, vários setores industriais têm demonstrado interesse em sistemas automáticos para classificação de produtos de acordo com sua aparência visual, como as indústrias da madeira, mármore, tecidos, papel, etc. Dentro das indústrias do mármore e do granito, em específico, o preço é determinado principalmente pelo seu valor estético, com menor destaque para suas propriedades mecânicas (FERNÁNDEZ et al., 2011). A aparência visual é o fator determinante na classificação das peças, existindo uma variedade de categorias

de cores (verde, rosa, vermelho, branco, etc.) e texturas (listrado, salpicado, homogêneo, etc.). Os padrões atuais de categorização do granito também envolvem especificações geométricas, como a dimensão longitudinal dos blocos, o nivelamento das lajes e a retidão das bordas dos azulejos. No entanto, até mesmo uma análise petrográfica mostra-se insuficientemente acurada, pois a aparência visual dos granitos pode diferir significativamente, ainda quando possuem mesma natureza mineralógica (FERNÁNDEZ et al., 2011).

Os blocos de mármore, por sua vez, são avaliados de acordo com o tipo, características geológicas gerais, parâmetros internos como coesão, pressão, ângulo interno de fricção, linha de processamento e características de uso. Descontinuidades, diferenças de cor, presença de partículas extrínsecas, formato dos blocos e níveis de rugosidade e brilho são os fatores principais na determinação da qualidade das peças de mármore (KARAALI; EMINAĞAOĞLU, 2021).

Muitas vezes, a deficiência de exatidão na especificação da aparência visual das rochas leva a desentendimentos entre fornecedores e clientes. Nessas situações, não é incomum que o cliente recuse lotes de lajes ou azulejos de granito, alegando falta de similaridade com o exemplar fornecido para a escolha do modelo. Outra também frequente fonte de reclamações é a falta de uniformidade na aparência visual das peças dentro de um lote (BIANCONI et al., 2012). Como resultado desses conflitos, empresas estão sujeitas a sofrer grandes perdas econômicas quando precisam arcar com os altos custos no frete e transporte de mercadoria recusada, além de possíveis multas contratuais por atrasos. Para prevenir essas perdas, as indústrias empregam sistemas de controle de qualidade baseadas em uma inspeção visual, realizada por profissionais especializados. De fato, essa análise qualitativa consegue evitar muitas rejeições por parte dos clientes, porém ainda não é uma solução definitivamente satisfatória, pois as decisões tomadas pelos agentes humanos são subjetivas e, nesse caso, muito dependentes da experiência do profissional (SINGH; SINGHSAINI, 2018).

Em busca de uma solução para esse desafio, o problema da classificação de rochas naturais já vem sido abordado há vários anos na literatura científica, sendo muitas das soluções propostas baseadas na aplicação de descritores de cor e textura (ARAÚJO et al., 2010; BIANCONI et al., 2006, 2012; LEPISTÖ, 2005). Mais recentemente, foi observado por Ferreira e Giraldi (2017), e em seguida por Sing e SinghSaini (2018), que os classificadores considerados tradicionais são

muitas vezes generalistas e atuam de forma apropriada somente em ambientes controlados, onde espera-se que certos atributos estejam presentes em todas as imagens das peças de granito e mármore. Por exemplo, em situações que os recortes de granito possuem as mesmas dimensões e as imagens obtidas possuem a mesma resolução e fonte de iluminação.

Explorando uma nova abordagem que aplica os conceitos de aprendizado de máquina, Ferreira e Giraldi (2017) propuseram um novo algoritmo de classificação de chapas de granito baseado em Redes Neurais Convolucionais e Aprendizado Profundo. Foram utilizados pequenos recortes das imagens para extração de padrões discriminativos através da retropropagação do erro nas redes neurais, contrastando com os métodos já bem estabelecidos baseados em Engenharia de Atributos, que utilizam imagens por inteiro. Os resultados alcançados foram comparáveis aos obtidos com os algoritmos estado-da-arte, com o diferencial de que o método proposto utilizando Redes Neurais possui menor sensibilidade a diferentes tipos de resoluções e iluminação, revelando um potencial para utilização em situações do mundo real.

De forma similar a Ferreira e Giraldi (2017), Singh e SingSaini (2018) também dividem as imagens em pequenos recortes e os utilizam como entrada da rede neural convolucional, que neste caso é uma versão customizada da rede neural CIFAR. Em seu estudo, Sing SingSaini (2018) afirmam terem alcançado resultados muitos satisfatórios, obtendo uma ferramenta de classificação de chapas de granito robusta a variações de resolução e a ambientes não controlados.

Em um estudo mais recente, Karaali e Eminağaoğlu1 (2021) desenvolveram um modelo de aprendizagem profunda para realizar a multi-classificação de imagens de placas de mármore, com seis diferentes tipos de qualidade. Foi aplicado o aumento de dados nas imagens utilizando filtros de suavização, filtro passa-baixa de convolução e filtros de erosão e, em seguida, foi desenvolvida e implementada uma arquitetura especial de CNN. Karaali e Eminağaoğlu1 (2021) observaram que a abordagem de aumento de dados para as amostras de imagens melhorou significativamente a precisão do modelo CNN, variando de 0,922 e 0,961.

Observando os resultados promissores encontrados por Ferreira e Giraldi (2017), Singh e SingSaini (2018), e Karaali e Eminağaoğlu1 (2021), neste trabalho também é empregada uma abordagem utilizando CNN através de modelos de detectores de objetos com a meta de atenuar

os efeitos negativos da variabilidade do fundo e características ambientais na detecção dos blocos naturais. Os modelos detectores de objetos realizam a determinação da localização do objeto em imagens, sendo o objetivo a detecção do local da imagem onde o bloco de rocha natural aparece.

Algumas técnicas tradicionais de processamento de imagens foram previamente avaliadas sobre as imagens da base de dados usada neste trabalho como possíveis soluções para detectar a presença e localização dos blocos de rocha nas imagens[1]. Como exemplos de técnicas avaliadas incluem segmentação por k-means clustering, processamento de histograma, detecção de bordas, filtros homomórficos e extração de texturas. Contudo, apesar de obter resultados razoáveis, as soluções encontradas envolviam a definição de vários hiperparâmetros, e apresentavam uma alta variabilidade da qualidade dos resultados, que eram em função da iluminação, variabilidade do fundo, região de sombras, semelhança da cor e textura do chão com os blocos, entre outros.

Neste trabalho, os detectores de objetos serão utilizados como uma ferramenta de pré-processamento, separando os blocos do fundo da imagem. Espera-se que esta ferramenta permita diminuir a influência do fundo em uma etapa subsequente de classificação e/ou reidentificação dos blocos, tornando mais provável a identificação correta das rochas em imagens capturadas em ambientes não controlados. Contudo, é necessário estabelecer que o escopo deste trabalho se limita apenas à detecção dos blocos de granito de rochas naturais.

1.3 Objetivos

1.3.1 *Objetivo Geral*

Tem-se como objetivo geral do projeto o desenvolvimento e análise de desempenho de um algoritmo de detecção de objetos baseado em redes neurais profundas. O objeto a ser detectado são blocos de granito de rochas naturais, cujas imagens foram adquiridas em ambientes não controlados. A detecção dos objetos se configura como uma etapa de pré-processamento para

[1] Pesquisa interna sem trabalho publicado.

uma posterior etapa de classificação e/ou reidentificação dos blocos. Assim, o propósito da detecção é facilitar a separação dos blocos em relação ao fundo da imagem.

1.3.2 *Objetivos Específicos*

- Organizar e preparar uma base de dados rotulada com imagens de blocos de rochas naturais, para treinamento do algoritmo detector de objetos.
- Treinar e otimizar dois algoritmos detectores de objetos nas arquiteturas *YOLOv5* e *SSD*, com o fim de identificar o objeto bloco dentro de imagens obtidas em ambientes não controlados.
- Calcular e comparar os desempenhos dos modelos treinados com o fim de determinar o modelo mais adequado para atingir o objetivo final de detecção dos blocos nas imagens.

1.4 Estrutura do texto

Neste trabalho, o texto está organizado em 5 capítulos. O Capítulo 1 apresenta uma contextualização da problemática abordada por este trabalho, expondo os objetivos com os quais foi desenvolvido e as suas motivações. O Capítulo 2 dá embasamento teórico aos conceitos fundamentais necessários ao entendimento do sistema proposto. O Capítulo 3 detalha a metodologia de desenvolvimento do projeto, onde são descritas as técnicas e etapas que constituem a solução proposta neste trabalho. O Capítulo 4 exibe os experimentos realizados e efetua uma análise dos resultados obtidos. Por fim, o Capítulo 5 apresenta as conclusões relativas aos resultados alcançados e discute possíveis melhorias e propostas de trabalhos futuros.

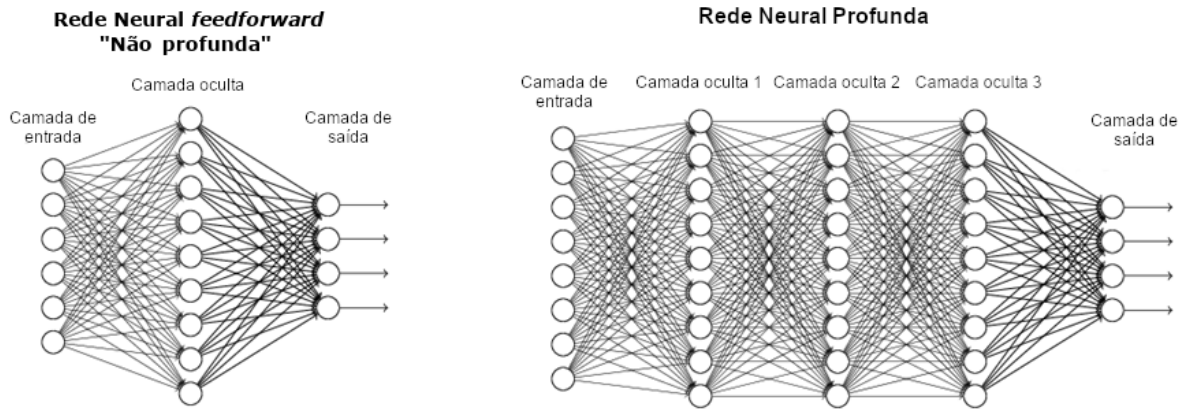
2 REFERENCIAL TEÓRICO

2.1 Deep Learning

Algoritmos de aprendizado de máquina são utilizados em uma grande variedade de aplicações: para identificar objetos em imagens, transcrever fala em texto, recomendar notícias, produtos ou posts relevantes a um usuário, retornar resultados relevantes em uma busca, etc. Nos últimos anos, as técnicas de *Deep Learning* vêm ganhando destaque dentro da área de aprendizado de máquina. Grandes avanços foram alcançados com *Deep Learning* na resolução de problemas que resistiram aos esforços da comunidade científica de inteligência artificial por vários anos. Os modelos de *Deep Learning* se mostraram especialmente eficazes em resolver problemas que lidam com um alto volume de dados, batendo recordes em reconhecimento de imagens e fala e tarefas de processamento de linguagem natural, e com isso encontrando aplicações em várias áreas da ciência, mercado e governo (LECUN; BENGIO; HINTON, 2015).

Representation Learning ou *Feature Learning* são um conjunto de métodos que permitem que um modelo receba dados brutos e descubra de forma automática as representações necessárias para tarefas de detecção ou classificação (BENGIO; COURVILLE; VINCENT, 2012). Métodos de *Deep Learning* são métodos de *Feature Learning* com múltiplos níveis de representação obtidos através da composição de vários módulos não-lineares, onde cada um transforma a representação, nível por nível, em uma representação de nível mais elevado e abstrato. Para tarefas de classificação de imagem, as camadas mais altas da representação destacam os aspectos da entrada que são importantes para a discriminação das características e suprimem as variações irrelevantes. A ideia central do *Deep Learning* é de que as camadas de características não são projetadas manualmente, e sim aprendidas a partir dos dados utilizando algum método de aprendizado (LECUN; BENGIO; HINTON, 2015). Na Figura 1, pode-se ver ilustrada a diferença no maior número de camadas ocultas presentes redes neurais profundas em relação às redes não-profundas.

Figura 1 – Redes neurais “não-profundas” (à esquerda) e redes profundas (à direita)



Fonte: Adaptado de Goodfellow et al., (2016).

O número de camadas é o fator mais evidente na diferenciação entre as redes profundas e as “não profundas”. Ainda não foi estabelecida uma regra universal que faça divisão entre redes profundas e redes “não profundas” quanto ao número de camadas. Contudo, pode-se dizer que a maioria dos pesquisadores concorda que a aprendizagem profunda envolve um número de camadas ocultas superior a 2. (SUGIYAMA, 2019)

2.2 Redes Neurais Convolucionais

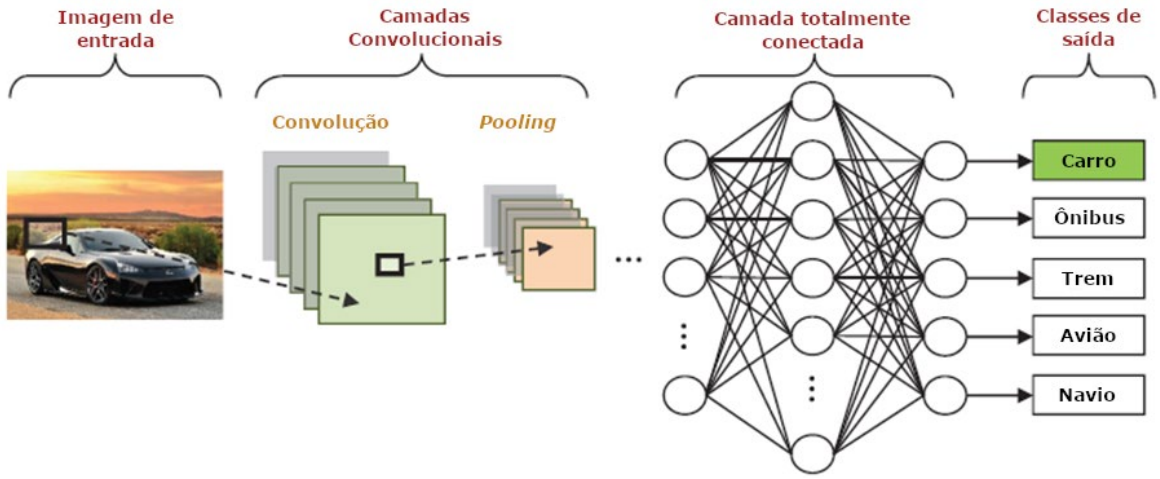
Dentre os diferentes tipos de Redes Neurais Profundas, As Redes Neurais Convolucionais (CNN, do inglês *Convolutional Neural Networks*) têm sido as mais estudadas nos últimos anos. Com o surgimento de grandes volumes de dados rotulados, o desenvolvimento de rápidos sistemas de computação paralela por meio de GPU, e avanços significativos no design de redes e estratégias de treinamento (ZHAO, Zhong-Qiu et al., 2018), as CNN tomaram lugar de destaque como estado-da-arte em aplicações de aprendizado profundo, realizando tarefas de classificação de imagens, visão computacional e processamento de linguagem natural (ALBAWI; MOHAMMED; AL-ZAWI, 2018).

2.2.1 *Arquitetura básica das Redes Neurais Convolucionais*

Segundo O’Shea e Nash (2015), CNN são redes do tipo alimentação direta (do inglês *feed-forward*), onde o fluxo de informações se dá em apenas uma direção, das entradas para as saídas. As arquiteturas das CNN variam, mas de forma geral são compostas por camadas de

convolução e camadas de *pooling*, ou subamostragem, que estão agrupadas em módulos, incluindo também camadas totalmente conectadas. Esses módulos são empilhados uns sobre os outros para formar uma rede profunda que consegue, camada por camada, transformar os dados de entrada originais utilizando as operações de convolução e subamostragem para produzir saídas, com propósito de classificação ou regressão. Nas camadas mais baixas da rede, encontra-se o chamado *backbone*. O *backbone* é utilizado para codificar o *input* da rede em uma certa representação de características, e é onde são extraídos os mapas de características a serem processados em camadas mais altas. O *backbone* comumente é uma rede já bem estabelecida na comunidade de *deep learning*. Temos como exemplo de *backbone* as redes VGG-16 e Darknet, utilizadas, respectivamente, na SSD e na YOLO. No final da rede, assim como na maioria das redes *feed-forward*, encontram-se uma ou mais camadas totalmente conectadas, onde a última camada geralmente é a que fornece a saída da tarefa de classificação ou regressão. Na Figura 2 há uma ilustração da arquitetura de uma CNN.

Figura 2 - Arquitetura de uma Rede Neural Convolucional



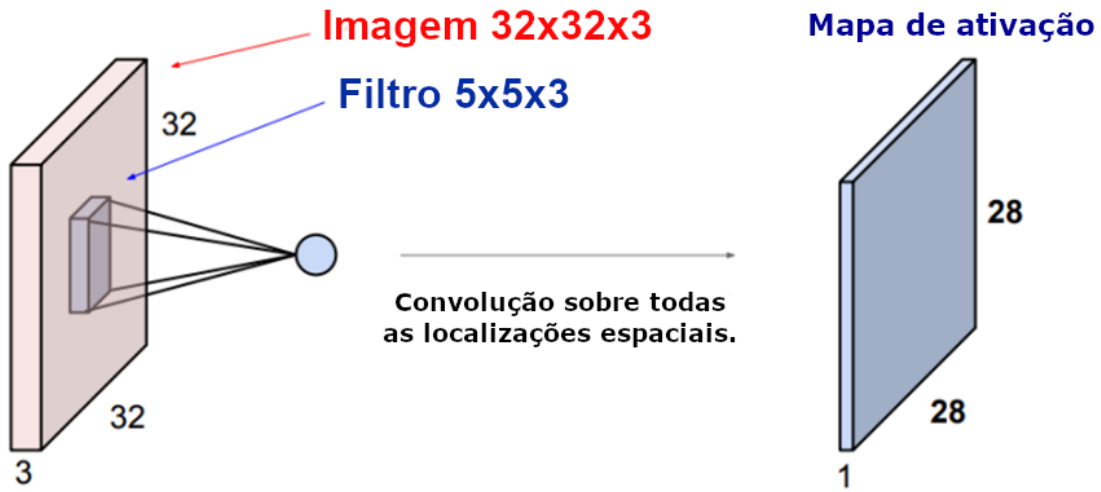
Fonte: Adaptado de Rawat; Wang (2017).

2.2.2 *Camadas Convolucionais*

As camadas convolucionais funcionam como extratores de características, e são as que aprendem as representações de características das imagens de entrada. As saídas dos neurônios nas camadas convolucionais se organizam em mapas de características, onde cada neurônio possui um campo receptivo que é conectado a uma vizinhança do mapa de características da camada anterior através de um conjunto de pesos treináveis, que muitas vezes é denominado de

banco de filtros (do inglês *filter bank*) (LECUN; BENGIO; HINTON, 2015). Como ilustrado na Figura 3, são realizadas convoluções nas entradas utilizando os pesos treinados a fim de gerar um novo mapa de características, e aplica-se uma função de ativação não-linear nos resultados das convoluções. Funções de ativação não-lineares permitem a extração de características não-lineares. Classicamente, empregava-se as funções sigmóide e tangente hiperbólica como funções de ativação, mas nos últimos anos tem predominado o uso da função unidade linear retificada (ReLU, do inglês *rectified linear activation function*), principalmente nas camadas mais profundas (RAWAT; WANG, 2017).

Figura 3 - Geração de um mapa de características em camadas de convolução



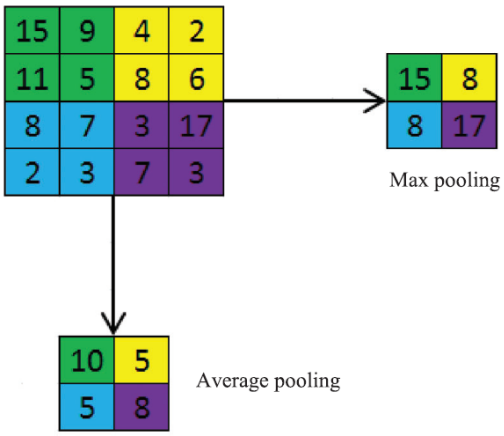
Fonte: Adaptado de Karpathy et al, 2016.

2.2.3 *Camadas de Pooling*

A ideia do *pooling* é realizar uma subamostragem a fim de reduzir a complexidade da representação para as camadas posteriores, diminuindo o número de parâmetros e a complexidade computacional do modelo. No campo de processamento de imagens em específico, o objetivo das camadas de *pooling* é reduzir a resolução espacial dos mapas de características e com isso melhorar a invariância espacial para distorções e translações na entrada (LECUN; BENGIO; HINTON, 2015; RAWAT; WANG, 2017). O método mais comum de *pooling* é o *max-pooling*, que opera em cada mapa de características da entrada e extrai o valor máximo quando aplicada dentro do campo receptivo.

Existe também outro tipo de *pooling* chamado *average pooling*, que funciona com os mesmos princípios do *max pooling*, com a exceção de que se computa a média aritmética (O'SHEA; NASH, 2017). O exemplo da Figura 4 ilustra a diferença entre o *max pooling* e o *average pooling*. Tomando-se uma imagem de tamanho 4x4, aplicando-se um filtro 2x2 de *stride*[2] 2, o *max pooling* retorna o maior valor de cada região 2x2 da imagem original, enquanto o *average pooling* retorna o valor inteiro arredondado da média aritmética de cada região subamostrada.

Figura 4 - Comparativo entre *max pooling* e *average pooling*.



Fonte:(Rawat; Wang, 2017)

2.2.4 *Camadas totalmente conectadas*

Na camada totalmente conectada, normalmente localizada no final da CNN, os neurônios são arranjados de maneira similar às camadas das redes neurais tradicionais. Como já descrito anteriormente, essas camadas interpretam as representações de características extraídas pelas camadas anteriores, de convolução e *pooling*, exercendo discernimento de alto nível (RAWAT; WANG, 2017; SIMONYAN; ZISSERMAN, 2014). Geralmente, a função de ativação aplicada à última camada totalmente conectada é diferente da aplicada nas camadas anteriores, a depender da tarefa que está sendo realizada. A função de ativação mais comumente aplicada em tarefas de classificação multiclasse é a função *softmax*, que normaliza os valores reais da

[2] O *stride* é um parâmetro do filtro da rede que modifica o quanto o filtro se move sobre a imagem ou video, ou seja, o *stride* é o número do *shift* de pixels sobre a matriz de entrada.

última camada totalmente conectada para as probabilidades das classes desejadas, onde cada valor varia de 0 a 1, sendo a soma de todos estes valores igual a 1 (YAMASHITA et al., 2018).

A maior desvantagem das camadas totalmente conectadas é que estas possuem muitos parâmetros, o que exige cálculos computacionais complexos no treinamento da rede (ALBAWI; MOHAMMED; AL-ZAWI, 2018).

2.3 Detecção de objetos

A detecção de objetos é um campo de pesquisa que incorpora as áreas de visão computacional, inteligência artificial e processamento de imagens, sendo uma parte essencial em tarefas como rastreamento de objetos, detecção de eventos, análise comportamental, reconhecimento facial e direção autônoma. Por definição, o problema de detecção de objetos consiste em: determinar onde os objetos estão localizados em uma dada imagem (localização de objetos), definir de forma precisa a qual categoria este objeto pertence, e estimar uma caixa delimitadora (do inglês *bounding box*) para cada classe detectada (Zhao et al., 2018).

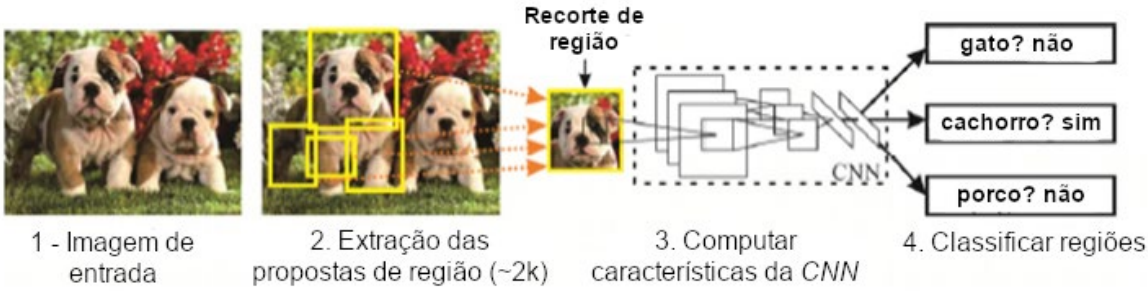
Figura 5 - Exemplo de imagem com *bounding boxes*

Fonte: Park, 2021.

No que diz respeito à *pipeline*, os algoritmos de detecção de objetos podem ser divididos em duas classes principais: algoritmos de estágio único baseados em proposta de região (do inglês *region proposal*) e algoritmos de dois estágios baseados em regressão (CARRANZA-GARCÍA et al., 2021):

- Nas arquiteturas de dois estágios, como a R-CNN (GIRSHICK et al., 2014) e a SPPNet (ZHU et al., 2016), o processo de detecção é dividido em um estágio de proposta de região e um estágio de classificação. Como mostrado na Figura 6, primeiramente são estimados vários candidatos a objetos, chamados de "regiões de interesse" (RoI, do inglês *Region of interest*) utilizando *boxes* de referência, também conhecidas como caixas âncora (do inglês, *anchor boxes*). No próximo estágio, as regiões propostas são classificadas e sua localização é apurada. Os métodos de dois estágios são relativamente mais lentos, mas são reconhecidos por seu alto desempenho (CARRANZA-GARCÍA et al., 2021).

Figura 6 - Exemplo de algoritmo de dois estágios



Fonte: Adaptado de Deng et al., 2020

- Os métodos de estágio único, por outro lado, necessitam de apenas uma execução através da rede neural, pulando a etapa de *region proposal* vista nos modelos de dois estágios e realizando predições diretamente sobre um *cluster* de localizações. Estes modelos costumam realizar inferências de maneira mais rápida, mas supõe-se que isso possa gerar um custo na eficácia. Os exemplos mais comuns de modelos estágio único são: SSD (LIU et al., 2015), YOLO (REDMON et al., 2015) e SqueezeDet (WU et al., 2017; DENG et al., 2020).

No trabalho aqui proposto, o desenvolvimento e as análises serão realizados para os modelos *YOLO* e SSD. A escolha desses modelos *one-stage* leva em consideração suas velocidades de predição, sua atualidade e a praticidade do tempo de treinamento.

*2.4 You Only Look Once (*YOLO*)*

Em 2015 surgiu o algoritmo *You Only Look Once* (YOLO) (REDMON et al., 2015), trazendo uma nova abordagem ao desafio da detecção de objetos. A YOLO propunha transformar a detecção de objetos em um problema de regressão, a ser executado em uma única rede neural. Com a YOLO, obteve-se um salto qualitativo nos algoritmos de detecção de objetos, chegando a resultados bastante superiores ao que se podia obter há uma década de seu lançamento (REDMON et al., 2015).

Até o momento da formulação deste projeto, o algoritmo YOLO já passou por 4 atualizações, estando em sua quinta versão. Em sua evolução, a YOLO vem incorporando várias ideias inovadoras da comunidade de pesquisa em visão computacional, sendo ainda considerado um algoritmo de desempenho exemplar. A quinta e mais recente geração da YOLO, conhecida como *YOLOv5*, não foi desenvolvida pelo autor original da YOLO, porém, deve-se ressaltar que o desempenho da nova versão continua sendo objetivamente superior ao alcançado pela versão anterior, a *YOLOv4*, tanto em termos de acurácia quanto de velocidade. Dentre as principais melhorias implementadas na *YOLOv5*, tem-se a migração da linguagem C para a linguagem *Python* e a implementação do cálculo adaptativo de *anchor boxes*.

O objetivo do autor da versão original consistia em construir um modelo que unificasse todas as fases de uma rede neural, ou seja, um modelo de estágio único. Por meio da imagem de entrada contendo os objetos a serem identificados e passando por uma única rede neural de múltiplas camadas convolucionais, o sistema produz vetores preditivos correspondentes a cada objeto presente na imagem. Ao invés de realizar iterações no processo de classificação de diferentes regiões na imagem, a arquitetura YOLO simultaneamente prevê múltiplas *bounding boxes* e probabilidades de classes para estas *boxes*, avaliando todas as características da imagem e fazendo predições para todos os objetos ao mesmo tempo. Por isso o nome *"You Only Look Once"* ("Você só olha uma vez") (REDMON et al., 2015).

2.4.1 *Metodologia de detecção*

O conceito principal da YOLO é dividir a imagem em uma grade de células de tamanho $S \times S$ (7×7, por padrão). Se o centro de um objeto está posicionado em uma célula, aquela célula é

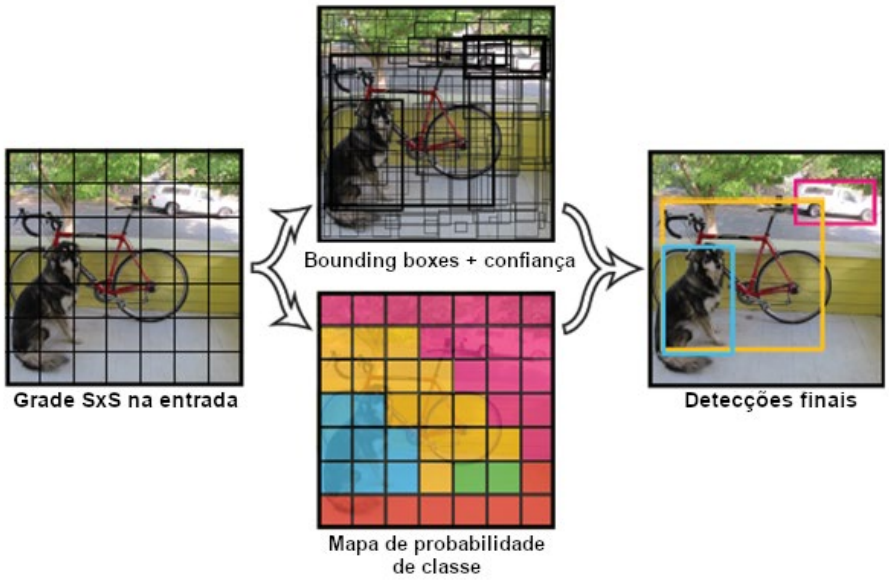
responsável por detectar este objeto. Cada célula da grade prevê *B bounding boxes* com seus parâmetros e notas de confiança. As notas de confiança refletem a presença ou ausência de um objeto dentro da *bounding box*. A nota de confiança pode ser definida como o produto da probabilidade da existência de um objeto na célula e a chamada IOU (*Intersection Over Union*), descrita na seção 2.6.1, das *bounding boxes* de predição e *ground truth* (THUAN, 2021):

$$nota_{confiança} = \Pr(Objeto) * IOU_{pred}^{truth} \tag{1}$$

Onde a nota de confiança é igual a 0 se não existe objeto dentro da célula. Caso exista um objeto dentro da célula, é desejável que a nota de confiança seja igual à IoU entre a *bounding box* prevista e o *ground truth*. Cada célula da grade também prevê *C* probabilidades de classe condicionadas a esta célula. É previsto apenas um conjunto de probabilidade de classes, independentemente do número *B* de *bounding boxes* (REDMON et al., 2015).

Na Figura 7, pode ser visto um exemplo simplificado da formação das *bounding boxes* no algoritmo da YOLO. Na ilustração das *bounding boxes* com nota de confiança, as linhas mais espessas indicam que existe uma maior probabilidade da existência de um objeto na região, pois são sobreposições de múltiplas *boxes* preditas por diferentes células. No mapa de probabilidade de classe, as cores em cada célula indicam a classe com a maior probabilidade de classe. Para gerar as detecções finais é aplicada a supressão de não-máximos, eliminando as *boxes* com menor nota de confiança. Adicionalmente, cada célula estabelece a classe da *box* que está contida nesta. Produz-se assim, a predição final da *bounding box* junto com a sua probabilidade de classe (REDMON et al., 2015).

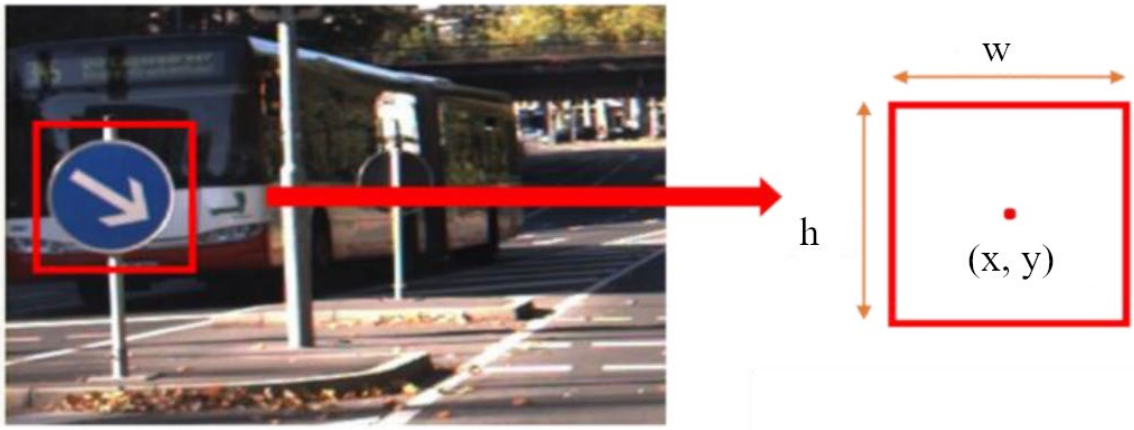
Figura 7 - Células e *bounding boxes*



Fonte: Adaptado de Redmon et al., 2015

Como ilustrado na Figura 8, cada *bounding box* contém 4 parâmetros (x, y, w, h) que correspondem às coordenadas do centro (x,y), o comprimento e altura (w,h) da *bounding box*. Dessa forma, as *bounding boxes* contêm 5 parâmetros no total, sendo o quinto parâmetro a nota de confiança (REDMON et al., 2015).

Figura 8 - Exemplo de parâmetros dimensionais de uma *bounding box*



Fonte: Adaptado de Manawadu; Wijenayake, 2018

Sumarizando, o modelo possui uma grade $S \times S$ células para uma imagem. Cada célula faz a predição de B *bounding boxes* que consistem de 5 parâmetros e compartilham probabilidades de predição de C classes. A saída do modelo será um tensor de $S \times S \times (5 \times B + C)$

(THUAN, 2021). Por exemplo, se for aplicado o *dataset Pascal VOC* (EVERINGHAM et al., 2010) na YOLO, os parâmetros utilizados seriam $S = 7$, $B = 2$ e, existindo 20 classes no *dataset*, $C = 20$, gerando um tensor de dimensões $7 \times 7 \times 30$ (REDMON et al., 2015).

Durante a evolução da *YOLOv1* até a *YOLOv5*, foram implementadas algumas mudanças na configuração das grades no funcionamento das *bounding boxes*. Na *YOLOv2*, cada célula da grade $S \times S$ deixa de realizar predições para apenas um objeto e passa a prever múltiplos objetos ao mesmo tempo. Ou seja, as *bounding boxes* preditas são capazes de detectar diferentes objetos mesmo que estes sejam previstos pela mesma célula da grade. Assim, ao invés de compartilhar com as demais, cada *bounding box* predita passa a ter seu próprio conjunto de probabilidades de C classes. Diferentemente da *YOLOv1*, da *YOLOv3* em diante o número total de parâmetros de saída para cada detector é de $S \times S \times \left(B \times (5 + C)\right)$ (THUAN, 2021).

Por fim, a YOLO aplica a técnica de Supressão de Não-Máximo (NMS, do inglês *Non-Maximum Supression*) para eliminar todas as *bounding boxes* que não contêm nenhum objeto, ou que contêm o mesmo objeto que as outras *bounding boxes* da vizinhança com maior nota de confiança. Escolhendo um valor de limiar, o NMS seleciona as predições com as notas de confiança máximas e suprime todas as outras predições que sobrepõem as predições selecionadas cujos IOU se encontram acima de um limiar. (HOSANG; BENENSON; SCHIELE, 2017).

2.4.2 *Arquitetura da rede*

A arquitetura da *YOLOv5* consiste em 4 partes principais: entrada, *backbone*, *neck*, *head* e saída. As camadas de entrada são responsáveis pelo pré-processamento dos dados, incluindo um tipo de *data augmentation* chamado de *mosaic augmentation* e preenchimento de imagem adaptativo. Para se adaptar a diferentes *datasets*, a *YOLOv5* implementa o cálculo adaptativo de *anchor boxes* na entrada, definindo automaticamente as *anchor boxes* iniciais quando o *dataset* muda. (WU et al., 2017; LI et al., 2022)

A *YOLOv5* incorporou uma Rede Parcial de Estado Cruzado (CSP, do inglês *Cross Stage Partial Network*) à *Darknet*, gerando a CSPDarknet como *backbone* (WANG et al., 2020). A CSPDarknet resolve o problema da repetição da informação de gradiente e integra as mudanças

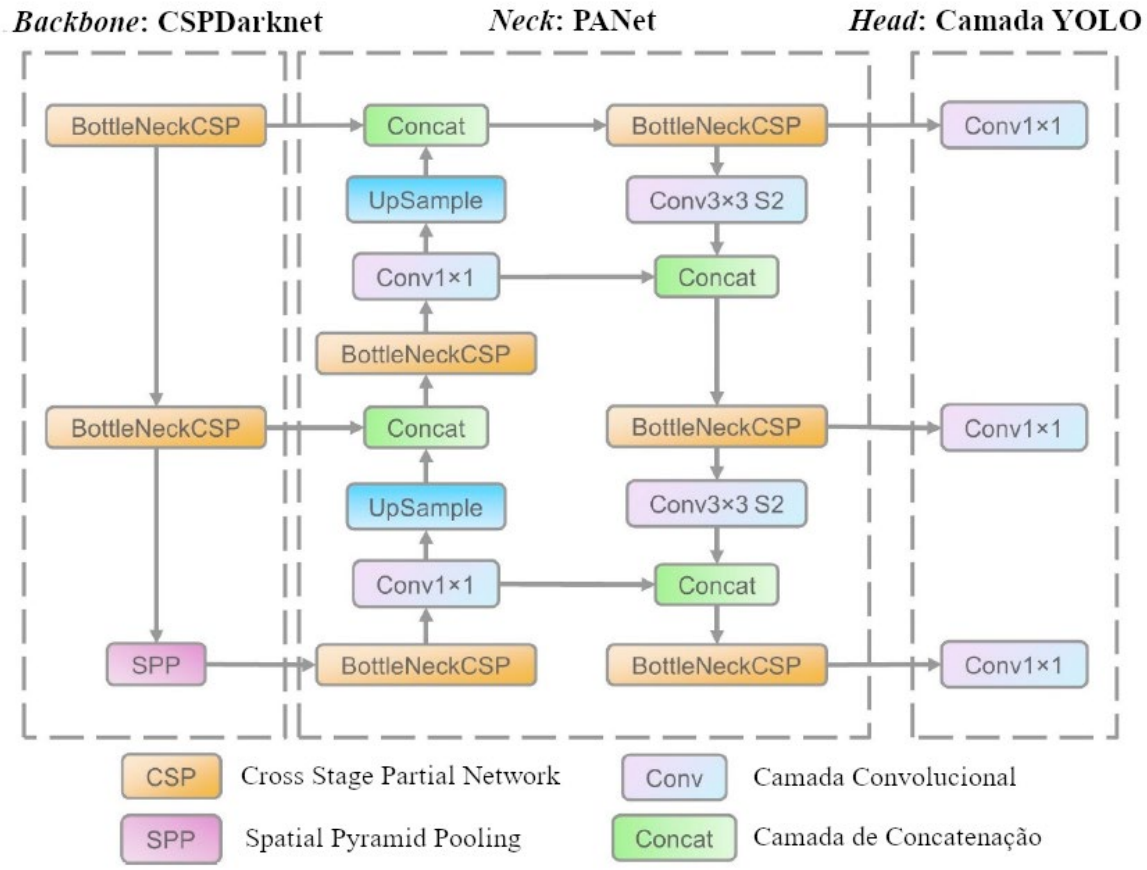
de gradiente no mapa de características, diminuindo o número de parâmetros e operações de número flutuante por segundo (FLOPS, do inglês *Floating-Point Operations Per Second*) do modelo, melhorando a velocidade e acurácia e reduzindo o tamanho da rede (XU et al., 2021).

Nas camadas intermediárias (o *neck* da rede), a *YOLOv5* incorpora uma estrutura de Rede de Agregação de Rotas (PANet, do inglês *Path Aggregation Network*) para ampliar o fluxo de informações (WANG et al., 2019). A PANet adota uma nova estrutura de Rede de Pirâmide de Características (FPN, do inglês *Feature Pyramid Network*), que melhora a propagação de características de baixo nível. A PANet melhora a utilização de sinais de localização em camadas mais baixas, o que pode melhorar a acurácia de localização do objeto. (XU et al., 2021)

Por fim, as camadas finais da rede *YOLOv5*, a *head*, chamada de camada YOLO gera três diferentes tamanhos (18×18, 36×36 e 72×72) de mapas de características para alcançar predições em múltiplas escalas, capacitando o modelo a lidar com objetos de pequeno, médio e grande dimensões (LI et al., 2022).

A arquitetura da rede *YOLOv5* é mostrada na Figura 9. Os dados passam pelo pré-processamento das camadas de entrada para o *backbone* CSPDarknet para extração de características, então são fornecidos à PANet para fusão das características. Por fim, a camada YOLO retorna os resultados da detecção.

Figura 9 - Arquitetura da rede *YOLOv5*



Fonte: Adaptado de Li et al., 2022

2.5 Single Shot Multibox Detector (SSD)

A rede SSD foi proposta por Liu et al. (2015), se tornando um dos mais populares algoritmos de detecção de objetos na comunidade de visão computacional. Existe uma variedade de estudos comparando o desempenho da SSD ao da YOLO (DEEPA et al., 2019; LIU et al., 2015; MORERA et al., 2020) que mostram uma vantagem da SSD em desempenho e velocidade em variadas situações e modelos. Por exemplo, pode ser vista na Tabela 1, retirada e adaptada do estudo de Liu et al. (2015), uma comparação entre a SSD, a Faster R-CNN e a YOLO.

Tabela 1 - Comparação entre as redes SSD300, SSD512, YOLO e Faster R-CNN.

Sistema	mAP Teste VOC2007	FPS (Titan X)	Número de *boxes*	Resolução de Entrada
Faster R-CNN (VGG16)	73.2	7	~6000	~1000 x 600
YOLO (customized)	63.4	45	98	448 x 448
SSD300* (VGG16)	77.2	46	8732	300 x 300
SSD512* (VGG16)	**79.8**	19	24564	512 x 512

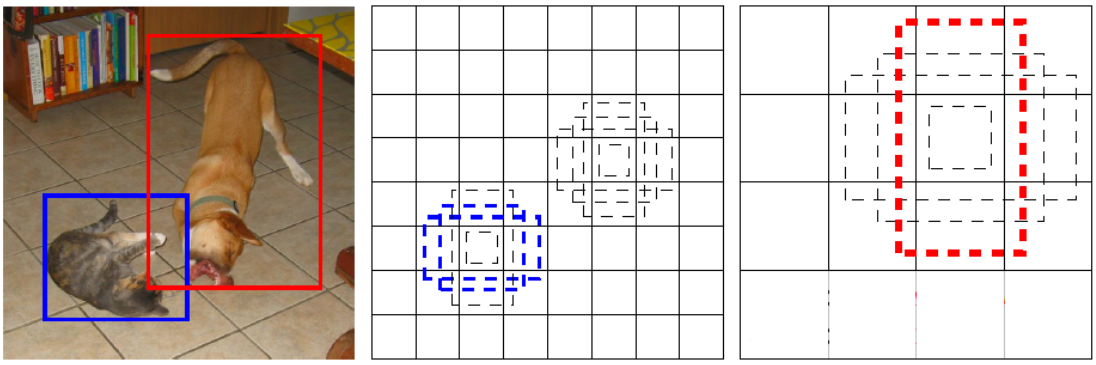
Fonte: Adaptado de Liu et al., 2015

As camadas iniciais são baseadas em uma arquitetura padrão derivada da *Faster* RCNN, utilizada em classificação de imagens de alta qualidade (porém, recortada antes das camadas de classificação). Após as camadas iniciais, a SSD adiciona camadas convolucionais para extração de características. Essas camadas diminuem em tamanho de forma progressiva, e permitem predições de detecções em múltiplas escalas. Com camadas de menor resolução, por exemplo, a *SSD* consegue detectar objetos maiores (LIU et al., 2015). A essa técnica de utilização de camadas com diferentes resoluções dá-se o nome de detecção com mapas de características multi-escala.

As técnicas de multi-escala e multi-referência foram introduzidas para compensar a perda de acurácia (que também ocorre com a YOLO) gerada ao se pular a etapa de *region proposal* vista nos algoritmos de dois estágios, como a R-CNN, em busca de um ganho em velocidade de detecção.

Na Figura 10, pode-se observar que os objetos gato (caixa azul) e cachorro (caixa vermelha), com *ground truth boxes* de diferentes dimensões, são detectados respectivamente pelos mapas de características 11(b) de resolução 8x8 e 11(c) de resolução 4x4.

Figura 10 - Detecção com mapas de características de diferentes dimensões

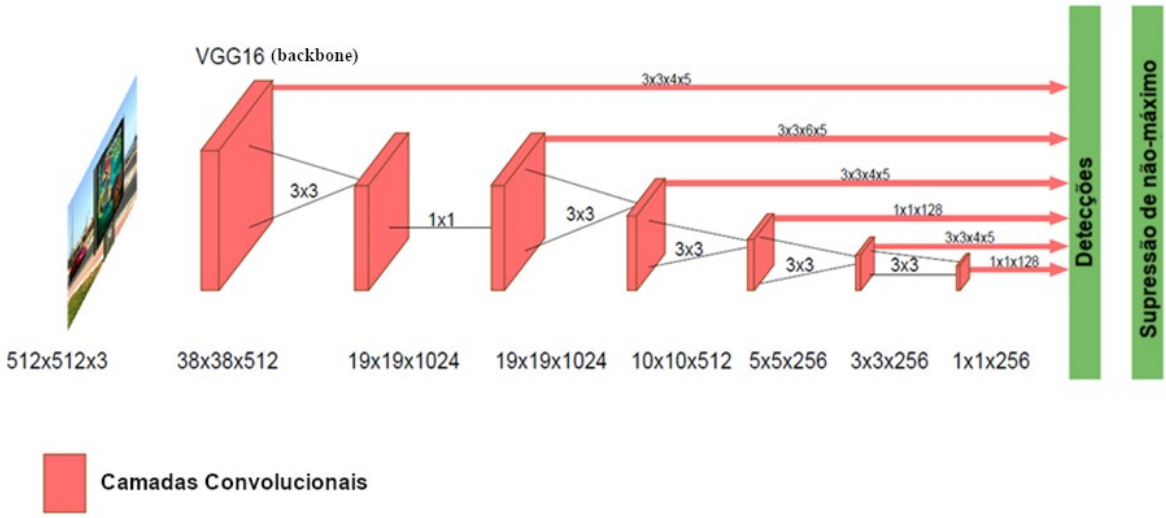


(a) Imagem com as ground truth boxes (b) Mapa de características 8x8 (c) Mapa de características 4x4

Fonte: Adaptado de Liu et al., 2016

Em sua operação, a SSD segue duas etapas: extrair mapas de características e aplicar filtros convolucionais para detecção de objetos. A SSD utiliza a arquitetura *VGG16* de *backbone* para extrair os mapas de características. Em seguida, ele faz a detecção dos objetos utilizando a camada *Conv4_3* do *VGG16*. A *Conv4_3* realiza um total de 38 × 38 × 4 predições, dividindo os mapas de características em 38 × 38 células e realizando quatro predições por célula, independentes da profundidade dos mapas de características. É esperado que muitas predições não contenham nenhum objeto, então utiliza-se a classe "0" para indicar que nenhum objeto foi detectado na imagem (MORERA et al., 2020). A Figura 11 mostra a arquitetura de camadas típica da rede SSD.

Figura 11 - Arquitetura de camadas da rede SSD



Fonte: Adaptado de Morera et al., 2020

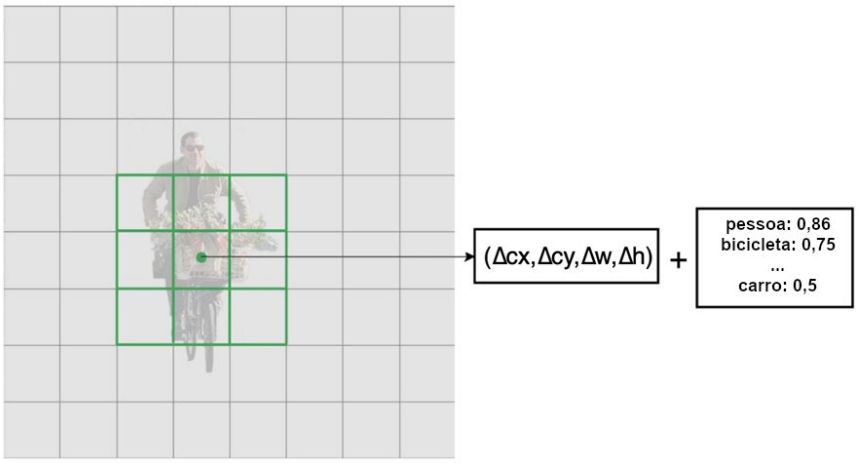
Na *SSD*, assim como na *YOLO*, existe um conjunto de *boundary boxes* definidas manualmente, as chamadas *anchor boxes* ou caixas-padrão (do inglês *default boxes*). São utilizadas 4 ou 6 *default boxes*, com uma predição para cada. No lugar de coordenadas globais para as *bounding boxes*, utiliza-se um sistema de *offsets* da localização da *bounding box* com relação às *default boundary boxes* (HUI, 2018).

Contrapondo o uso do *region proposal* dos detectores de dois estágios, a SSD calcula tanto a localização quanto os *scores* das classes utilizando filtros convolucionais de pequenas dimensões. Após a extração dos mapas de características no *backbone*, a SSD aplica filtros convolucionais 3×3 em cada célula, a fim de realizar predições. Para cada uma quantidade k de *bounding boxes* em uma dada localização, computa-se os c *scores* de classe e os 4 *offsets* relativos à posição da *default box* original. Isso resulta em um total de $(c + 4) \times k$ filtros que são aplicados sobre cada localização no mapa de características, resultando em $(c + 4) \times k \times m \times n$ saídas para um mapa de características de dimensões $m \times n$.

É válido observar que, comparado aos modelos de detecção de objetos de dois estágios, baseados em regiões (Faster R-CNN, R-FCN, etc.), a SSD sofre de ter um menor desempenho na detecção de objetos pequenos devido à baixa resolução dos mapas de características gerados (HUI, 2018; LIU et al., 2015).

Como pode ser visto na Figura 12, as coordenadas das *bounding boxes* $(\Delta cx, \Delta cy, \Delta w, \Delta h)$ são relativas às *default boundary boxes* de cada célula, ou seja, são os *offsets* com relação à *default box* em cada célula pelo seu centro (cx, cy), comprimento e altura. Nesta ilustração, a aplicação do filtro 3×3 gera uma saída com todos os *scores* de classe e as coordenadas de uma *bounding box*.

Figura 12 - Ilustração de uma *bounding box* com *scores* de classe



Fonte: Adaptado de Hui, 2018

A SSD emprega um sistema de correspondências positivas e negativas para calcular o custo de localização, ou seja, a disparidade entre as *bounding boxes*. Dá-se uma correspondência positiva quando o IoU da *default box* com relação à *ground truth box* é maior que 0,5. Essa estratégia de correspondência favorece a predição de *bounding boxes* com formatos mais próximos da *default box* correspondente, aumentando a estabilidade no treinamento (LIU et al., 2015; HUI, 2018).

Como função de perda, a *SSD* aplica uma soma ponderada da perda de localização e da perda de confiança (*localization loss* e *confidence loss*). A perda de localização é dada pela disparidade da *bounding box* prevista e a *ground truth box*. A perda de confiança é a perda nas predições de classes, e é calculada a partir do *score* de confiança da classe correspondente. Seja $x_{ij}^{p} = \{0,1\}$ o indicador do pareamento da i-ésima *default box* com a j-ésima *ground truth box* de categoria *p*, a definição geral da função de perda é dada pela Equação 1.

$$L(x, c, l, g) = \frac{1}{N}\Big(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)\Big) \tag{2}$$

onde *N* é o número de *default boxes* pareadas, L_{loc} é a perda L1 entre os parâmetros da *bounding box* prevista (*l*) e a *ground truth box* (g), e L_{conf} é a perda de confiança *c* calculada pelo *score* de confiança de classes (MORERA et al., 2020).

Ao fim da etapa de pareamento, a maioria das *default boxes* são negativas, gerando um desbalanceamento entre classes que prejudica o treinamento. Ao invés de se utilizar todos os exemplos negativos, estes são selecionados pela maior perda de confiança para cada *default box*, onde são escolhidas as maiores. Esse processo chama-se de Mineração Dura Negativa (do inglês *Hard Negative Mining*), e auxilia com uma otimização mais rápida e treinamento mais estável (HUI, 2018).

A SSD também faz uso de *data augmentation* (aumento de dados) para melhorar a acurácia. De forma randomizada, cada imagem de treinamento pode ser utilizada de forma integral, ou pode-se utilizar um pedaço da imagem com IOU maior que um limiar, ou até mesmo um pedaço amostrado aleatoriamente. Esses pedaços, ou a própria imagem inteira, podem ser redimensionados, invertidos ou distorcidos. Por fim, assim como a YOLO, a SSD faz uso da Supressão de Não-Máximos para remover predições duplicatas ligadas ao mesmo objeto (LIU et al., 2015).

2.6 Transfer Learning

Transfer learning é a facilitação do aprendizado de uma nova tarefa através da transferência de conhecimento de uma tarefa relacionada que já foi aprendida. No caso de redes profundas, é comum utilizar uma rede que foi previamente treinada para uma tarefa "A" para realizar uma tarefa "B". A tarefa "B" pode ser aprendida pela rede com um esforço bem menor do que foi necessário para aprender a realizar a tarefa "A", desde que as tarefas "A" e "B" tenham um certo grau de semelhança. A maioria dos algoritmos de aprendizado de máquina são projetados para abordar tarefas isoladas. O *transfer learning* tenta mudar isso ao desenvolver métodos para transferir o conhecimento aprendido em uma ou mais tarefas de origem e utilizá-lo para melhorar o aprendizado em uma tarefa de destino relacionada (TORREY; SHAVLIK, 2010).

No mundo real, podemos observar muitos exemplos de *transfer learning*. Por exemplo, podemos observar que aprender a reconhecer maçãs pode ajudar a reconhecer pêras. Da mesma forma, aprender a tocar o teclado eletrônico pode ajudar a facilitar a aprendizagem do piano. *Transfer learning* está relacionado a problemas como aprendizado multitarefas e deriva de conceito (do inglês *concept drift*) e não é uma área de estudo exclusiva de *Deep Learning*. (TORREY; SHAVLIK, 2010).

O *transfer learning* é popular na comunidade de *Deep Learning* dados os enormes recursos necessários para treinar modelos de *Deep Learning* ou os grandes e desafiadores *datasets* com os quais os modelos de *Deep Learning* são treinados. A ideia principal por trás do *transfer learning* é pegar emprestados dados rotulados ou extrair conhecimentos de alguns domínios relacionados para ajudar um algoritmo de aprendizagem de máquinas a alcançar um maior desempenho no domínio de interesse (TORREY; SHAVLIK, 2010).

No *transfer learning*, primeiro treinamos uma rede de base num conjunto de dados e tarefa iniciais, e depois redirecionamos as características aprendidas, ou as transferimos, para uma segunda rede alvo a ser treinada num conjunto de dados e tarefa novo. Este processo tende a funcionar se as características forem gerais, ou seja, adequadas tanto às tarefas iniciais quanto às tarefas de destino, em vez de específicas para a tarefa inicial (YOSINSKI et al., 2014).

Numa CNN pré-treinada, as primeiras camadas da base convolucional normalmente são as que aprendem características gerais das imagens de entrada, pouco dependentes do problema em si, tornando-as potencialmente reutilizáveis para problemas de visão computacional diferentes para o qual foram treinadas. Adicionalmente, as últimas camadas da base convolucional e as camadas detectoras aprendem características mais específicas, que dependem do problema para o qual a rede está sendo treinada (YOSINSKI et al., 2014).

2.6.1 *Fine-tuning*

O *fine-tuning* é um procedimento baseado no conceito de *transfer learning*. Originalmente, o *fine-tuning* referia-se ao processo de pré-treinar uma rede neural com um objeto generalista seguido por uma fase adicional de treinamento com um objetivo específico, utilizando o mesmo *dataset*. Mais recentemente, o *fine-tuning* passou a se definir como a reutilização de valores de parâmetros estimados em *datasets* potencialmente grandes para inicialização em aplicações com acesso limitado a dados rotulados. Esta abordagem preparou o caminho para ganhos significativos de desempenho em muitas aplicações (KÄDING et al., 2016).

A aplicação do *fine-tuning* em um modelo pré-treinado envolve a atualização dos pesos do modelo pré-treinado através da continuação do algoritmo de retropropagação. O *fine-tuning* pode ser aplicado a todas as camadas da rede ou é possível manter algumas das camadas

anteriores (também chamadas camadas congeladas) e apenas ajustar algumas camadas de nível superior. Isto pode ser feito porque as características das camadas iniciais são genéricas e, portanto, transferíveis para uma variedade de conjuntos de dados e tarefas específicas, enquanto que as características posteriores se tornam progressivamente mais específicas para um problema em particular (YOSINSKI et al., 2014).

2.7 Métricas de desempenho

No campo de detecção de objetos, diversas abordagens têm sido implementadas na tentativa de suprir a crescente necessidade de modelos de detecção de objetos precisos. Mais recentemente, com a popularização das redes neurais convolucionais e de arquiteturas aceleradas por GPU, os algoritmos de detecção de objetos começaram a ser desenvolvidos e avaliados por uma diferente perspectiva (LECUN; BENGIO; HINTON, 2015).

Existem várias métricas utilizadas em diversos trabalhos e implementações para avaliação de desempenho em problemas de detecção de objetos. Idealmente, para que existam *benchmarkings* confiáveis diante de diferentes abordagens, seria necessária uma implementação flexível que possa ser utilizada por todos, independente da base de dados utilizada. Dentre as bases de dados mais populares na comunidade científica e em competições de detecção de objetos, a métrica mais comumente utilizada é a precisão média (AP, do inglês *average precision*) (PADILLA et al., 2020).

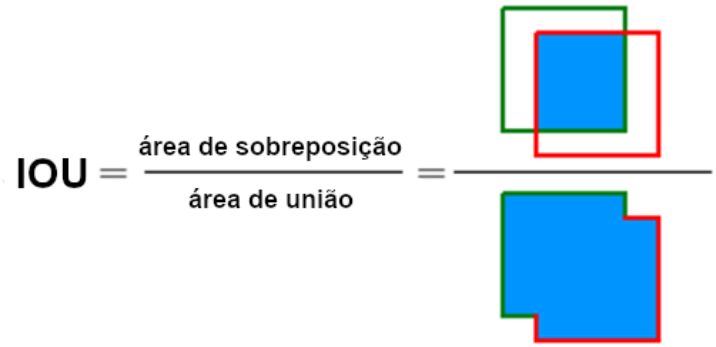
2.7.1 *Interseção Sobre União*

A Interseção Sobre União (IOU, do inglês *Intersection Over Union*) é uma métrica baseada no índice de Jaccard que avalia a sobreposição entre duas *bounding boxes*. A IOU necessita de uma caixa delimitadora de referência (do inglês *ground truth bouding box*) e da *bounding box* prevista. Ao aplicar a IOU, pode-se estabelecer se uma detecção é válida (positivo verdadeiro) ou não (falso positivo). A IOU é calculada a partir da área sobreposta (interseção) entre a *bounding box* prevista (B_p) e a *bounding box* de *ground truth* (B_{gt}) dividida pela área de união entre as duas (PADILLA et al., 2021), como pode ser visto na Equação 2.

$$IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{3}$$

Os valores obtidos na Equação 1 variam de 0,0 a 1,0, onde 1,0 indica uma correspondência exata e 0,0 indica a ausência de sobreposição $area(B_p \cap B_{gt}) = 0$ (COCHARD, 2021). Pode-se ver na Figura 13 uma ilustração da IOU entre uma *ground truth bounding box* (em verde) e a *bounding box* detectada (em vermelho).

Figura 13 - Definição da IOU



Fonte: Adaptado de Padilla et al., 2021

Devem ser definidos, também, alguns conceitos básicos relacionados às métricas aqui descritas (PADILLA et al., 2020):

a) Verdadeiro Positivo (VP): Uma detecção correta, que obteve IOU ≥ limiar.
b) Falso Positivo (FP): Uma detecção incorreta, que obteve IOU < limiar.
c) Falso Negativo (FN): A *ground truth* não foi detectada pelo modelo, não há interseção entre uma *bounding box* detectada e a *ground truth*.
d) Verdadeiro Negativo (VN): Não se aplica.
e) Precisão: Indica a capacidade do modelo em identificar apenas os objetos relevantes. É a porcentagem de predições positivas corretas, e é calculada pela Equação 3.

$$P = \frac{VP}{VP + FP} = \frac{VP}{todas\ as\ detecções} \tag{4}$$

f) Sensibilidade (do inglês, *recall*): É capacidade do modelo de encontrar todos os casos relevantes (todas as *ground truth bounding boxes*). É a porcentagem de predições

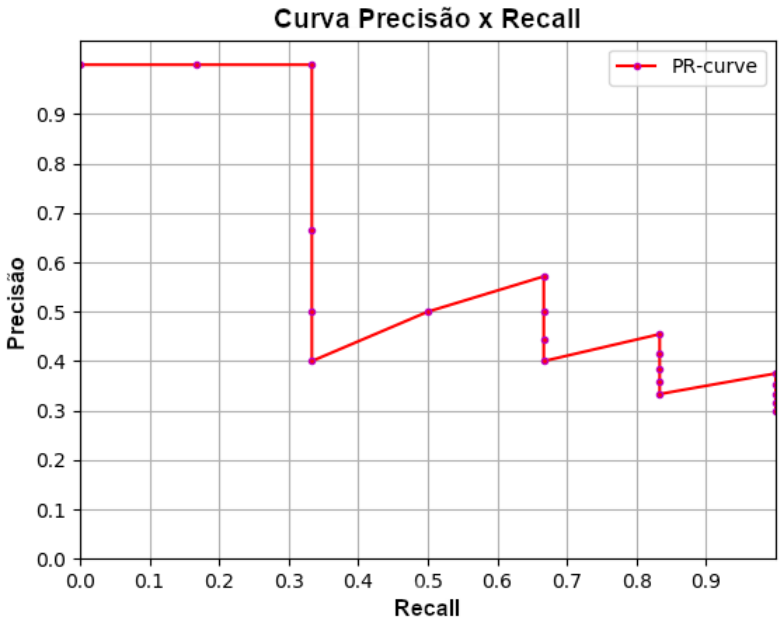
positivas corretas dentre todas as *ground truths* informadas. A Sensibilidade é calculada pela Equação 4.

$$R = \frac{VP}{VP + FN} = \frac{VP}{todas\ as\ ground\ truths} \tag{5}$$

2.7.2 *Curva Precisão x Recall*

Uma curva Precisão x *Recall* é construída calculando e traçando a precisão, no eixo vertical, com relação ao *recall*, no eixo horizontal, para um único classificador em uma variedade de limiares de confiança. A relação mais relevante entre o *recall* e a precisão pode ser observada na área sob os "degraus" do gráfico (BOYD; ENG; PAGE, 2013; COOK; RAMADAS, 2020). Comparando os máximos e mínimos de precisão e também a área sob a curva, é possível analisar por exemplo, a taxa de falsos positivos, falsos negativos, e a precisão geral do sistema. No Gráfico 1 há um exemplo de uma curva de *Precisão* x *Recall* construída a partir dos pontos de precisão e *recall* obtidos variando-se o limiar de confiança.

Gráfico 1 - Degraus da curva de Precisão x *Recall.*



Fonte: Adaptado de Zhang, 2018

A partir da curva P*recisão* x *Recall* de um dado modelo de detector de objetos, é possível avaliá-lo como um bom detector se sua precisão se mantém alta enquanto seu *recall* aumenta. Isso significa que se o limiar de confiança varia, a precisão e o *recall* continuam altos. Portanto, uma área sob a curva de valor elevado costuma indicar que o modelo possui alta precisão e alto *recall*. A área sobre a curva, por si mesma, é uma métrica de desempenho chamada de precisão média (AP, do inglês *average precision*), que é extensivamente utilizada para avaliação de modelos de detectores de objetos (PADILLA et al., 2020).

2.7.3 *Average Precision e Mean Average Precision*

Uma das maneiras mais utilizadas para se comparar o desempenho de modelos de detectores de objetos é calcular a área sob a curva (AUC, do inglês *area under curve*) do gráfico de Precisão x *Recall*, dessa forma, obtém-se a AP, que é a AUC, do modelo. Como as curvas de Precisão x *Recall* normalmente possuem uma configuração em zigue-zague, como mostrado no Gráfico 1, existe uma certa dificuldade em compará-las diretamente. Por isso, é utilizada a AP para realizar uma comparação mais exata entre os diferentes modelos de detectores. Na prática, calcula-se a AP pela média dos valores de precisão associados aos valores de *recall* entre 0 e 1 (PADILLA et al., 2021).

Para o cálculo da AP, podem ser utilizadas duas abordagens: a abordagem da interpolação de 11 pontos e a abordagem da interpolação de todos os pontos.

2.7.3.1 Interpolação de 11 pontos

A ideia da interpolação de 11 pontos é sumarizar o formato da curva de Precisão x *Recall* fazendo a média da precisão em um conjunto de 11 valores de *recall* de 0 a 1, igualmente espaçados em intervalos de [0,1], como pode ser visto na Equação 4.

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,\ldots,1\}} \rho_{interp}(r) \quad \text{, sendo } \rho_{interp} = \max_{\tilde{r}:\tilde{r} \geq r} \rho(\tilde{r}) \tag{6}$$

Onde $\rho(\tilde{r})$ é a precisão medida no *recall* $\tilde{r}$.

Nessa definição de AP, ao invés de se utilizar a precisão $\rho(r)$ obtida em cada valor de *recall r*, a AP é obtida interpolando-se a precisão apenas nos 11 valores de r, tomando a maior precisão associada ao valor de *recall* maior ou igual a *r*.

2.7.3.2 Interpolação utilizando todos os pontos

Como mostrado na Equação 6 e na Equação 7, na interpolação de todos os pontos, ao invés de se interpolar apenas 11 pontos igualmente espaçados, efetua-se a interpolação de todos os pontos n da curva de Precisão x *Recall*:

$$AP_{all} = \sum_{n} (r_{n+1} - r_n)\rho_{interp}(r_{n+1}) \tag{7}$$

onde:

$$\rho_{interp}(r_{n+1}) = \max_{\tilde{r}:\tilde{r} \geq r_{n+1}} \rho(\tilde{r}) \tag{8}$$

Agora, consegue-se obter a AP através da interpolação da precisão a cada nível, tomando a maior precisão com valor de *recall* maior ou igual a r_{n+1} (PADILLA et al., 2021).

2.7.3.3 *Mean Average Precision* (mAP)

A média da precisão média (mAP, do inglês *mean average precision*) é a métrica utilizada para medir a precisão de um detector de objetos sobre todas as classes, dentro de uma base de dados específica. De forma simplificada, a mAP é a média de AP. Em alguns casos, mAP pode ser obtida computando-se a AP para cada classe e tirando sua média, como mostrado na Equação 8, sendo *N* o número de pontos computados e AP_i a AP de cada ponto *i*.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{9}$$

É importante ressaltar que, em alguns contextos, mAP e AP possuem o mesmo significado. Por exemplo, no site do popular *dataset COCO*, considera-se que mAP e AP são termos equivalentes (RIZZOLI, 2022).

2.7.4 *Average Recall*

A Sensibilidade Média (do inglês, *Average Recall*) (AR) é uma métrica de avaliação utilizada para medir a assertividade de um detector de objetos para uma dada classe. Ao contrário da precisão média, os *confidence scores* não são levados em conta no cálculo do *recall* médio.

A AR faz uma avaliação contabilizando todos os valores de *recall* obtidos para limiares de IOU no intervalo [0,5 , 1]. Uma IOU de 0,5 pode ser interpretada como uma localização grosseira de um objeto, e é a menor IOU aceitável pela maioria das métricas, enquanto uma IOU de 1 corresponde à localização perfeita de um objeto detectado. Assim, ao ser fazer uma média dos valores de *recall* no intervalo [0,5 , 1], avalia-se a condição em que o modelo é consideravelmente preciso (PADILLA, 2021).

Seja o a sobreposição IOU entre uma *ground truth* e uma *bounding box* detectada, como calculado na Equação 1, e $R_{C_{IOU}}(o)$ uma função que retorna o *recall* para uma dada IOU o. Na Equação 9, a AR é definida como duas vezes a área sob a curva de $R_{C_{IOU}}(o)$ para um intervalo de IOU de [0,5 , 1], ou seja:

$$AR = 2\int_{0,5}^{1} R_{C_{IOU}}(o)\, do \tag{10}$$

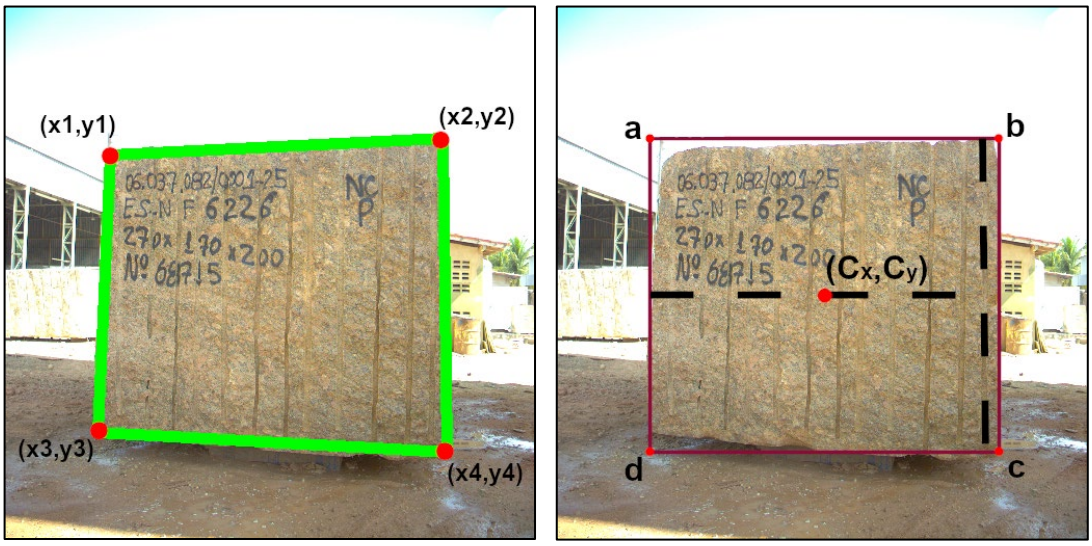
3 METODOLOGIA

3.1 Base de Dados

As imagens que compõem a base de dados foram fornecidas por uma empresa exportadora de granito e mármore. Estas imagens foram fotografadas em campo, em um pátio de carga e descarga, onde pode-se observar variadas condições de iluminação e fundo. Destaca-se que nas imagens amostradas as rochas possuem formato aproximado de paralelepípedo, algumas com textura lisa, mas sendo a maioria de textura irregular e áspera. Foram utilizadas para treinamento dos modelos apenas as imagens que contém uma única face da rocha vista de frente e normalmente centralizada na imagem, como forma de padronização. Ao todo, o *dataset* é composto de 339 imagens anotadas.

Como mostra a Figura 14, para serem empregadas no treinamento dos modelos, as imagens foram anotadas manualmente delineando-se a face exposta da rocha ao longo de quatro vértices, formando um quadrilátero. Tendo os quatro vértices (x_n, y_n) com $n = [1,2,3,4]$, as caixas delimitadoras foram estabelecidas na imagem de forma que o comprimento da caixa corresponde à maior distância x entre dois vértices (x_n, y_n) e a altura corresponde à maior distância y entre dois vértices (x_n, y_n).

Figura 14 - Determinação das *bounding boxes*



(a) Marcação dos quatro vértices (b) Coordenadas da *bounding box*

Fonte: Produção do próprio autor

Assim, os quatro vértices a, b, c e d serão determinados pelas Equações 10,11,12 e 13:

$$x_a = x_d = \min(x_1, x_2, x_3, x_4) \tag{11}$$

$$y_a = y_b = \min(y_1, y_2, y_3, y_4) \tag{12}$$

$$x_b = x_c = \max(x_1, x_2, x_3, x_4) \tag{13}$$

$$y_d = y_c = \max\,(y_1, y_2, y_3, y_4) \tag{14}$$

O centro da *bounding box* é calculado pela média dada na Equação 14.

$$c_x = \frac{x_b + x_a}{2}\,, c_y = \frac{y_a + y_d}{2} \tag{15}$$

A partir do conjunto original, com as imagens anotadas manualmente, gerou-se três novos conjuntos: um no formato YOLO Darknet TXT para uso na *YOLOv5*, e outro no formato TFRecord para uso no modelo *SSD MobileNet V1 FPN* de resolução 640 × 640. A conversão foi realizada a partir de scripts modificados do GitHub (https://github.com/mwindowshz/YoloToTfRecords) e/ou criados pelo autor. No caso da *YOLOv5m-P6*, as imagens foram redimensionadas para a resolução padrão de entrada de 1280 × 1280 pixels. Com a *SSD MobileNet V1*, utilizou-se a resolução padrão de 640 × 640 pixels.

3.1.1 *Aumento de dados*

Tendo em vista o número relativamente baixo de imagens no conjunto original (339), é possível que exista uma tendência a *overfitting* no treinamento do modelo, o que poderia prejudicar o desempenho do modelo ao lidar com novas imagens obtidas no mundo real. Com o fim de aumentar a diversidade das imagens presentes no *dataset* original e melhorar a robustez do modelo, empregou-se a técnica de aumento de dados (do inglês, *data augmentation*). O aumento de dados é uma prática já bem estabelecida no treinamento de modelos que trabalham com Visão Computacional (SHORTEN; KHOSHGOFTAAR, 2019).

Com a aplicação do aumento de dados, o *dataset* de treino foi ampliado de 113 para 922 imagens, totalizando: 922 imagens de treino, 113 imagens de validação e 113 imagens de teste. Foram mantidas as imagens originais nos *datasets* de validação e teste, não tendo sido

adicionadas a estes *datasets* imagens resultantes de aumento de dados. Os procedimentos de *data augmentation* foram realizados através da biblioteca *Albumentations* (BUSLAEV et al., 2020), disponível gratuitamente.

Para obter o aumento de dados, aplicou-se nas imagens de forma aleatória, sendo possível o acúmulo de mais de uma transformação, as transformações descritas no Quadro 1.

Quadro 1 – Transformações de aumento de dados aplicadas

TRANSFORMAÇÕES DE DATA AUGMENTATION	
Transformação	Valor ou tipo
Suavização Gaussiana	Máscara 3 × 3
Espelhamento	Vertical
Variação de brilho	-30% a +30%
Variação de saturação	-30% a +30%

Fonte: Produção do próprio autor

Na Figura 15 pode-se observar um mosaico, criado apenas a caráter ilustrativo, que exibe alguns exemplos de imagens geradas através do aumento de dados que compõem o *dataset* de treino.

Figura 15 - Exemplos de aplicação de operações de aumento de dados

Fonte: Produção do próprio autor.

3.2 Seleção dos modelos

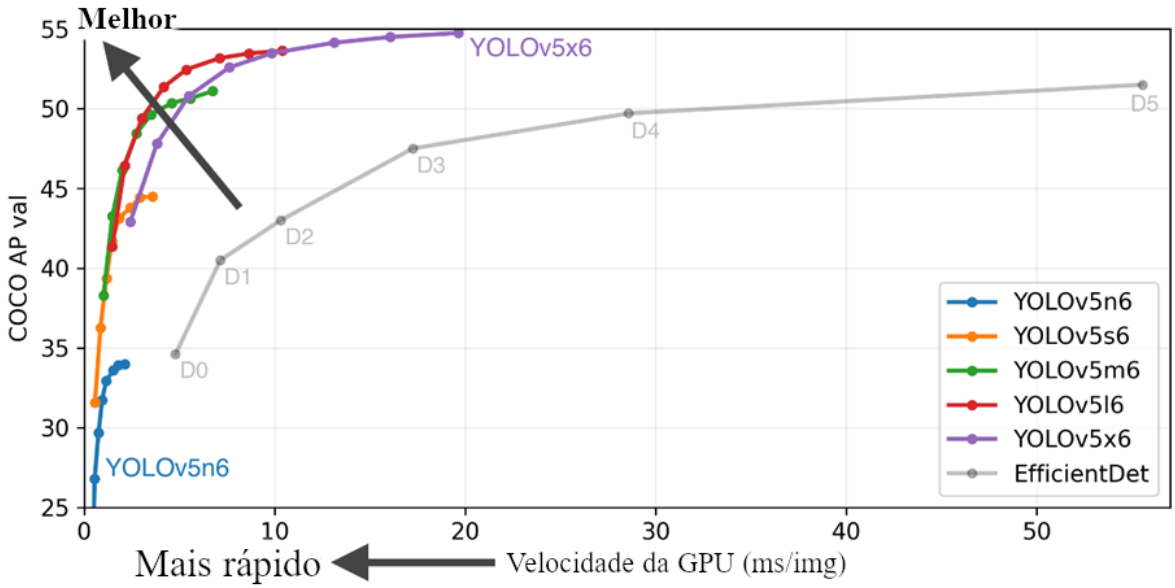
Como já visto anteriormente, existem diversos modelos de redes neurais do tipo detector de objetos derivados dos algoritmos YOLO e SSD. Nesta seção, será feita uma breve explanação das versões específicas dos modelos selecionados e os motivos de sua escolha.

3.2.1 *YOLOv5m-P6*

Dentre as versões da *YOLOv5* desenvolvidas por Jocher et al., 2020, a família de redes P6 foi desenvolvida com o intuito de alcançar um melhor desempenho na detecção de objetos grandes. Tendo em vista a parcela da área da imagem ocupada pelos pixels dos objetos a serem identificados no trabalho presente (blocos de granito e mármore), esta é uma otimização bem-vinda.

Na família de redes P6 da *YOLOv5* existem 4 diferentes tamanhos de redes: a *YOLOv5s-P6* (pequena), a *YOLOv5m-P6* (média), a *YOLOv5l-P6* (grande) e a *YOLOv5x-P6* (extra grande). Uma comparação do desempenho entre as redes pode ser vista no Gráfico 2. Cada ponto do gráfico representa um par de valores de tempo de inferência (ms) × AP (%) para determinados valores de resolução de imagem de entrada, variando de 256 × 256 pixels a 1536 × 1536 pixels.

Gráfico 2 - Comparação de desempenho das redes *YOLOv5* de diferentes tamanhos



Fonte: Adaptado de Ultralytics

Para o desenvolvimento de um modelo de detecção de objetos neste trabalho, foi escolhida a rede de tamanho médio, a *YOLOv5m-P6*, pois busca-se uma rede com um bom balanço entre tempo de inferência e precisão média (AP).

3.2.2 *SSD MobileNet V1 FPN 640x640*

O modelo baseado na arquitetura SSD escolhido foi a *SSD MobileNet V1 FPN* 640 × 640. A arquitetura *SSD MobileNet* utiliza a *MobileNet* como rede de base, funcionando como extratora de características. A *MobileNet V1* é um tipo de rede convolucional desenvolvida com o intuito de ser utilizada em dispositivos móveis e embarcados. Ela se baseia em uma arquitetura simplificada que usa convoluções separáveis em profundidade para compor redes neurais profundas leves e de baixa latência (HOWARD et al., 2017).

Dentre os 41 modelos pré-treinados no *dataset* COCO 2017 disponíveis no *Model Zoo* da API de Detecção de Objetos do TensorFlow 2 (ABADI et al., 2015), foi escolhida a rede *SSD MobileNet V1 FPN 640 × 640*. Como pode ser visto na Tabela 2, comparada às demais redes *SSD MobileNet* oferecidas no *Model Zoo*, esta apresenta um bom equilíbrio entre tempo de inferência e mAP.

Tabela 2 - Desempenho dos modelos SSD

Nome do modelo	Tempo de inferência (ms)	COCO mAP
SSD MobileNet v2 320x320	19	20.2
SSD MobileNet V1 FPN 640x640	48	29.1
SSD MobileNet V2 FPNLite 320x320	22	22.2
SSD MobileNet V2 FPNLite 640x640	39	28.2

Fonte: Adaptado de Yu et al., 2020

3.3 Ambiente de desenvolvimento e frameworks

O *Google Colaboratory* (GOOGLE, 2020), também conhecido como *Colab*, é um ambiente de desenvolvimento integrado (do inglês, *Integrated Development Environment*) (IDE) do Google para fins de apoio à pesquisa e ao aprendizado de Inteligência Artificial. O *Colab* disponibiliza

um ambiente de programação em *Jupyter Notebook,* com acesso gratuito a uma GPU. O *Google Colaboratory* possui já pré-instalados bibliotecas e *frameworks* popularmente utilizados em tarefas relacionadas a *Deep Learning*, como o Tensorflow, Pytorch, Keras e OpenCV.

Como o treinamento de modelos de *machine learning* exige do sistema alta velocidade e poder de processamento (normalmente baseado em GPU), computadores normais muitas vezes se mostram insuficientes para a tarefa. Para prover um ambiente de desenvolvimento adequado à pesquisa de *Deep Learning*, o *Colab* fornece uma instância na nuvem correspondente a uma máquina virtual com as seguintes especificações: (i) Sitema Operacional Linux, (ii) Processador Intel Xeon 2-Core 2,30 GHz, (iii) Memória RAM de 12 GB e (iv) placa de vídeo NVIDIA Tesla T4 de 16 GB. É necessário ressaltar que o *Colab* possui um limite de tempo de 12 horas consecutivas para utilização de uma única sessão, o que é um fator limitante crucial no planejamento do treinamento dos modelos.

Assim como nas versões anteriores, a família de arquiteturas *YOLOv5* foi disponibilizada gratuitamente via repositório no GitHub (ULTRALYTICS, 2022). Nessa versão, a companhia Ultralytics foi a responsável pela conversão da YOLO para o framework Pytorch, que é escrito na linguagem de programação Python. São disponibilizados pela Ultralytics modelos YOLO pré-treinados, além de scripts para treinamento, avaliação, customização e implantação dos modelos.

A arquitetura MobileNet é implementada na API de Detecção de Objetos, também disponível no GitHub (TENSORFLOW, 2022), um *framework* desenvolvido pelo Google construído sobre o Tensorflow 2 (ABADI et al., 2015). Através da API é possível, construir, treinar e implantar modelos de detecção de objetos, além de carregar variados modelos de detecção de objetos pré-treinados disponíveis gratuitamente.

3.4 Treinamento dos modelos

Após a configuração do *dataset* e ambiente de desenvolvimento, os modelos são treinados através de *scripts* disponibilizados nas API. Para tanto, foram importados modelos pré-treinados de seus respectivos repositórios. Ambos os modelos empregados nesta etapa foram previamente treinados com o *dataset* COCO.

Com o fim de reduzir o tempo das etapas de treinamento e obter um melhor desempenho final do modelo, e considerando o tamanho relativamente reduzido do dataset, foi empregado o método da transferência de aprendizado com congelamento de camadas (do inglês, *transfer learning*). Os dois métodos foram utilizados no lugar do método, mais direto, onde se treina a rede por completo, o que normalmente exige um conjunto de dados maior e tempo de treinamento mais extenso (GAO et al., 2020).

3.4.1 *Transfer Learning*

O *transfer tearning* de redes pré-treinadas é uma abordagem muito bem estabelecida para o treinamento de um novo modelo através do uso de modelos de CNN pré-treinados (PAN; YANG, 2009). Essa estratégia é um método de aprendizado eficiente, pois evita que a rede seja treinada do princípio, o que gera uma economia de tempo e maior estabilidade no decorrer do treinamento da rede. Levando-se em conta o limite de tempo de uso da máquina virtual gratuita do ambiente de treinamento, esse é um ponto positivo bastante relevante.

Tradicionalmente, após a seleção do modelo pré-treinado mais adequado, realiza-se o *transfer learning* com as camadas de detecção sendo substituídas por novas camadas de detecção customizadas para o novo problema. Para o caso deste projeto, as camadas mais altas (profundas) não foram substituídas, pois decidiu-se manter integralmente os modelos originais em ambos os casos, para a *YOLOv5* e para a *MobileNet V1*.

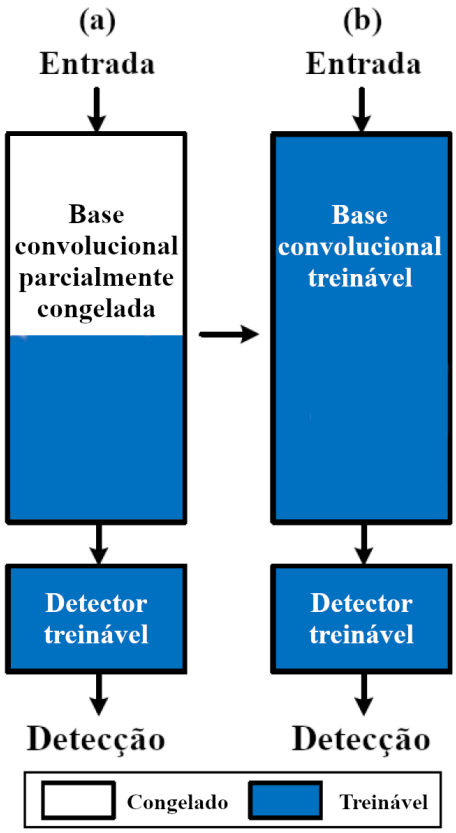
Junto ao processo de *transfer learning*, utilizou-se também no treinamento o chamado congelamento de camadas. O congelamento de camadas é realizado se fixando os pesos das camadas congeladas, excluindo-as do processo de treinamento. Este processo leva a uma redução do tempo de treinamento do modelo, além de conservar as características gerais extraídas nas camadas mais baixas (rasas).

No caso presente, foi congelada parte das camadas mais baixas da rede, o *backbone* convolucional. De acordo com Gao et al., 2020, o congelamento de parte do *backbone* é especialmente interessante ao se considerar a situação na qual os modelos estão sendo desenvolvidos: o *dataset* é relativamente pequeno, mesmo com o aumento de dados, e há baixa similaridade com o *dataset* no qual os modelos pré-treinados foram gerados.

Realizado o treinamento dos modelos com os *backbones* congelados, geraram-se novos modelos com os pesos das camadas mais altas treinadas. Em seguida, todas as camadas destes modelos passaram por um segundo processo de treinamento, no chamado ajuste fino (do inglês, *fine-tuning*), onde aplica-se uma taxa de aprendizado (do inglês, *learning rate*) baixa, da ordem de 10^{-3}, por um número menor de épocas.

O *fine-tuning* pode melhorar ainda mais o desempenho dos modelos, ao mesmo tempo que evita distorções significativas das características aprendidas pelas camadas convolucionais inferiores devido à retro-propagação do sinal de erro através destas camadas. Na Figura 16 tem-se um esquemático das etapas de treinamento dos modelos, com treinamento com o *backbone* congelado (a), seguido do *fine-tuning* (b).

Figura 16 - Congelamento de camadas e *fine-tuning*



Fonte: Adaptado de Khademi G.; Dan S., 2019

3.5 Avaliação de desempenho

Com os modelos *YOLOv5* e *MobileNet V1* treinados, o próximo passo é a realização de testes para que possam ser averiguados os seus respectivos desempenhos. Tanto a API de Detecção de Objetos, onde treinou-se a *MobileNet V1*, quanto o repositório da *YOLOv5* fornecem ferramentas e *scripts* para avaliação dos modelos, com métricas e representações gráficas próprias.

As ferramentas de avaliação nativas de ambas as arquiteturas possuem clara utilidade na comparação de desempenho entre diferentes modelos originários da mesma arquitetura. Porém, é necessário que se empregue uma ferramenta de avaliação universal, que possa gerar métricas a partir dos mesmos algoritmos e operações matemáticas, pois não se pode garantir que os resultados dos métodos de avaliação originais das duas arquiteturas sejam implementados da mesma maneira internamente.

Para executar uma análise de ambos modelos, utilizou-se o conjunto de ferramentas (do inglês, *toolkit*) de código aberto *Object Detection Metrics* (PADILLA et al., 2021). Conforme ilustrado na Figura 17, o *toolkit* opera a partir das detecções realizadas pelos modelos utilizando as imagens de entrada, as coordenadas das *bounding boxes* de *ground truth* (em vermelho), e as coordenadas das *bounding boxes* resultantes das detecções nas mesmas imagens de entrada (em verde). Como o *toolkit* trabalha com os resultados de detecção diretos, ele independe das métricas de avaliação internas das API, mostrando-se, portanto, uma ferramenta de avaliação universal.

Figura 17 - *Bounding boxes* de *ground truth* (em vermelho) e detecção (em verde).

Fonte: Produção do próprio autor

Atualmente, o *toolkit* oferece 12 métricas diferentes, além de gerar gráficos de Precisão x *Recall* para cada classe. Abaixo, tem-se as métricas utilizadas para avaliação dos modelos neste trabalho, que são as métricas padrões de avaliação do *dataset* COCO correspondentes à precisão (excluindo as métricas de *recall*):

1. AP@.5 e AP@.75: Essas duas métricas avaliam a curva de Precisão x *Recall* com interpolação de N = 101 pontos de *recall*, o que difere do algoritmo utilizado para o dataset PASCAL VOC. AP@.5 significa que o limiar de IOU utilizado para o cálculo é da AP é de t = 0,5. Por sua vez, para o cálculo de AP@.75 utiliza-se t = 0,75.
2. AP@[.5:.05:.95]: Essa métrica estende a métricas AP@.5 e AP@.75 ao computar a AP em dez diferentes limiares de IOU (t = [0,5, 0,55, ..., 0,95]) e tomar a média entre todos os resultados computados. Também é usualmente abreviada como AP.
3. AP_S, AP_M e AP_L: Também chamadas de *AP Across Scales* (AP através de escalas), essas três métricas aplicam a AP@[.5:.05:.95] levando em consideração o tamanho da *ground truth* do objeto. AP_S avalia apenas os objetos de *ground truth* de pequeno tamanho (área $< 32 \times 32$ pixels); AP_M considera apenas a *ground truth* de objetos de tamanho médio

(32×32 pixels $<$ área $<$ 96×96 pixels); AP_L considera apenas objetos de *ground truth* com área $>$ 96×96 pixels.

4. AR_S, AR_M e AR_L: Também conhecidas como *AR Across Scales* (AR através de escalas), essas métricas são definidas da mesma maneira que a *AP Across Scales*, porém computando o *recall* médio no lugar da precisão média.

4 EXPERIMENTOS E RESULTADOS

Para o desenvolvimento dos modelos, foram utilizados códigos em formato de notebook Jupyter, que estão disponíveis no GitHub do autor deste trabalho: https://github.com/vtrv123/

4.1 Primeira etapa de treinamento

Tendo os *datasets* carregados em seus respectivos ambientes de desenvolvimento, inicia-se o treinamento dos modelos. Para isso, é necessário que se estabeleçam os hiperparâmetros de treinamento. O Quadro 2 apresenta os principais hiperparâmetros usados em ambos os modelos, na primeira etapa de treinamento, com parte dos pesos dos *backbones* das redes congelados.

Quadro 2 - Hiperparâmetros utilizados nos treinamentos dos modelos, com parte dos pesos congelados

HIPERPARÂMETROS – *BACKBONE* COM PARTE DOS PESOS CONGELADOS		
Parâmetro	Valor (*SSD MobileNet*)	Valor (*YOLOv5*)
Taxa de aprendizado	0,04	0,015
Batch size	16	20
Épocas	337 (15000 *steps*)	300
β_1 (Adam)	0,9	0,937
Warmup epochs	45	3

Fonte: Produção do próprio autor

No Quadro 2, pode-se observar o hiperparâmetro *warmup epochs*. Definindo de forma resumida, as épocas de aquecimento (do inglês, *warmup epochs*) são o número de épocas n através das quais haverá um acréscimo linear p/n da taxa de aprendizado durante o treinamento, até atingir o valor nominal p da taxa de aprendizado na época n. Normalmente utiliza-se o método das épocas de aquecimento com otimizadores adaptativos (Adam, RMSProp, etc.) para ajudar a computar corretamente as estatísticas dos gradientes no início do treinamento da rede (ZHANG, 2021).

Nesta etapa, a taxa de aprendizado, o valor de β_1 para o otimizador Adam, e o número de épocas de aquecimento (do inglês, *warmup epochs*) foram mantidos de acordo com os valores padrão dos arquivos de configuração de ambos os modelos *SSD MobileNet* e *YOLOv5*. O *batch size* foi escolhido empiricamente para aproveitar ao máximo a capacidade de memória da GPU (NVIDIA Tesla T4) disponibilizada pelo *Google Colaboratory*. O número de épocas foi

escolhido de acordo com o tempo de treinamento possível dentro da janela de 12 horas oferecida pelo ambiente de treinamento. Antes do início do treinamento, parte dos *backbones* foram congelados em ambos os modelos, fixando os pesos das camadas extratoras de características menos profundas.

Os dois modelos levaram um período de tempo de cerca de 6 a 8 a horas de treinamento. De acordo com a avaliação gerada pelas ferramentas nativas de cada *framework*, a *SSD MobileNet* atingiu uma AP de 0,954 enquanto a *YOLOv5* apresentou uma AP 0,965. É necessário ressaltar que tais medidas foram aferidas de acordo com o *dataset* de validação, e que foram geradas a partir de ferramentas diferentes, originárias da API de Detecção de Objetos no caso da SSD, e da API da Ultralytics no caso da *YOLOv5*. Portanto, não podem ser usadas para basear conclusões definitivas. De qualquer forma, os bons resultados apontados por estas métricas indicam positivamente que os modelos convergiram de forma satisfatória.

4.2 Fine-tuning

Seguindo a primeira etapa de treinamento, procura-se aprimorar o desempenho dos modelos através de uma nova fase de treinamento. Nesta fase, as camadas congeladas dos *backbones* dos modelos são descongeladas, o que significa que todos os pesos dos modelos estão sujeitos a treinamento. Em seguida, é efetuado o chamado *fine-tuning*: são feitos ajustes nos hiperparâmetros no intuito de retreinar o modelo de forma leve e gradativa, buscando-se adaptar o *backbone* descongelado aos novos dados de entrada (lembrando que o *backbone* congelado havia sido treinado no *dataset* COCO) em conjunto com as camadas detectoras. Quando fora aplicada a técnica de *fine*-tuning esperava-se promover melhoras na acurácia dos modelos, mesmo que estas fossem melhoras sutis.

Ao se realizar o *fine-tuning*, o principal parâmetro a ser ajustado é a taxa de aprendizado. Existem diferentes abordagens práticas para definir o novo valor da taxa de aprendizado, como escolher uma taxa de aprendizado em um valor na ordem de 10^{-4} (PENG; WANG, 2020), ou na ordem 10^{-5} (CHOLLET, 2020), ou um valor dez vezes menor do que a taxa de aprendizado da primeira etapa de treinamento (LU et al., 2017).

Para este trabalho, optou-se por utilizar uma taxa de aprendizado uma ordem de grandeza menor do que a taxa de aprendizado da etapa inicial. Tal escolha foi feita levando-se em consideração a existência de arquivos de configuração de treinamento para diferentes *datasets* com hiperparâmetros padronizados, disponibilizados no repositório da *YOLOv5*, que implementavam uma taxa de aprendizado uma ordem de grandeza menor do que a taxa de aprendizado anterior. Para este caso, foi escolhido o arquivo de configuração de treinamento da da *YOLOv5* para o *dataset* Pascal VOC (EVERINGHAM et al., 2010), cujos principais hiperparâmetros podem ser vistos no Quadro 3.

A opção pelos arquivos padronizados, os arquivos de configuração padrão importados junto com os modelos pré-treinados, deve-se à tendência de que os hiperparâmetros além da taxa de aprendizado tenham sido otimizados junto com a taxa de aprendizado. Ao invés de se optar por alterar apenas a taxa de aprendizado, escolhe-se utilizar os arquivos padrão. Além disso, os valores de taxa de aprendizado de 10^{-4} e 10^{-5} estariam muito distantes dos valores de taxa de aprendizado na primeira etapa de treinamento. Por fim, ressalta-se que na segunda etapa a taxa de aprendizado da *SSD MobileNet* foi ajustada para um valor próximo da taxa de aprendizado aplicada na *YOLOv5*.

O Quadro 3 apresenta os hiperparâmetros utilizados na etapa de *fine-tuning* dos modelos *SSD MobileNet* e *YOLOv5*.

Quadro 3 -- Hiperparâmetros utilizados no *fine-tuning* dos modelos

HIPERPARÂMETROS – *FINE-TUNING*		
Parâmetro	Valor (*SSD MobileNet*)	Valor (*YOLOv5*)
Taxa de aprendizado	0,003	0,00334
Batch size	16	20
Épocas	222,47 (10000 *steps*)	200
β_1 (Adam)	0,9	0,74832
Warmup epochs	20	3,3835

Fonte: Produção do próprio autor.

4.3 Avaliação do desempenho

4.3.1 *Métricas*

Estando os modelos treinados e disponíveis para inferência, modificou-se os *scripts* das APIs para gerar as coordenadas das *bounding boxes* para cada respectiva imagem de entrada. Com as imagens de entrada, as coordenadas das *bounding boxes* de *ground truth* e as coordenadas das *bounding boxes* das detecções, pode-se utilizar o *toolkit Object Detection Metrics* para extrair os valores das métricas de desempenho para avaliação. No Quadro 4, têm-se os valores das métricas de desempenho do modelo na arquitetura *YOLOv5* ao lado dos valores das métricas de desempenho do modelo na arquitetura *SSD MobileNet*.

Quadro 4 - Métricas de desempenho obtidas com o *toolkit Object Detection Metrics*

MÉTRICAS DE DESEMPENHO		
Métrica	Valor (*SSD MobileNet*)	Valor (*YOLOv5*)
AP@[.5:.05:.95]	0,9604	0,9693
AP@.5	1,000	0,9901
AP@.75	0,9898	0,9901
AP_L	0,9604	0,9693
AR_L	0,9663	0,9770

Fonte: Produção do próprio autor

Faz-se necessário observar que as métricas exibidas são a Precisão Média (AP) e não a Média da Precisão Média (mAP), que costuma ser o padrão de avaliação de modelos detectores de objetos. Sabe-se que, no caso de um detector de classe única, na Equação 7 há N = 1, o que significa que $mAP = AP$, portanto, para os detectores analisados, mAP e AP são equivalentes.

Nota-se, também, que foram omitidos os valores de AP_S, AP_M, AR_S e AR_M. Os resultados exibidos para estas métricas foram "nan", ou "número não-disponível", o que significa na prática que não haviam objetos com $área \leq 96\ x\ 96\ pixels$ tanto na *ground truth* quanto nas detecções.

Em ambos os modelos, a AP para as IOUs de 0,5 e 0,75 foi muito próxima de 1. Isso mostra que, para estes valores de IOU, a precisão é praticamente constante para todos os valores de *Recall* e é igual a aproximadamente 1, ou seja, com a curva de Precisão x *Recall* tomando um formato próximo do de um detector ideal. Com a YOLO, o valor de AP@[.5:.05:.95], ou

simplesmente AP, é igual a 0,9693, com a *SSD MobileNet* também alcançando o valor muito próximo de 0,9604. Isso demonstra que os detectores possuem um bom desempenho, sendo praticamente igual entre os dois, em vários limiares diferentes de IOU.

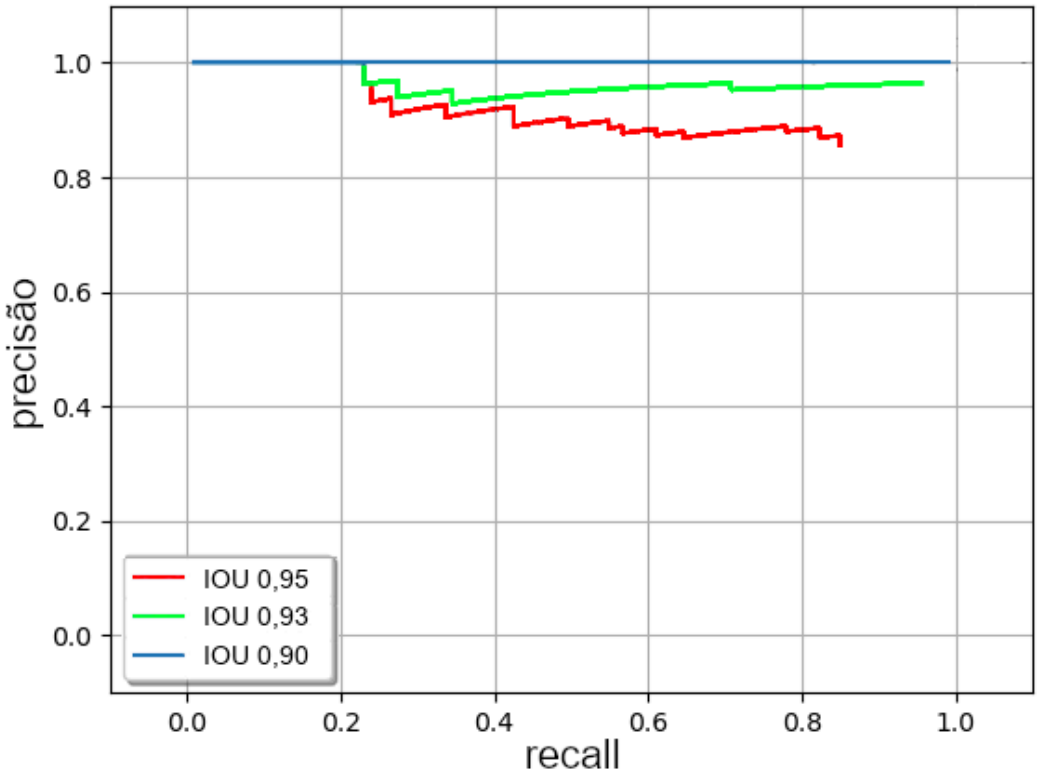
Tanto a *YOLOv5* quanto a *SSD MobileNet* obtiveram bons resultados de *recall* médio, alcançando valores acima de 0,95, o que significa um bom *recall* das propostas de objetos e precisão de localização ao longo do intervalo de limiares de IOU.

4.3.2 *Gráficos*

Variando os valores de limiar de confiança (do inglês, *confidence threshold*) e obtendo os valores de precisão e *recall*, o *toolkit* Object Detection Metrics também é capaz de gerar gráficos de Precisão x *Recall* para cada modelo. Busca-se com a análise das curvas de Precisão x R*ecall*, encontrar valores de *confidence threshold* e de IOU para funcionamento dos modelos.

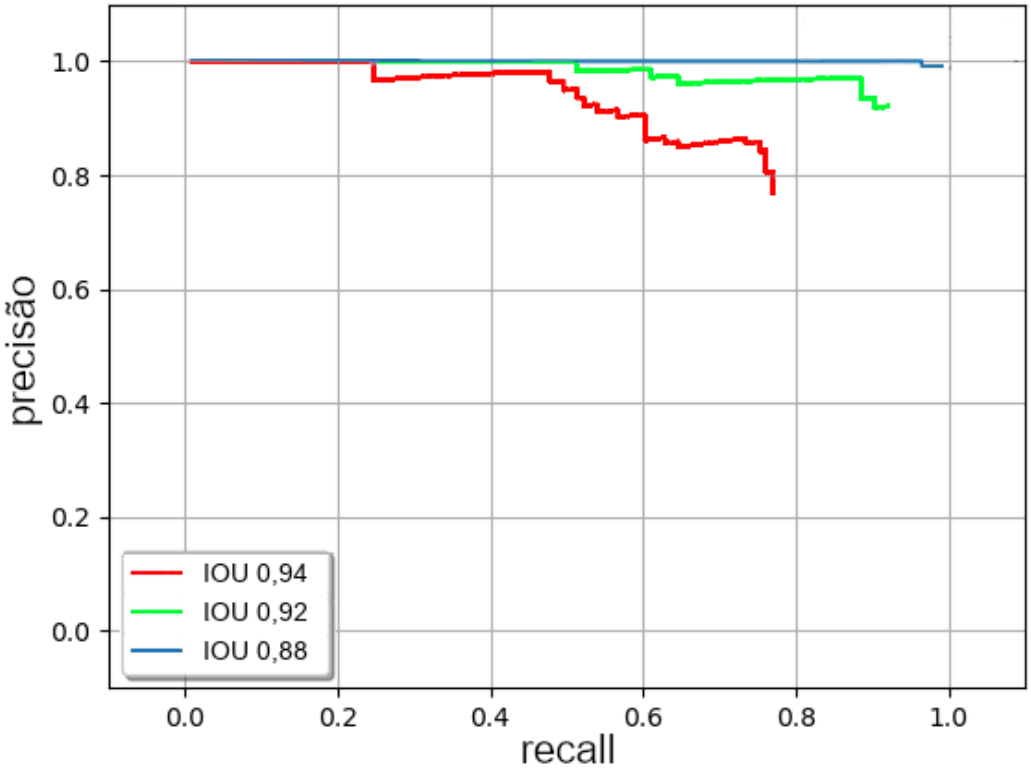
Os Gráfico 3 e Gráfico 4 representam as curvas de Precisão × *Recall* dos modelos *YOLOv5* e *SSD MobileNet*, respectivamente. Os gráficos de cada cor representam uma curva de Precisão × *Recall* para diferentes limiares de IOU. Como já explicado anteriormente, as curvas de Precisão × *Recall* são obtidas contabilizando os valores de precisão para cada *recall* à medida em que varia-se o limiar de confiança.

Gráfico 3 - Curva de Precisão x *Recall* modelo *YOLOv5*



Fonte: Produção do próprio autor

Gráfico 4 – Curva de Precisão x *Recall SSD MobileNet*



Fonte: Produção do próprio autor

Levando-se em consideração os altos valores de AP encontrados para os modelos, percebeu-se na etapa de geração dos gráficos que as curvas de Precisão x *Recall* se mostravam como linhas horizontais com valor constante de precisão igual a 1 para diversos limiares de IOU. Como visto nas curvas em verde e vermelho no Gráfico 3, na *YOLOv5* as curvas de Precisão x *Recall* só deixavam de assumir tal comportamento acima de limiares de IOU com valores próximos de 0,9, onde o gráfico se distorce, pois a precisão não consegue manter o valor de 1 à medida que o *recall* aumenta. O mesmo ocorreu com as curvas de Precisão x *Recall* do modelo *SSD MobileNet*, mostrada no Gráfico 4, que só deixa de representar um valor constante, como na curva azul, nas curvas em vermelho e verde para limiares de IOU superiores a aproximadamente 0,88.

Considerando que as curvas de Precisão x *Recall* se comportam como curvas ideais para ambos os modelos em determinados valores de IOU, é cabível afirmar que um IOU adequado de operação para o modelo *YOLOv5* seja menor ou igual a 0,9 e, para o modelo *SSD MobileNet*, menor ou igual a 0,88. Como busca-se criar uma aplicação de detecção de objetos que seja uma possível etapa de pré-processamento de uma futura aplicação de identificação dos blocos de granito, é importante que os modelos detectem os objetos desejados com a menor presença de fundo possível. Portanto, a escolha mais adequada é o maior valor de IOU para operação dos modelos. Nesta situação, IOU de 0,9 para o modelo *YOLOv5* e IOU de 0,88 para o modelo *SSD MobileNet.*

4.4 Resultados

Efetuando a detecção dos blocos de rochas naturais nas imagens através dos modelos finais, pode-se obter os resultados com as *bounding boxes* demarcadas com a classe e o *score* de confiança. A Figura 18 mostra um exemplo de detecção realizada pelo modelo da arquitetura *YOLOv5*, em uma imagem do *dataset* de teste.

Figura 18 - Exemplo de detecção do modelo *YOLOv5*

Fonte: Produção do próprio autor

Assim como visto no exemplo da Figura 18, o modelo *YOLOv5* apresentou resultados com *confidence scores* bastante elevados. De todas as detecções aplicadas ao *dataset* de teste, o menor *confidence score* encontrado pelo modelo *YOLOv5* foi de 95%, não tendo sido identificados falsos positivos ou falsos negativos. Tal desempenho do modelo condiz com os resultados encontrados pelas métricas de avaliação na seção anterior, com precisão média e *recall* médio consideravelmente elevados. As leves quedas de *confidence score* podem ser observadas nas imagens, como a Figura 18, em que os blocos possuem altura significativamente maior que o comprimento ou com um fundo de brilho elevado.

Na Figura 19 tem-se a detecção realizada pelo modelo *SSD MobileNet* na mesma imagem da Figura 18, que fora detectada pelo modelo *YOLOv5*. Nota-se neste exemplo que o *confidence score* estimado pelo modelo *SSD MobileNet* (100%) é maior que o estimado pelo modelo *YOLOv5* (96%). De forma geral, observou-se que os *confidence score* determinados pela *SSD MobileNet* foram maiores que os *confidence score* estimados pela *YOLOv5* nas imagens de teste.

Figura 19 - Exemplo de detecção do modelo *SSD MobileNet*

Fonte: Produção do próprio autor

Em ambos os modelos, os *confidence score* elevados condizem com o alto desempenho encontrado pelas métricas de avaliação. Tanto para o modelo *SSD MobileNet* quanto para o modelo *YOLOv5*, o menor *confidence score* encontrado foi o de 93% e também não foram identificados falsos negativos nem falsos positivos nas detecções realizadas no *dataset* de teste.

Ambos os modelos demonstraram desempenho muito semelhante. Levando em consideração as métricas de avaliação, o modelo *YOLOv5* apresentou uma AP (0,9693) e AR (0,9770) maiores que o modelo *SSD MobileNet* (AP de 0,9604 e AR de 0,9663) sendo, porém, uma diferença muito pequena. Quanto aos gráficos de Precisão x *Recall*, as curvas do Gráfico 3 correspondentes ao modelo *YOLOv5* se assemelham mais à curva de um modelo ideal (uma reta com valor de precisão constante igual a 1) do que as curvas do Gráfico 4, correspondentes ao modelo *SSD MobileNet*, mesmo para valores de limiar de IOU mais elevados.

Realizada a análise das métricas e dos gráficos, conclui-se que o modelo *YOLOv5* demonstrou desempenho levemente superior ao modelo *SSD MobileNet*. Porém, a depender do contexto e da aplicação, a pequena diferença de desempenho apresentada pelo modelo *SSD MobileNet* pode se tornar algo negligível. Como já dito anteriormente, a rede *SSD MobileNet* é uma rede curta, desenvolvida principalmente com o fim de propiciar uma execução leve e rápida. Sendo o modelo *YOLOv5* aqui desenvolvido uma rede de tamanho médio, espera-se que seu

desempenho seja menos satisfatório em aplicações como dispositivos móveis ou embarcados, que são meios nos quais a *SSD MobileNet* pode se mostrar mais vantajosa.

5 CONCLUSÕES

O objetivo central deste trabalho foi a construção de um modelo detector de objetos baseado em redes neurais profundas, com o fim de realizar a detecção de rochas naturais em imagens obtidas em ambientes não controlados. Foram desenvolvidos dois modelos diferentes, um na arquitetura *YOLOv5* e outro na arquitetura *SSD MobileNet*, porém as técnicas e ambiente de treinamento utilizadas em ambos foram as mesmas. Os dois modelos também foram treinados e avaliados em uma mesma base dados aprimorada com técnicas de aumento de dados, e a ambos foram aplicadas as técnicas de *transfer learning* e *fine-tuning*.

Começando com uma base de dados de 339 imagens, ressalta-se que o procedimento de *data augmentation* foi essencial para compensar o baixo tamanho do *dataset* de treino e possibilitar um treinamento eficaz das redes. Em seguida, o uso da técnica de *transfer learning* com parte das camadas do *backbone* congeladas permitiu treinar os modelos em um espaço de tempo adequado ao disponibilizado no ambiente de treinamento, além de também ajudar a compensar a baixa quantidade de dados trabalhados. A execução de uma segunda etapa de treinamento, o *fine-tuning*, possibilitou que todas as camadas do modelo fossem treinadas, mas evitando alterações bruscas no *backbone* convolucional.

Tendo em vista que os modelos foram desenvolvidos em *frameworks* diferentes, mostrou-se necessário que a avaliação destes fosse realizada por uma ferramenta externa, pois não se podia garantir a equivalência dos métodos de extração e cálculo das métricas de desempenho nativos dos dois *frameworks*. Foram escolhidas as métricas de avaliação padrão do Desafio de Detecção do *Dataset* COCO (do inglês, COCO *Detection Challenge*) para análise dos modelos, pela diversidade de métricas e por estas também serem utilizadas amplamente em estudos para comparação de desempenho entre modelos de redes neurais.

Quanto aos resultados obtidos, o modelo *YOLOv5* obteve melhor desempenho em quase todas as métricas utilizadas, se comparado ao modelo *SSD MobileNet*. No entanto, ambos modelos mostraram um desempenho bastante satisfatório. Como visto no Quadro 4, para os limiares de IOU de 0,5 e 0,75 os modelos obtiveram valores de AP muito próximos de 1. O desempenho dos modelos também pode ser constatado pelos Gráficos 3 e 4, onde as curvas de Precisão x

Recall assumem o comportamento de curvas de modelos ideais em limiares de IOU acima de 0,8.

A rede *SSD MobileNet* é projetada com o intuito de ser utilizada em dispositivos móveis e embarcados, onde normalmente espera-se limitações de poder de processamento e memória. O modelo utilizado foi treinado a partir de uma rede *SSD MobileNet V1 FPN*, uma rede de tamanho pequeno e baixa latência, com resolução 640 × 640, metade da resolução da rede *YOLOv5* empregada. Como um *trade-off* da simplicidade da rede, era esperado que a *SSD MobileNet* apresentasse um desempenho inferior de precisão média, porém, o que se observou foi um desempenho muito próximo do obtido pelo modelo *YOLOv5*.

Como possibilidade de desenvolvimento futuro, pode-se experimentar com o treinamento de outros modelos de redes neurais, disponíveis no *Model Zoo* da API de Detecção de Objetos do *Tensorflow* 2, utilizando a mesma base de dados. De fato, durante o desenvolvimento deste projeto, foi estudada a possibilidade do treinamento de um terceiro modelo de rede neural: A rede *SSD ResNet101 V1 FPN 1024x1024* (*RetinaNet101*). No entanto, não foi possível obter sucesso no desenvolvimento deste terceiro modelo, pois o tempo de uso disponibilizado pelo *Google Colaboratory* não era suficiente para que o modelo pudesse alcançar convergência.

Este projeto também foi elaborado com o intuito de que possa ser utilizado como uma etapa de pré-processamento em trabalhos futuros que executem a reidentificação dos blocos detectados pelo modelo. A partir das coordenadas das *bounding boxes* geradas na detecção do modelo é possível separar a imagem do bloco do fundo, gerando uma imagem contendo apenas uma face da rocha, a qual pode ser utilizada por outra aplicação para identificar suas características e, posteriormente, reidentificá-la em outra imagem.

Adicionalmente, podem ser realizados estudos de utilização dos modelos em situações reais, para estabelecimento de variáveis como tempo de detecção, ocupação de memória e processamento. Este tipo de estudo seria especialmente relevante levando-se em consideração que é possível implementar os modelos aqui desenvolvidos em dispositivos móveis ou embarcados para, até mesmo, testes em campo.

Também é possível abordar o problema de separação do bloco do fundo através da segmentação semântica. Redes neurais como a U-Net (RONNEBERGER; FISCHER; BROX, 2015) a DeepLab (CHEN et al., 2017) e a FastFCN (WU et al., 2019) realizam a segmentação de classes de objetos na imagem, onde teoricamente seria possível separar os pixels correspondendo apenas ao bloco do ambiente de fundo. Caso bem-sucedida, essa abordagem atingiria o mesmo objetivo de redução da influência do ambiente de fundo para uma futura etapa de identificação das rochas.

Por fim, como possibilidade futura, pode ser estudada a utilização de redes mais simples e de tamanho reduzido, visto que é possível que a utilização de redes profundas pode se configurar como uma solução excessivamente complexa. É cabível também supor que redes menores e simples tenham um tempo de resposta mais rápido e exijam menos capacidade de processamento e, por isso, possam ter um custo/benefício mais desejável.

REFERÊNCIAS BIBLIOGRÁFICAS

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.** 2015. Disponível em: www.tensorflow.org. Acesso 13 mar. 2022.

ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. **2017 International Conference on Engineering and Technology (ICET)**, ago. 2017.

ARAÚJO, M. et al. Identification of Granite Varieties from Colour Spectrum Data. **Sensors**, v. 10, n. 9, p. 8572–8584, 14 set. 2010.

ATİK, M. E.; DURAN, Z.; ÖZGÜNLÜK, R.. Comparison of YOLO Versions for Object Detection from Aerial Images. **International Journal of Environment and Geoinformatics**, v. 9, n. 2, p. 87–93, 15 jun. 2022.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation Learning: A Review and New Perspectives. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 35, n. 8, p. 1798–1828, ago. 2013.

BIANCONI, F. et al. Automatic classification of granite tiles through colour and texture features. **Expert Systems with Applications**, v. 39, n. 12, p. 11212–11218, 15 set. 2012.

BIANCONI, F.; FERNÁNDEZ, A. Granite texture classification with Gabor filters. *In:* **Proc. of the 18th International Congress on Graphical Engineering**, Sitges, mai. 2006.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection**. arXiv:2004.10934 [cs, eess]**, 22 abr. 2020.

BOYD, K.; ENG, K. H.; PAGE, C. D. Area under the precision-recall curve: point estimates and confidence intervals. In: **Joint European conference on machine learning and knowledge discovery in databases**. Springer, Berlin, Heidelberg, 2013. p. 451-466.

BUSLAEV, A. et al. Albumentations: fast and flexible image augmentations. **Information**, v. 11, n. 2, p. 125, 2020.

CARRANZA-GARCÍA, M. et al. On the Performance of One-Stage and Two-Stage Object Detectors in Autonomous Vehicles Using Camera Data. **Remote Sensing**, v. 13, n. 1, p. 89, 29 dez. 2020.

CHEN, L. et al. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. **IEEE transactions on pattern analysis and machine intelligence**, v. 40, n. 4, p. 834-848, 2017.

CHIODI, C. **Balanço das Exportações e Importações Brasileiras de Rochas Ornamentais em 2019**. Belo Horizonte - Minas Gerais: Associação Brasileira da Indústria de Rochas Ornamentais - ABIROCHAS, 12 jan. 2020. Disponível em: https://abirochas.com.br/informes-2019/. Acesso em: 13 mar. 2022.

CHOLLET, F. **Complete guide to transfer learning & fine-tuning in Keras.** 2020. Disponível em: https://keras.io/guides/transfer_learning/. Acesso em 14 jun. 2022.

COOK, J.; RAMADAS, V. When to consult precision-recall curves. **The Stata Journal**, v. 20, n. 1, p. 131-148, 2020.

DEEPA, R. et al. Comparison of yolo, ssd, faster rcnn for real time tennis ball tracking for action decision networks. In: **2019 International conference on advances in computing and communication engineering (ICACCE)**. IEEE, 2019. p. 1-4.

DENG, J. et al. A review of research on object detection based on deep learning. *In:* **Journal of Physics: Conference Series**. IOP Publishing, 2020. p. 012028.

EVERINGHAM, M. et al. The Pascal Visual Object Classes (VOC) challenge. **International journal of computer vision**, v. 88, n. 2, p. 303-338, 2010.

FERNÁNDEZ, A. et al. Evaluation of robustness against rotation of LBP, CCR and ILBP features in granite texture classification. **Machine Vision and Applications**, v. 22, n. 6, p. 913–926, 24 fev. 2011.

FERREIRA, A.; GIRALDI, G. Convolutional Neural Network approaches to granite tiles classification. **Expert Systems with Applications**, v. 84, 2017.

FU, C. et al. Dssd: Deconvolutional single shot detector. **arXiv preprint arXiv:1701.06659, 2017.**

GAO, Y. et al. Deep learning and transfer learning models of energy consumption forecasting for a building with poor information data. **Energy and Buildings**, v. 223, p. 110156, 2020.

GIRSHICK, Ross et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2014. p. 580-587.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE A. Deep learning. **Genetic Programming and Evolvable Machines**, v. 19, n. 1-2, p. 305–307, 29 out. 2017.

GOOGLE. **Google Colaboratory**. [S. l.]: 2022. Disponível em: https://colab.research.google.com/notebooks/intro.ipynb. Acesso em: 13 mar. 2022.

HOSANG, J.; BENENSON, R.; SCHIELE, B. Learning non-maximum suppression. *In:* **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2017. p. 4507-4515.

HUANG, G. et al. Densely connected convolutional networks. *In:* **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2017. p. 4700-4708.

HUI, J. **SSD object detection: Single Shot MultiBox Detector for real-time processing**. 2018. Disponível em: https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06. Acesso em 22. mar. 2022

HOWARD, A.-G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. **arXiv preprint arXiv:1704.04861, 2017.**

KÄDING, Christoph et al. Fine-tuning deep neural networks in continuous learning scenarios. In: **Asian Conference on Computer Vision**. Springer, Cham, 2016. p. 588-605.

KANJEE R., **YOLOv4 — Superior, Faster & More Accurate Object Detection**. 2020. Disponível em: https://augmentedstartups.medium.com/yolov4-superior-faster-more-accurate-object-detection-7e8194bf1872. Acesso em 13 mar. 2022.

KARAALI, İ.; EMINAĞAOĞLU, M. A convolutional neural network model for marble quality classification. **Journal of the Faculty of Engineering and Architecture of Gazi University**, v. 36, n. 1, p. 347-357, 2021.

KHADEMI, G.; SIMON, D. Convolutional neural networks for environmentally aware locomotion mode recognition of lower-limb amputees. In: **Dynamic Systems and Control Conference**. American Society of Mechanical Engineers, 2019. p. V001T07A005.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436-444, 2015.

LEPISTÖ, L; KUNTTU I.; VISA A. Rock image classification using color features in Gabor space. **Journal of Electronic Imaging**, v. 14, n. 4, p. 040503, 1 out. 2005.

LI, M et al. Agricultural greenhouses detection in high-resolution satellite images based on convolutional neural networks: Comparison of faster R-CNN, YOLO v3 and SSD. **Sensors**, v. 20, n. 17, p. 4938, 2020.

LI, Z. al. A two-stage industrial defect detection framework based on improved-yolov5 and optimized-inception-resnetv2 models. **Applied Sciences**, v. 12, n. 2, p. 834, 2022.

LIN, T.-Y. et al. Microsoft COCO: Common Objects in Context. **Computer Vision – ECCV** 2014, p. 740–755, 2014.

LIU, W. et al. Ssd: Single shot multibox detector. *In*: **European conference on computer vision**. Springer, Cham, 2016. p. 21-37.

MANAWADU, M.; WIJENAYAKE, U. Voice-Assisted Real-Time Traffic Sign Recognition System Using Convolutional Neural Network. 2021.

MENEGAZ, M. **Understanding YOLO**. 2018. Disponível em: https://hackernoon.com/understanding-yolo-f5a74bbc796. Acesso em 13 mar. 2022

MORERA, A. et al. SSD vs. YOLO for detection of outdoor urban advertising panels under multiple variabilities. **Sensors**, v. 20, n. 16, p. 4587, 2020.

NELSON J.; SOLAWETZ J. **YOLOv5 is Here: State-of-the-Art Object Detection at 140 FPS**. 2020. Disponível em: https://blog.roboflow.com/yolov5-is-here/. Acesso em 22 mar. 2022.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks**. arXiv preprint arXiv:1511.08458**, 2015.

PADILLA, R.; NETTO, S. L.; DA SILVA E. AB. A survey on performance metrics for object-detection algorithms. In: **2020 international conference on systems, signals and image processing (IWSSIP)**. IEEE, 2020. p. 237-242.

PADILLA, R. et al. A comparative analysis of object detection metrics with a companion open-source toolkit. **Electronics (Switzerland)**, v. 10, n. 3, p. 1–28, 1 fev. 2021.

PAN, S.; YANG, Q. A survey on transfer learning. **IEEE Transactions on knowledge and data engineering**, v. 22, n. 10, p. 1345-1359, 2009.

PARICO, A. I. B.; AHAMED, T. Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT. **Sensors**, v. 21, n. 14, p. 4803, 14 jul. 2021.

PARK, S. **A guide to Two-stage Object Detection: R-CNN, FPN, Mask R-CNN**. 2021. Disponível em: https://medium.com/codex/a-guide-to-two-stage-object-detection-r-cnn-fpn-mask-r-cnn-and-more-54c2e168438c. Acesso em 13 mar. 2022.

PENG, Peng; WANG, Jiugen. **How to fine-tune deep neural networks in few-shot learning?**. arXiv preprint arXiv:2012.00204, 2020.

RAWAT, W.; WANG, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. **Neural Computation**, v. 29, n. 9, p. 2352–2449, set. 2017.

REDMON, J. et al. You only look once: Unified, real-time object detection. *In:* **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2016. p. 779-788.

REDMON, J.; FARHADI, A. YOLO9000: better, faster, stronger. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2017. p. 7263-7271.

RIZZOLI, A. **Mean Average Precision (mAP) Explained: Everything You Need to Know**. 2022. Disponível em: https://www.v7labs.com/blog/mean-average-precision. Acesso em 24 mar. 2022.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: **International Conference on Medical image computing and computer-assisted intervention**. Springer, Cham, 2015. p. 234-241.

SHORTEN, C.; KHOSHGOFTAAR, T.-M. A survey on image data augmentation for deep learning. **Journal of big data**, v. 6, n. 1, p. 1-48, 2019.

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

SINGH, N.; SINGHSAINI, B. An Improved Convolutional Neural Network for Classification of Small Patches of Granite Tiles. **In: 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN).** IEEE, 2018. p. 1125-1130.

SOLAWETZ. **YOLOv4 Explained. 2020**. Disponível em: https://blog.roboflow.com/a-thorough-breakdown-of-yolov4/. Acesso em 13 mar. 2022.

SOLAWETZ J. **YOLOv5 New Version - Improvements And Evaluation**. 2020. Disponível em: https://blog.roboflow.com/yolov5-improvements-and-evaluation/. Acesso em 13 mar. 2022.

SUGIYAMA, S. **Human Behavior and Another Kind in Consciousness: Emerging Research and Opportunities**. IGI Global, 2019.

TENSORFLOW. **TensorFlow Object Detection API repository**. GitHub, 2022. Disponível em: https://github.com/tensorflow/models/tree/master/research/object_detection. Acesso em 22 mar. 2022.

THUAN, D. **Evolution of YOLO algorithm and YOLOv5: The State Object Detection Algorithm**. 2021. Tese (Bacharelado em Tecnologia da Informação) – Oulu University of Applied Sciences, Oulu, 2021.

TORREY, Lisa; SHAVLIK, Jude. Transfer learning. In**: Handbook of research on machine learning applications and trends: algorithms, methods, and techniques**. IGI global, 2010. p. 242-264.

TSANG, S. **Review: SSD — Single Shot Detector (Object Detection)**. 2018. Disponível em: https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11. Acesso em 22 mar. 2022.

ULTRALYTICS. **YOLOv5 GitHub repository**. GitHub, 2022. Disponível em: https://github.com/ultralytics/yolov5. Acesso em 20 jul. 2022.

WANG, C. et al. CSPNet: A new backbone that can enhance learning capability of CNN. *In:* **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops**. 2020. p. 390-391.

WANG, C.; BOCHKOVSKIY, A.; LIAO, H. M. Scaled-yolov4: Scaling cross stage partial network. *In:* **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition**. 2021. p. 13029-13038.

WANG, K. et al. Panet: Few-shot image semantic segmentation with prototype alignment. *In*: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. 2019. p. 9197-9206.

WU, B. et al. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In: **Proceedings of the IEEE conference on computer vision and pattern recognition workshops**. 2017. p. 129-137.

WU, C. et al. A compact dnn: approaching googlenet-level accuracy of classification and domain adaptation. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2017. p. 5668-5677.

WU, H. et al. Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation. **arXiv preprint arXiv:1903.11816**, 2019.

YAMASHITA, R. et al. Convolutional neural networks: an overview and application in radiology. **Insights into Imaging**, v. 9, n. 4, p. 611–629, 22 jun. 2018.

YOSINSKI, J. et al. How transferable are features in deep neural networks?. **Advances in neural information processing systems**, v. 27, 2014.

YU et al. **Tensorflow Model Garden**. 2020. Disponível em: https://github.com/tensorflow/models. Acesso em 10 jun. 2022.

ZHANG, A. et al. Dive into deep learning. **arXiv preprint arXiv:2106.11342**, 2021.

ZHANG, A. **Critérios de Avaliação de Detecção de Objetos-AP mAP**. 2018. Disponível em: https://arleyzhang.github.io/articles/c521a01c/. Acesso em 20 ago. 2022.

ZHAO, L.; LI, S. Object Detection Algorithm Based on Improved YOLOv3. **Electronics**, v. 9, n. 3, p. 537, 24 mar. 2020.

ZHAO, Z.-Q. et al. Object Detection With Deep Learning: A Review. **IEEE Transactions on Neural Networks and Learning Systems**, v. 30, n. 11, p. 3212–3232, nov. 2019.

ZHU, Rui et al. Text detection based on convolutional neural networks with spatial pyramid pooling. In: **2016 IEEE International Conference on Image Processing (ICIP)**. IEEE, 2016. p. 1032-1036.