UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

CIÊNCIA DA COMPUTAÇÃO

RENAN RODRIGUES VALE COSTA

COMPARAÇÃO DE DESEMPENHO ENTRE ALGORITMOS DE RECONHECIMENTO DE OBJETOS EM IMAGEM

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2019

RENAN RODRIGUES VALE COSTA

COMPARAÇÃO DE DESEMPENHO ENTRE ALGORITMOS DE RECONHECIMENTO DE OBJETOS EM IMAGEM

Trabalho de Conclusão de Curso apresentada como requisito parcial à obtenção do título de Bacharel em Ciência da Computação, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dra. Mauren Louise Sguario Coelho de Andrade.

PONTA GROSSA

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharelado em Ciência da Computação

TERMO DE APROVAÇÃO

COMPARAÇÃO DE DESEMPENHO ENTRE ALGORITMOS DE RECONHECIMENTO DE OBJETOS EM IMAGEM

por

RENAN RODRIGUES VALE COSTA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 20 de Novembro de 2019 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof(a). Dra. Mauren Louise Sguario
Orientador(a)

_____
Prof(a). MSc. Geraldo Ranthum
Membro titular

_____
Prof. Dr. Gleifer Vaz Alves
Membro titular

_____
Prof(a). MSc. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof(a). Dra. Mauren Louise Sguario Coelho de Andrade
Coordenador do curso

AGRADECIMENTOS

Certamente estes parágrafos não irão atender a todas as pessoas que fizeram parte dessa importante fase de minha vida. Portanto, desde já peço desculpas àquelas que não estão presentes entre essas palavras, mas elas podem estar certas que fazem parte do meu pensamento e de minha gratidão.

Agradeço a minha orientadora Prof. Dra. Mauren Sguario, que me acolheu como orientando de última hora. E mesmo com a falta de tempo de ambos os lados, me guiou até o presente momento.

Agradeço também a Prof. Dra. Hellyane, pela compreensão, apoio e auxilio oferecido nos primeiros momentos do trabalho, justamente pelo problema que tive com o orientador anterior.

Agradeço, ao Prof. Dr André Kosciaski, que sem os seus ensinamentos durante o período que fiz Iniciação Cientifica com ele, acredito que não teria chegado onde cheguei.

Agradeço ao William, pois sempre que precisei de ajuda, independente do horário do dia, e de quão cheio de trabalhos ele deveria fazer, sempre me ajudou.

Agradeço a Camila, por me ajudar em vários pontos escritos do trabalho e não deixar que eu desistisse logo quando estava tão próximo de terminar o TCC e a Universidade.

Por último, mas não menos importantes, agradeço ao Rafael Valença e ao JEK da Bsoft, por terem me dispensado dos afazeres do trabalho da empresa, para que eu pudesse concluir esse trabalho em um período curto de tempo, e assim realizar a entrega do mesmo.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

RESUMO

Costa, Renan Rodrigues Vale. **Comparação de desempenho entre algoritmos de reconhecimento de objetos em imagem.** 2019. 62 f. Trabalho de Conclusão de Curso (Bacharelado, em Ciência da Computação) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2019.

O trabalho tem como objetivo comparar métodos a fim de servir como um trabalho de validação, além de realizar um estudo sobre algoritmos de reconhecimento de objeto em imagens, partindo desde a introdução a uma imagem, até a comparação entre o desempenho de dois algoritmos, *You only Look Once* e *Faster* RCNN, em um conjunto de dados da MS COCO 2017 e outro da Pascal VOC 2007. Os resultados obtidos foram que, foi notado uma diferença considerável entre o desempenho entre esses dois métodos de reconhecimento de objetos em cena. Tanto no quesito desempenho, quanto no quesito velocidade, no qual foi testado em um conjunto pequeno de imagens. O único ponto no qual o *Faster* RCNN se mostrou superior em relação ao YOLOv3, foi em relação a velocidade de treinamento e quantidade de memória utilizada durante o mesmo processo. Isso ocorre devido a esse método utilizar uma rede menor do que a utilizada pelo YOLOv3 nessa terceira versão.

Palavras-chave: Aprendizagem cognitiva. Inteligência artificial. Sistemas de reconhecimento de padrões.

ABSTRACT

Costa, Renan Rodrigues Vale. **Performance comparison between image object recognition algorithms**. 2019. 62 p. Work of Conclusion Course (Graduation in Ciancia da Computação) - Federal Technology University - Paraná. Ponta Grossa, 2019.

The objective of this work is to compare methods in order to serve as a validation work, additionally, to study object recognition algorithms in images, starting from the introduction to an image, to the comparison between the performance of two algorithms, You only Look Once and Faster RCNN, in a dataset from MS COCO 2017 and another from Pascal VOC 2007. The results obtained were that a considerable difference between the performance of these two scene object recognition methods was noted. Both performance and speed, which was tested on a small set of images. The only point at which Faster RCNN was superior to YOLOv3 was the training speed and amount of memory used during the same process. This is because this method uses a smaller network than the one used by YOLOv3 in this third version.

Keywords: Cognitive leaning. Artificial inteligence. Pattern system recognition.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

mAP	*mean Average Precision*
R-CNN	*Region Convolucional Neural Network*
ZF ou ZF Net	Zeiler & Fergus Net

SUMÁRIO

1 INTRODUÇÃO

Os diversos problemas enfrentados pela sociedade, tais como, crimes, queimadas, trânsito e saúde possuem uma solução que pode envolver redes neurais convolucionais (RNC), ou do inglês *Convolucional neural network* (CNN), uma técnica que aplica um processo de destacar um objeto de uma imagem (processo conhecido como segmentação) e em seguida aplicar o método da convolução e utilizar técnicas de aprendizagem de máquina com o objetivo de reconhecer um ou mais objetos de interesse em uma imagem.

O processo de reconhecer um objeto específico em uma imagem pode ser utilizado para identificação de corpos estranhos em um local, por exemplo, uma câmera dentro de uma pessoa reconhecendo células cancerígenas. Também pode ser usada em sistemas de segurança. Como exemplo, quando uma pessoa acesse uma área não permitida, por exemplo. Existem várias outras aplicações possíveis para reconhecimento de objetos em imagens, suas variações são limitadas pelo conhecimento e imaginação humana.

A técnica utiliza a aplicação de redes neurais, que é uma técnica que faz com um algoritmo funcione de uma maneira a imitar o cérebro humano (neurônios), na qual cada ciclo (interação) da rede neural, faz com que ela altere seus valores, com o objetivo de fazer com que o algoritmo aprenda a reconhecer objetos em uma imagem (com base na resposta dada previamente ao algoritmo) por meio de processos de segmentação e classificação, via treinamento com uma grande escala de dados.

Existem diversos métodos para reconhecimento de objetos em imagens, visando solucionar o problema de realizar a detecção em relação à precisão e velocidade. Este trabalho tem como objetivo a comparar alguns desses métodos utilizados com maior frequência, utilizando como cenário os conjuntos de dados MS COCO (LIN, 2014) e Pascal VOC (EVERINGHAM, 2010).

1.1 OBJETIVOS

O objetivo deste trabalho é realizar um estudo comparativo entre os métodos de rede neural *convolucional* YOLOv3 e *Faster* CRNN a fim de determinar sua eficiência no reconhecimento de objetos em imagens digitais em conjuntos de dados conhecidos. Esses métodos serão avaliados comparando a precisão no reconhecimento dos objetos nas imagens e nos conjuntos de dados, e o tempo levado para a realização do processo de reconhecimento desses objetos nas imagens.

1.2 JUSTIFICATIVA

Pesquisa sobre reconhecimento de objetos em imagens estão em um panorama crescente, principalmente no ramo de Inteligência Artificial, que abrange veículos autônomos, robótica e aprendizagem de máquina em geral, porém a validação desses dados não é uma constatação frequente na literatura.

Acredita-se que os resultados desse trabalho poderão contribuir com futuras validações entre algoritmos de reconhecimento de objetos em uma cena, além de poder ser utilizado como material para apoio ao estudo sobre o assunto, visto que, o trabalho explica desde o conceito de filtros, segmentação de objetos, métodos convolucionais até uma comparação entre algoritmos que utilizam esses conceitos para detecção de objetos em imagens.

Apresentar um trabalho que realiza a validação de dados, comparação entre algoritmos e contribui para pesquisas na área, é de extrema importância para a compreensão dos elementos que hoje beneficiam ou desfavorecem o cotidiano de muitas pessoas.

1.3 ESTRUTURA DO TRABALHO

O trabalho está organizado em capítulos, sendo o Capítulo 1 a introdução do trabalho, contendo seus objetivos e justificativa. O Capítulo 2 contém a fundamentação teórica, contendo os conceitos básicos sobre rede neural, rede

neural *convolucional* e suas aplicações em imagens. No Capítulo 3, é apresentada a fundamentação teórica do trabalho, contendo menções a trabalhos já realizados na área, explicação do funcionamento do algoritmo R-CNN e suas variações até *Faster* R-CNN, além do algoritmo *You Only Look Once* e sua variação para a versão 3. No Capítulo 4 será demonstrada como foi realizada a etapa de treinamento de cada algoritmo. No Capítulo 5, é mencionado os parâmetros de realização da etapa de teste e no Capítulo 6 os resultados obtidos juntamente com a conclusão do trabalho e menções a trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados todos os conceitos pesquisados e aplicados para a elaboração deste trabalho. Nele é abordado o assunto sobre redes neurais, mais especificamente a rede neural *perceptron*. É abordado também o conteúdo sobre convolução de uma imagem, com exemplos e como é aplicada, tanto por um vetor, quanto por uma matriz, que caracteriza uma imagem.

Após os conceitos básicos serão apresentados os conceitos sobre redes neurais convolucionais e sua variação (rede neural convolucional por região), buscando como apoio trabalhos reconhecidos já realizados anteriormente por outros pesquisadores.

Por fim, para esclarecer ao leitor, é apresentada uma comparação entre a funcionalidade dos dois algoritmos.

2.1 CONCEITOS BÁSICOS

Uma imagem é a representação, reprodução ou imitação da forma de uma pessoa, de um objeto ou de uma cena em particular seja essa cena real ou abstrata. Para reproduzir esse aspecto, o computador, representa uma imagem como uma matriz contendo vários *pixels* de várias cores diferentes espalhados por toda a imagem.

Um *pixel* é a menor representação possível de uma imagem no ambiente computacional. Ao comparar com a realidade, o pixel seria o átomo. Dentro de cada pixel normalmente contêm 3 valores, cada um desses valores é referente a uma escala de cor, sendo elas vermelho, verde e azul (ou RGB como é mais conhecida). A mistura entre os valores dessas 3 intensidades de cores, geram uma gama de outras cores que são utilizadas em toda a imagem.

Na Figura 1 está da esquerda para a direita uma “aproximação” feita em uma imagem até que seja possível ver os valores dos *pixels*. Na esquerda está a imagem original, no centro está uma aproximação dessa imagem, sendo exibido os pixels dela e na direita é apresentado de maneira fictícia como os pixels da imagem do centro são vistos pelo computador.

Figura 1 - Demonstração da aproximação de uma imagem até a exibição de seus pixels

Fonte: Autoria própria.

O conceito de *pixels* em imagens facilita no processo de entendimento dos métodos de pré-processamento que serão mencionados em algumas das subseções seguintes.

Pré-processamento é uma etapa do processamento de imagem que se define em preparar uma imagem para ser processada por algum outro procedimento.

2.1.1 Convolução

Convolução é uma técnica muito importante quando o objetivo é realizar uma análise de uma imagem (SONKA, 2014). Com ela é possível realizar operações como realçar elementos da imagem, ou seja, detectar bordas, contornos, linhas horizontais, linhas verticais. Também é possível restaurar ou eliminar pontos da imagem, melhorar as imagens ou suavizando diferenças, deformar a imagem, entre outras operações (OSÓRIO, 2000).

A filtragem de uma imagem, utilizando uma técnica de convolução, geralmente ocorre na etapa de pré-processamento. O objetivo da filtragem é minimizar a quantidade de ruído (geralmente são definidos por chuviscos nas imagens ou desfoque devido à má qualidade desta imagem) que tal imagem pode possuir, mas também pode ser utilizado para o processo de detecção de bordas, realce, entre outros (COSER, 2009).

Para a realização desse pré-processamento, é necessário fazer convoluções em uma imagem utilizando uma matriz máscara. Uma matriz máscara é uma matriz normalmente pequena (2x2, 3x3,4x4 ou 5x5 por exemplo) cujo seus valores terão impacto direto em uma outra matriz maior (imagem). Dependendo dos valores inseridos para a matriz máscara, o resultado obtido pode ser completamente diferente. O cálculo de aplicação de uma convolução tendo como máscara uma matriz 3x3, é dado pela seguinte formula:

$$d\ (x, y) = m1 * p1 + m2 * p2 + \ldots + m9 * p9 = \sum mi * pi \text{ sendo } i = 1 \text{ até } 9$$

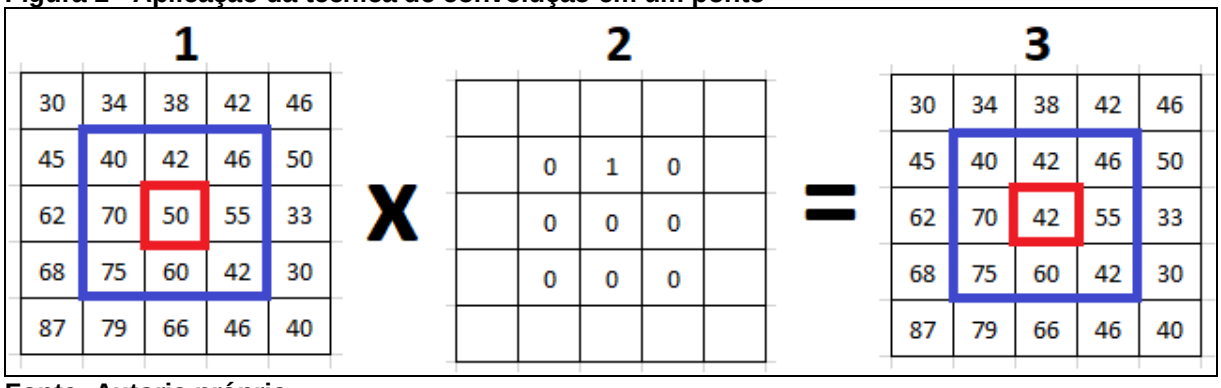
Sendo que, d (x, y) é o valor central da imagem, em relação a posição da matriz mascara, “m” representam valores da matriz 3x3, “p” é referente ao valor na imagem onde tal posição está sobreposta na matriz e “i” é a posição do elemento na matriz.

Ambos os valores “p” e “m” são vistos em forma matricial. A matriz com os valores de “p” é apenas um pedaço de uma imagem (o pedaço onde a máscara atualmente se encontra), e “m” são os valores da matriz máscara.

A matriz máscara percorre todos os pixels da imagem, realizando o cálculo já demonstrado. Isso justifica o alto custo computacional para aplicar o método de convolução em uma imagem.

A Figura 2 a seguir, exemplifica a aplicação da fórmula apresentada. Nela a matriz 1 refere-se a imagem original, a matriz 2 é a máscara a ser aplicada e a matriz 3 é o novo valor resultante para aquele campo, após aplicar a convolução.

Figura 2 - Aplicação da técnica de convolução em um ponto



Fonte: Autoria própria

O quadrado 3x3 na matriz 1, indica onde a matriz máscara 2 será aplicada. O quadrado central indica qual campo será modificado após a aplicação da matriz 2.

2.1.2 Aplicação em Imagem

A aplicação da técnica de convolução numa imagem é um processo que requer um gasto em processamento elevado, esse gasto é relativo ao tamanho da máscara de convolução que será aplicado para aquele cenário. A escolha do tamanho da máscara é variável para cada problema, sendo alguns requerendo uma máscara menor (como por exemplo, uma matriz 6x6) e outros uma máscara maior (exemplo, 20x20). Essa decisão está diretamente relacionada à eficiência do algoritmo para solucionar um problema em um determinado caso.

Vale notar que, quanto maior a matriz escolhida para a resolução de um problema, maior será o valor i da função de somatória, e com isso maior será o custo para a realização dessa convolução. Por esse motivo pesquisadores, como Liang-Chieh et al (CHEN et al, 2018), Shaoqing et al (REN et al, 2015), Ross (GIRSHICK, 2015), Jianan et al (LI, 2018) entre outros, constantemente demonstram em seus trabalhos a preocupação com o custo de processamento de seus algoritmos/métodos.

Como já mencionado anteriormente, realizar a convolução de uma imagem pode ter vários objetivos. Nesta seção serão demonstrados três exemplos da aplicação da convolução em imagens, com a finalidade de: focar, reduzir o brilho e segmentar essa imagem.

O primeiro passo foi selecionar uma imagem e converte-la para uma imagem em níveis de cinza, pois assim o algoritmo terá que trabalhar com a convolução de apenas uma matriz. Caso fosse selecionada uma imagem colorida, seria necessário aplicar algoritmo de convolução em ao menos 3 matrizes diferentes. A Figura 3 a seguir demonstra à esquerda a imagem original no formato RGB (colorida) e à direita a mesma imagem em níveis de cinza.

Figura 3– À esquerda, uma imagem com uma rosa em cores. À direita, a mesma imagem em níveis de cinza

Fonte: Autoria própria

Os próximos passos serão separados em subseções, pois a partir do momento que a imagem está em níveis de cinza (nesse caso) é possível obter resultados diferentes com base na convolução realizada.

Na primeira subseção serão apresentados os passos necessários para fazer com que a imagem em níveis de cinza da Figura 5 fique mais nítida. A segunda subseção será realizar o processo oposto ao da subseção 1, ou seja, fazer com que a imagem fique menos nítida (desfocada). Por fim, na terceira subseção será realizado o processo de realçar as bordas da imagem, aumentando assim a tonalidade nos pontos de diferença de coloração da imagem original em níveis de cinza.

2.1.2.1 Reduzindo o brilho

Para realizar a redução do brilho da imagem da flor em escala de cinza, será aplicado a seguinte matriz M a Figura 3:

$$M(3,3) = \begin{vmatrix} 0 & 0 & 0 \\ 0 & 0{,}2 & 0 \\ 0 & 0 & 0 \end{vmatrix}$$

Aplicando essa matriz na fórmula demonstrada anteriormente:

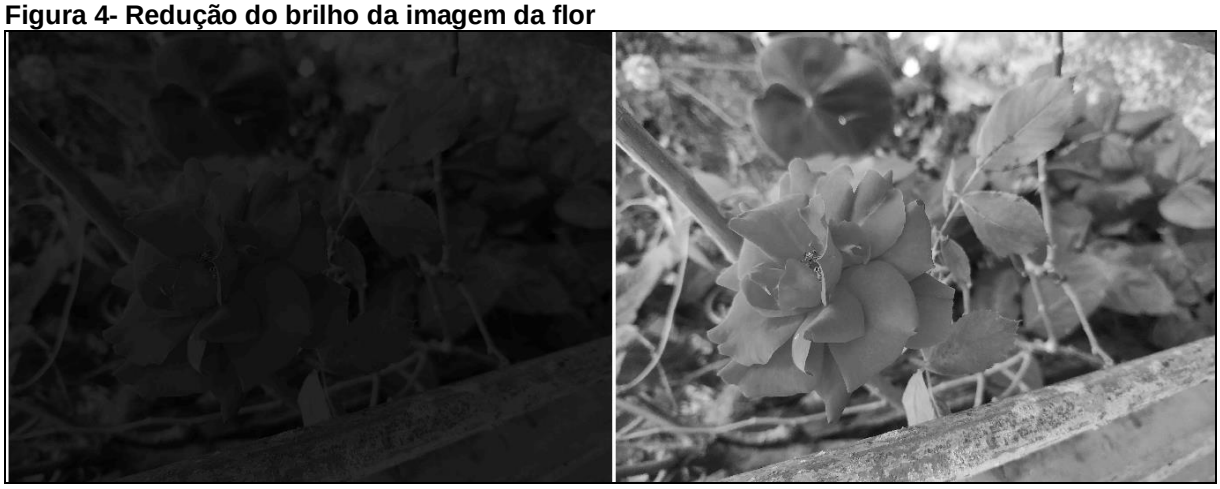
$d\ (x,\ y) = 0 * p1 + 0 * p2 + 0 * p3 + 0 * p4 + 0{,}2 * p5 + 0 * p6 + 0 * p7 + 0 * p8 + 0 * p9$

Então:

$d\ (x,\ y) = 0{,}2 * p5$

Sabendo que "$p5$" é igual ao valor atual de d (x, y), é possível afirmar que essa matriz ignora todos os elementos ao redor do principal e faz a multiplicação desse elemento por 0.2, reduzindo assim seu valor. Então fazendo isso para toda a matriz, todos os valores estarão 80% menores do que era originalmente, resultando em uma imagem com brilho reduzido. Na Figura 4, sabendo que a imagem à direita é a original e a que está à esquerda é a imagem com a aplicação da matriz para redução do brilho, é possível verificar que, a imagem ficou notavelmente mais escura do que ela originalmente era.

Figura 4- Redução do brilho da imagem da flor



Fonte: Autoria própria.

2.1.2.2 Segmentando a imagem

Segmentar uma imagem, é destacar algum objeto de interesse presente nesta, a fim de reconhecer aquele objeto, ou seja, se um algoritmo consegue segmentar corretamente um objeto em uma imagem, significa que esse mesmo algoritmo conseguiu reconhecer aquele objeto na imagem.

Para realizar a segmentação da flor da imagem à direita, na Figura 3, primeiramente é necessário separar as matrizes que geram as cores da imagem da flor, gerando assim 3 matrizes (R, G e B) que individualmente elas geram imagens

em níveis de cinza. Quanto maior o valor do *pixel*, maior será a intensidade daquela cor naquele ponto da imagem.

Após obter a matriz R da imagem a esquerda da Figura 5, o próximo passo pode ser fazer a subtração da matriz G com a matriz R, por ser necessário apenas a flor, não há a necessidade de exibir a vegetação na segmentação. A Figura 5 demonstra a matriz R, G e a matriz V, que é a matriz resultante da subtração.

Figura 5- A esquerda matriz R, ao centro G e a direita R-G

Fonte: Autoria própria.

Partindo da matriz V, aplicando um filtro para redução das cores, deixando a imagem mais “limpa”, utilizando uma função para binarização, ou seja, conversão da imagem em uma imagem com apenas 2 cores (branco ou preto) e por fim utilizando uma outra função que realiza o preenchimento da imagem, é obtida a imagem da Figura 6.

Figura 6- Flor segmentada

Fonte: Autoria própria.

Observando o resultado obtido na Figura 6, é possível verificar que a aplicação desses filtros resultou na segmentação correta da flor em relação a imagem original.

Agora assumindo uma outra imagem (Figura 7, à esquerda) em que existam outros objetos com a coloração avermelhada. Ao aplicar exatamente os mesmos filtros dos aplicados anteriormente para gerar a segmentação da Figura 6, o resultado não é a segmentação da flor, como esperado (Figura 7, à direita).

Figura 7- À esquerda, foto original, à direita a mesma imagem, porém, segmentada

Fonte: Autoria própria.

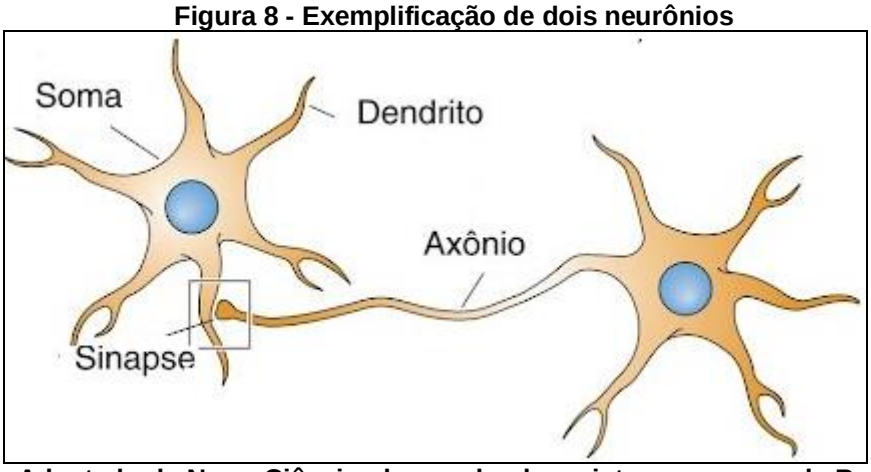
2.1.3 Redes neurais

Redes neurais são modelos computacionais que tem como principal inspiração o cérebro humano (ou rede neural biológica) e a maneira que ele se comporta em relação a aprendizagem das coisas e processamento de informação (AGATONOVIC-KUSTRIN, 2000).

A Figura 8 foi adaptada do livro NeuroCiência: desvendando o sistema nervoso de Bear (2008), e nela está sendo representada uma rede neural biológica, ou seja, a cerebral. Possuí uma unidade principal para tratar todos os pulsos que chegam até ele, chamada neurônio (mais especificamente a soma do neurônio). Toda a informação (pulsos) a serem “processados”, passam por um conjunto de

dendritos, passando-as para o corpo da célula, e por fim repassando para outros neurônios (ou para o sistema nervoso que coordena um dado músculo) através do axônio, com o auxílio das sinapses (YEGNANARAYANA, 2009) (BEAR, 2008).

Figura 8 - Exemplificação de dois neurônios



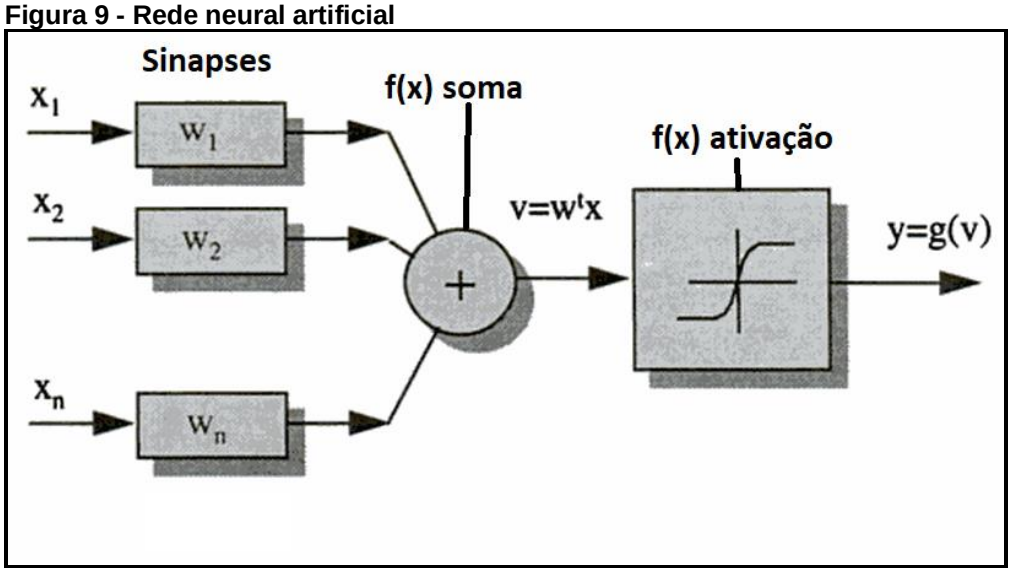
Fonte: Adaptado de NeuroCiência: desvendando o sistema nervoso de Bear (2008).

"O cérebro é um computador (sistema de processamento de informação) altamente complexo, não linear e paralelo" (HAYKIN, 2007, p.27). Partindo desse princípio, foi pensado (em meados do século XX) elaborar um algoritmo que fosse capaz de processar informações de maneira semelhante ao cérebro humano, ou seja, foi criada a rede neural artificial.

"A maneira que uma rede neural funciona é muito semelhante ao funcionamento do neurônio, nos aspectos referentes ao conhecimento que é adquirido pela rede através de um processo de aprendizagem, e na relação de força de conexão entre neurônios, ou seja, pelos pesos entre as sinapses" (HAYKIN, 2007).

Uma rede neural artificial possui sinapses, função de soma, função de ativação e saída. A Figura 9 demonstra como o neurônio é representado no ambiente computacional, deixando de maneira explicita que sua inspiração vem dessa relação.

Figura 9 - Rede neural artificial



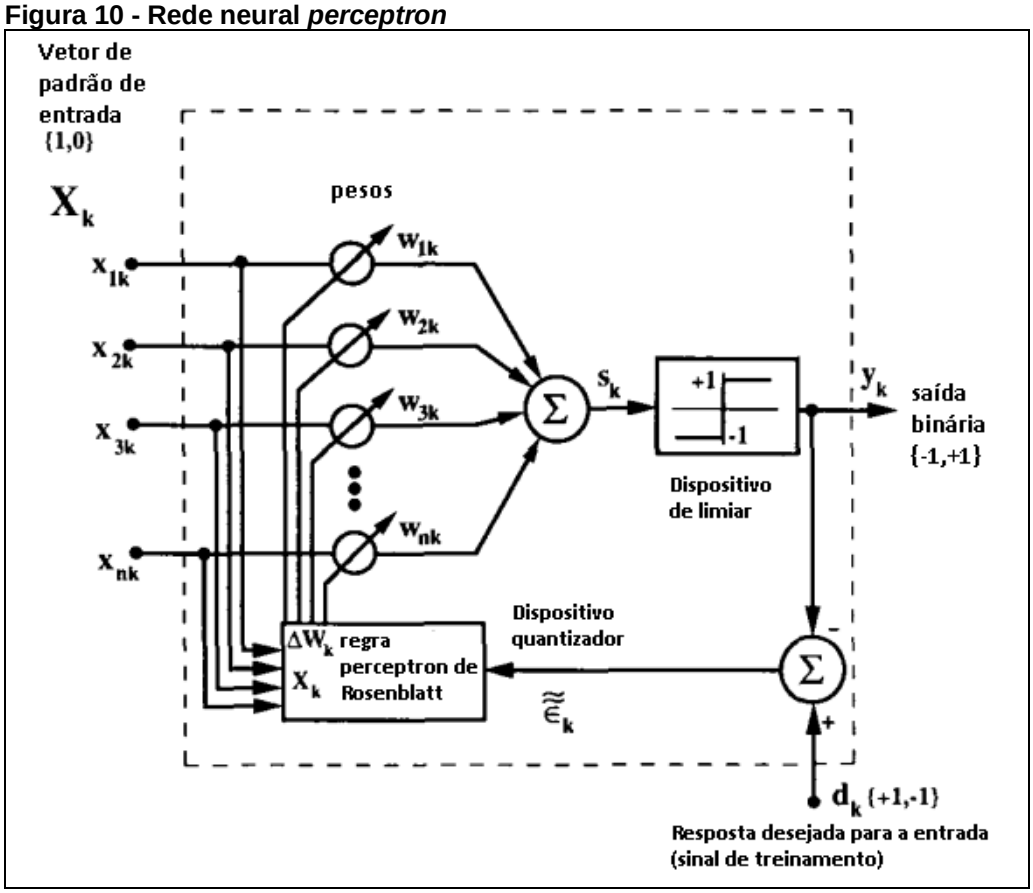
Fonte: Adaptado de Redes neurais artificiais de Kovács (2002).

Como apresentado na Figura 9, “x1, x2 ..., xn” são os valores de entrada do neurônio, em seguida, cada valor possui um peso que é definido pela sinapse especifica de cada entrada. Após calculado os pesos, os valores passam por uma função de soma e uma outra de ativação, para assim alcançar uma saída.

2.1.3.1 Rede neural *Perceptron*

A Rede Neural Artificial *Perceptron*, tem como função principal, fazer com que os pesos das sinapses gerem, com base na entrada fornecida, a saída desejada. A Figura 10 é possível observar como a rede neural *perceptron* opera. Primeiro os valores são inseridos cada referente a um peso específico, em seguida é aplicado uma função que sua saída vai ser aplicada a uma função de ativação, se o valor não for o suficiente para ativar a função de ativação, o algoritmo recalcula os pesos das sinapses de entrada em seguida é executado o processo novamente. Caso o valor seja o suficiente para a função de ativação, o valor é exibido na saída da rede neural. Caso a saída esteja correta, a convergência da *Perceptron* não irá atualizar o valor dos pesos das sinapses (WIDROW, 1990).

Figura 10 - Rede neural *perceptron*



Fonte: Adaptado de *30 years of adaptive neural networks: perceptron, madaline, and backpropagation* de Widrow (1990).

Sempre que uma camada for executada e resultar em erro o método deve atualizar seu peso e continuar a execução do algoritmo com o novo peso a partir daquela camada. A execução deve persistir até que a convergência seja precisa, com base nas respostas informadas nos casos de erro, ou caso o algoritmo alcance o limite de iterações definido previamente (GARDNER, 1998).

2.2 APRENDIZAGEM DE MÁQUINA

De acordo com o dicionário da língua portuguesa, aprendizagem é o processo de aquisição de conhecimentos, habilidades, valores e atitudes, possibilitado através do estudo, do ensino ou da experiência. O processo fundamental na aprendizagem é a imitação, ou seja, repetição de um processo observado.

Neste sentido, aprendizagem de máquina é a capacidade de fazer com que uma máquina aprenda através de tentativa e erro a realizar alguma ação com eficiência. O aprender de uma máquina pode ser definido como o reconhecimento de padrões em algum cenário específico, seja ele, uma sequência de imagens, uma sequência de números probabilísticos, alterações no clima de uma determinada região, etc.

Técnicas de aprendizagem de máquina são necessárias para melhorar a precisão dos modelos preditivos. Dependendo da natureza do problema em questão, existem diferentes abordagens com base no tipo e no volume dos dados. Sejam elas aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço.

Um outro método é a aprendizagem profunda. É um método específico de aprendizado de máquina que incorpora redes neurais em camadas sucessivas para aprender com os dados de uma maneira iterativa. Aprendizagem profunda é especialmente útil quando se está tentando aprender padrões de dados não estruturados.

2.3 APRENDIZAGEM PROFUNDA

Embora a aprendizagem de máquina tenha se tornado parte integrante do processamento de dados, uma das principais diferenças em relação à aprendizagem profunda é que ela exige uma intervenção manual na seleção dos recursos a serem processados. Extração de recursos na aprendizagem de máquina requer o trabalho do cientista de dados em pré-processar os dados e entregar aos algoritmos, dados que possam ser explorados em busca de padrões.

A aprendizagem profunda ou *Deep Learning* como também é conhecida, é um paradigma para a realização de aprendizagem de máquina. Aprendizagem profunda foi desenvolvida inspirando-se na capacidade do cérebro de aprender. Assim como o cérebro humano pode identificar um objeto em milissegundos, a aprendizagem profunda pode refletir esse instinto com quase a mesma velocidade e precisão. Por exemplo, enquanto muitos módulos de visão computacional tradicionais podem facilmente reconhecer qualquer objeto dado, um segundo objeto

em uma imagem, com uma ligeira obstrução, provavelmente não seria reconhecido, ou pelo menos não de forma adequada.

Aprendizagem profunda é capaz de resistir a pequenas mudanças e com isso, consegue generalizar a partir de dados parciais, tornando mais fácil para identificar corretamente um objeto parcialmente obstruído. Aprendizagem profunda tem a capacidade de avaliar um objeto, processar adequadamente a informação e adaptar-se a diferentes variantes. A tecnologia pode aprender recursos de alto nível, não-lineares necessários para uma classificação precisa. Os dados brutos são alimentados através de redes neurais profundas, que aprendem a identificar o objeto no qual são treinados.

2.3.1 CNN

Rede Neurais Convolucionais, ou *Convolucional Neural Network* (CNN) como é conhecida, é a implementação de uma Rede Neural *Perceptron* (ou por camadas) aos métodos já conhecidos de convolução.

Ela é aplicada da seguinte maneira: cada valor da matriz de convolução (matriz máscara) representa o peso de uma sinapse. Então cada iteração errada da rede neural ao realizar uma convolução, é modificada e adaptada para tentar gerar o resultado esperado. Em função da CNN possuir poucos parâmetros e conexões, ela se torna mais fácil de ser treinada, pois basta após um resultado inesperado, o desenvolvedor exibir qual era a resposta esperada para tal caso (KRIZHEVSKY, 2012).

Redes Neurais convolucionais (CNN), utilizam o conceito de *backpropagation,* que é um algoritmo que se baseia em duas etapas, a *Foward Pass* e o *backward pass*, e tem como objetivo otimizar os pesos para que a rede neural possa aprender a mapear corretamente as entradas para as saídas. (LECUN, 1989) (GOODFELLOW, 2016).

O passo para frente, ou *forward pass,* é onde as entradas são passadas através da rede e as previsões de saída obtidas.

O passo para trás, ou *backward pass*, é onde o gradiente da função é calculado para poder ter conhecimento da perda na camada final, ou seja, camada

de previsão da rede. Esse gradiente é utilizado para atualizar os pesos das sinapses da rede neural (GOODFELLOW, 2016).

Um ponto negativo da utilização de redes convolucionais neurais, é o gasto de processamento elevado. Pois para realizar a convolução de uma matriz 3x3 em uma imagem 3000x3000 é realizado em torno de 81 milhões de cálculos matemáticos. Após realizar esses cálculos, o algoritmo realiza a verificação se os novos valores atendem ao requisito esperado. Caso não atendam é realizado mais 81 milhões de cálculos até que seja retornado o valor esperado.

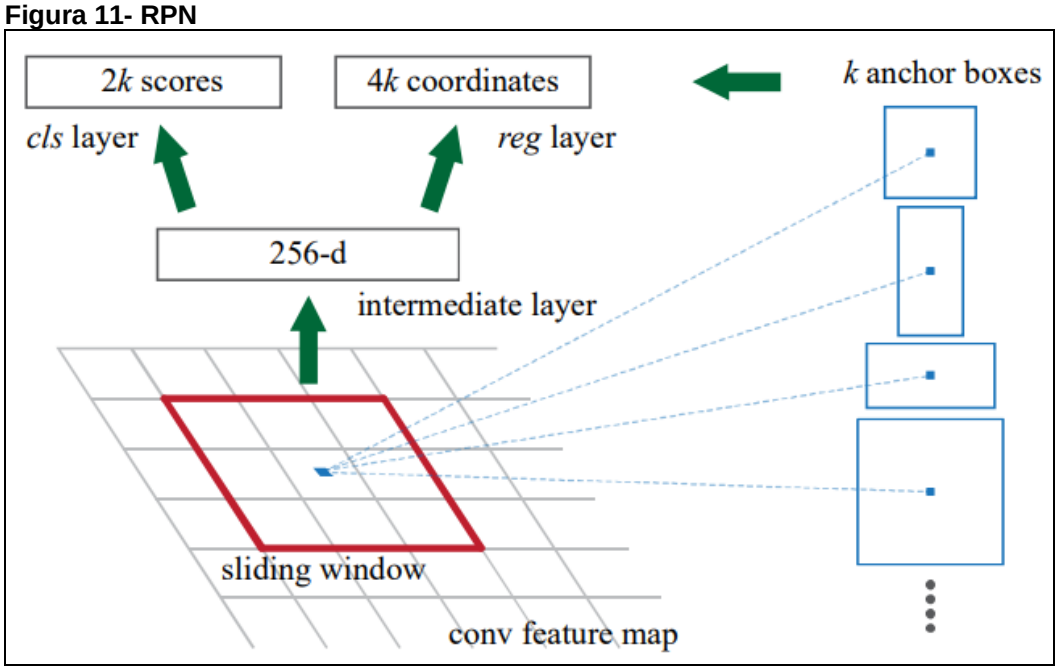
2.3.2 *Faster* R-CNN

Faster R-CNN foi implementado por Ren et al. (2015) e é uma junção do algoritmo de Fast R-CNN e do algoritmo demonstrado no trabalho, chamado RPN (*Region Proposal Network*).

Ren observou que o algoritmo de Fast R-CNN é muito eficiente, porém sua execução e seus testes foram realizados assumindo que a imagem inicial recebida pelo algoritmo é uma imagem com as suas regiões contendo possíveis imagens já selecionada.

O algoritmo RPN, partindo de uma entrada, sendo ela o último mapa convolucional gerado aplicando o modelo VGG de Simonyan e Zisserman (2014) e o modelo ZF de Zeiler e Fergus (2014), ele projeta uma pequena *network* sobre o esse último mapa gerado, aplica k modelos de caixas ancora, sendo elas N x N, 2N x N, N x 2N, N/2 x N/2X, etc. Cada k caixa ancora é utilizada com a aplicação de rede neural onde será obtido 4 saídas referentes a coordenada dessa caixa ancora e 2 saídas referentes a pontuação que estima a probabilidade de existir um objeto proposto naquela caixa ancora. A Figura 11 demonstra graficamente uma RPN.

Figura 11- RPN



Fonte: Adaptado de Ren et al. (2015)

A etapa final é realizar a escolha da caixa ancora com melhor pontuação para aquela região para que ela possa representar aquele possível objeto. Após escolher a caixa ancora com maior pontuação é executado o algoritmo já demonstrado de *Fast* R-CNN para que seja realizado o reconhecimento daquele objeto.

A combinação dos 2 algoritmos é realizada em 4 etapas:

- Na primeira etapa, é realizado o treinamento da RPN conforme já explicado.
- Na segunda etapa é realizado o treinamento da *Fast* R-CNN utilizando as possíveis regiões geradas pela etapa 1 (com RPN).
- Na terceira etapa é realizado o treinamento novamente utilizando a RPN, mas dessa vez fazendo reajustes na camada exclusiva para o RPN com base no registro gerado pela etapa 2, assim as duas redes passam a compartilhar suas camadas.
- A quarta etapa mantendo as camadas convolucionais compartilhadas fixas, são ajustadas as camadas exclusivas da *Fast* R-CNN. Com isso, ambas as redes compartilham as mesmas camadas convolucionais e formam uma rede unificada. (REN et al., 2015)

Por essa união e devido à isso, o algoritmo gerar resultados melhores (vide Tabela 1) em relação a *Selective Search* (SS) (UIJLINGS et al., 2013). Essa combinação dos 2 algoritmos foi chamada de *Faster* R-CNN.

Tabela 1 - Comparação da aplicação de RPN com SS nos modelos VGG e ZF

modelo	sistema	conv	proposta	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0,5 fps
VGG	RPN + Fast R-CNN	141	10	47	198	5 fps
ZF	RPN + Fast R-CNN	31	3	25	59	17 fps

Fonte: Adaptado de Ren et al. (2015)

2.3.3 *You Only Look Once* v3

You Only Look Once Version 3, ou como também é conhecido, YOLOv3. É a terceira versão aprimorada por Redmon e Farhadi (2018) do algoritmo YOLO, também desenvolvido por Redmon como já descrito em capítulos anteriores deste trabalho.

A melhoria conforme mencionado pelos autores foi o desenvolvimento de uma nova rede para realizar a extração de recursos. A nova rede é uma mistura entre projetos anteriores e esse novo e residual material de rede. Ela usa camadas convolucionais sucessivas, e também possui algumas conexões de atalho o que a deixa significativamente maior (comparando com a versão anterior, pois possuía somente 19 camadas convolucionais). Possui 53 camadas convolucionais e é chamada de Darknet53. (REDMON e FARHADI, 2018)

A Tabela 2 demonstra como Redmon e Farhadi projetaram e desenvolveram as camadas de convoluções para o Darknet53. Com essa nova estrutura eles visaram solucionar a falha do algoritmo YOLO (REDMON, 2015) por ter problemas para reconhecer objetos muito pequenos em uma imagem, que existia em versões anteriores que fazia com que o mesmo não apresentasse um bom resultado em precisão média (*mean Average Precision* – (mAP)), pois este realiza uma verificação em várias escalas diferentes de camadas na cena, tendo assim uma melhor precisão para encontrar tanto objetos grandes, quanto objetos pequenos.

Tabela 2 - Camadas convolucionais do Darknet53

	Tipo	Filtros	Tamanho	Saída
	Convolucional	32	3 x 3	256 x 256
	Convolucional	64	3 x 3 / 2	128 x 128
1x	Convolucional	32	1 x 1	
	Convolucional	64	3 x 3	
	Residual			128 x 128
	Convolucional	128	3 x 3 / 2	64 x 64
2x	Convolucional	64	1 x 1	
	Convolucional	128	3 x 3	
	Residual			64 x 64
	Convolucional	256	3 x 3 / 2	32 x 32
8x	Convolucional	128	1 x 1	
	Convolucional	256	3 x 3	
	Residual			32 x 32
	Convolucional	512	3 x 3 / 2	16 x 16
8x	Convolucional	256	1 x 1	
	Convolucional	512	3 x 3	
	Residual			16 x 16
	Convolucional	1024	3 x 3 / 2	8 x 8
4x	Convolucional	512	1 x 1	
	Convolucional	1024	3 x 3	
	Residual			8 x 8
	Seleção Média		Global	
	Connected		1000	
	Softmax			

Fonte: Adaptado de Yolov3: *An incremental improvement* de Redmon e Farhadi (2018)

2.4 RESUMO DO CAPÍTULO

No capítulo foi feita uma breve explicação sobre a definição de uma imagem, métodos convolutivos e sobre a aplicação de segmentação de imagens. Foi explicado também a definição de redes neurais, que são modelos computacionais que tem como principal inspiração o cérebro humano e a maneira que ele se

comporta em relação a aprendizagem e processamento de informações. Foi mencionado sobre a diferença da rede neural para a rede neural *Perceptron*.

Ao final do capítulo foi explicado parte do processo feito pelos algoritmos *Faster* R-CNN e YOLOv3 para efetuar o reconhecimento de objetos em imagens. Sendo eles o processo de aplicação de camadas convolucionais, a fim de reduzir o tamanho da imagem, buscando reconhecer objetos de diferentes dimensões.

3 TRABALHOS RELACIONADOS

Akcay et al. (2018) utilizou o conceito da rede neural convolucional para reconhecer objetos em malas utilizando as imagens de raio X, contribuindo para a identificação de pessoas que praticam tráfico ilegal de algum objeto ou substância.

Zhang et al. (2018) aplicou o conceito da CNN a fim de conseguir detectar através de imagens, fumaça em florestas, prevenindo assim que eventos de queimadas, sejam elas legalmente autorizados ou não, causem muitos danos a natureza.

Bojarski et al. (2016) utilizou a rede neural convolucional com sucesso ao utilizar um sistema que aprende a dirigir com dados mínimos de treinamento de seres humanos em estradas. O sistema consegue operar também, segundo os autores, em áreas com orientação visual pouco clara, como estacionamentos e estradas não pavimentadas.

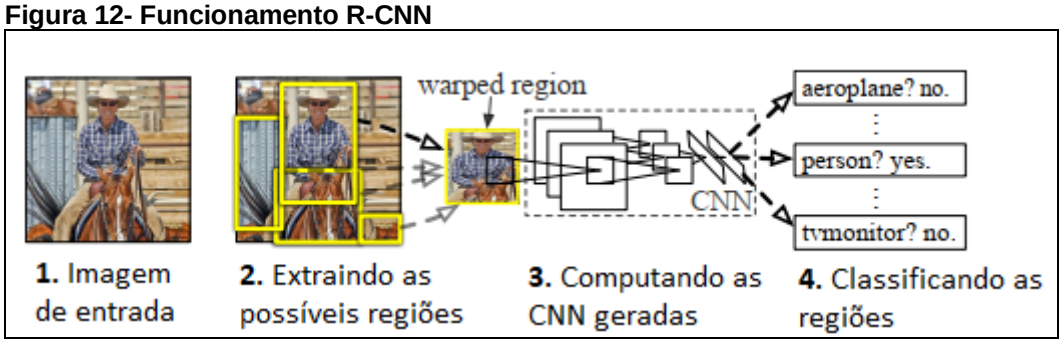
Zhu et al. (2017) propôs empregar uma rede neural convolucional, utilizando a informação convolucional para segmentar com precisão a próstata a partir de imagens de ressonância magnética. Os resultados obtidos nessa pesquisa, com a aplicação da CNN, foram, como demonstrado pelos autores, superiores em relação a outros métodos relatados no trabalho, como U-Net e *Fully Convolucional Networks* (FCN).

3.1 R-CNN

Ross Girshick et al. (2014) propuseram um método em que ao invés de realizar a convolução sempre em todos os pontos da imagem, a convolução será realizada em regiões diferentes, na qual acredita-se haver um objeto.

Na Figura 12 é apresentado como o algoritmo implementado por Girshick et al. funciona. Com base em uma imagem de entrada (1) o sistema extrai cerca de 2000 possíveis regiões que possuem um objeto a ser reconhecido, em seguida (3) é realizado a convolução utilizando redes neurais para cada uma das regiões. Por fim (4) o sistema classifica a cada região da imagem utilizando um classificador linear SVM para assim gerar um resultado de qual objeto foi segmentado (ex: pessoa, cachorro, parede, ave, etc).

Figura 12- Funcionamento R-CNN



Fonte: Girshick et al. (2014)

Como descrito pelo autor, o algoritmo foi dividido em 3 etapas, sendo a primeira, gerar propostas de região independentes de categoria. A segunda etapa consiste em aplicar o conceito de rede neural convolucional que extrai um vetor de tamanho fixo de cada região. A terceira etapa aplica os recursos obtidos em um método de classificação linear, para assim obter os resultados. Essas etapas são melhor exploradas nas próximas seções desse documento.

3.1.1 Regiões

Uijlings et al. (2013) em seu trabalho demonstrou a aplicação do método de segmentação a fim de reconhecer objetos em uma cena. O algoritmo apresentado por ele funciona com diferentes perspectivas de reconhecimento:

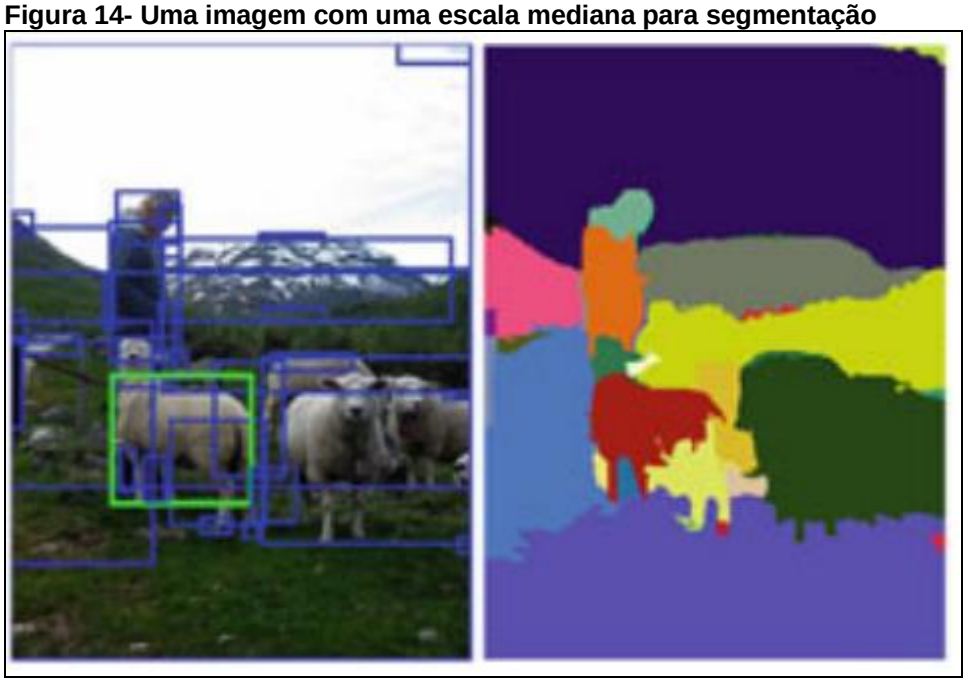
Escala de segmentação pequena: Utilizando uma segmentação com critérios mais rigorosos para padrões de cores nas imagens, o algoritmo segmenta os mínimos detalhes de objetos em cena. Um exemplo dessa segmentação pode ser observado na Figura 13. À direita é apresentada a imagem segmentada. Cada variação entre as cores segmentadas, é referenciada como um objeto diferente encontrado.

Figura 13- Uma imagem com uma escala pequena para segmentação

Fonte: Adaptado de Uijlings et al. (2013).

Escala de segmentação mediana: Utilizando esse modelo o algoritmo consegue uma tolerância maior para as mudanças de coloração de um objeto. Com isso acaba reconhecendo um objeto maior, ou um objeto que o componente dele foi reconhecido pela segmentação mínima. Um exemplo dessa segmentação pode ser observado na Figura 14. Como mencionado, é possível ver que na aplicação desse modelo de tolerância cada ovelha no rebanho foi reconhecida juntamente com suas patas, que haviam sido observadas individualmente com a mínima tolerância.

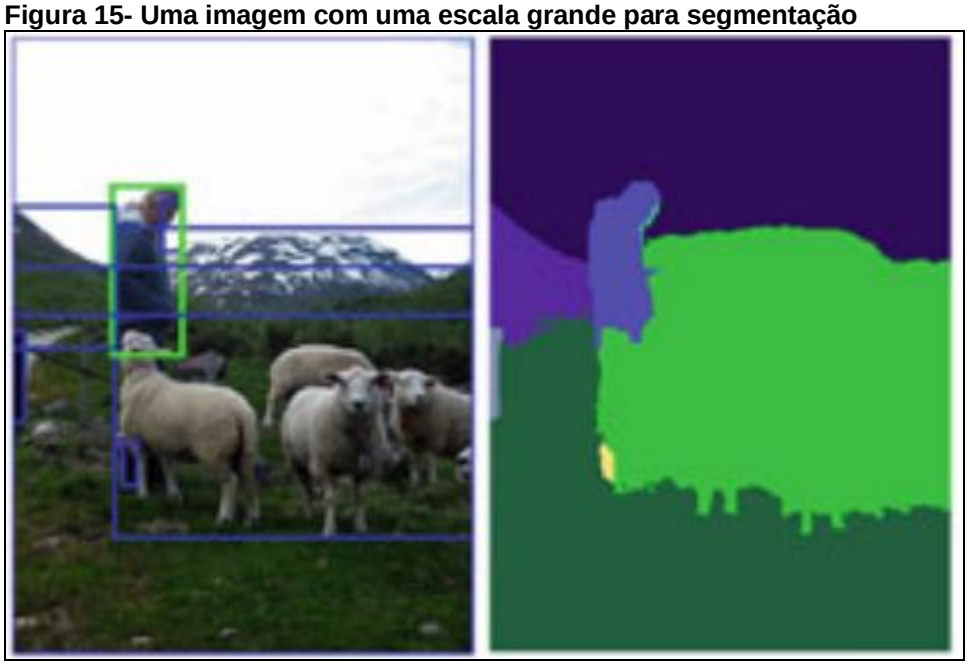
Figura 14- Uma imagem com uma escala mediana para segmentação



Fonte: Adaptado de Uijlings et al. (2013).

Escala de segmentação maior: É utilizada para reconhecer objetos grandes e com uma grande tolerância de variação de coloração. Um exemplo dessa segmentação pode ser observado na Figura 15 a esquerda.

Figura 15- Uma imagem com uma escala grande para segmentação



Fonte: Adaptado de Uijlings et al. (2013).

A variância entre estas perspectivas é definida com base em um valor para escala de um possível objeto na cena, ou seja, o algoritmo não é necessariamente dividido em três perspectivas diferentes, sendo cada uma delas para uma categoria de escala.

Girshick utilizou esse algoritmo como uma das bases para a implementação do *Region Convolucional Neural Network*. Variando entre os níveis de tolerância o algoritmo R-CNN divide a cena inicial em 2000 regiões onde provavelmente podem conter um objeto a ser reconhecido.

3.1.2 Detecção

Nesse passo o algoritmo aplica o conceito já conhecido e já apresentado nesse trabalho sobre CNN, nas 2000 regiões selecionadas pelo passo anterior.

3.1.3 Classificação

A cada execução de detecção o algoritmo utiliza classificação linear SVM para poder inserir rótulos para esses objetos com seus possíveis nomes.

SVM ou *Suport Vector Machine*, são algoritmos que tem como objetivo classificar um objeto. Eles utilizam de uma base inicial e com base nas características dessa base, é traçado uma reta onde delimita um objeto de outro em uma cena.

3.2 FAST R-CNN

Conforme observado por Girshick (2015), os principais pontos negativos são:

- Treinamento em multi estágios de processamento em cadeia;
- Devido ao fato de o sistema proposto sempre necessitar fazer escrita em disco, por causa do elevado gasto de memória, o processo para reconhecimento de certa de 4 mil imagens, leva cerda de dois dias e meio.
- A detecção de objetos é muito “lenta”. Para realizar toda a detecção em uma imagem, o sistema leva em por volta de 47 segundos.

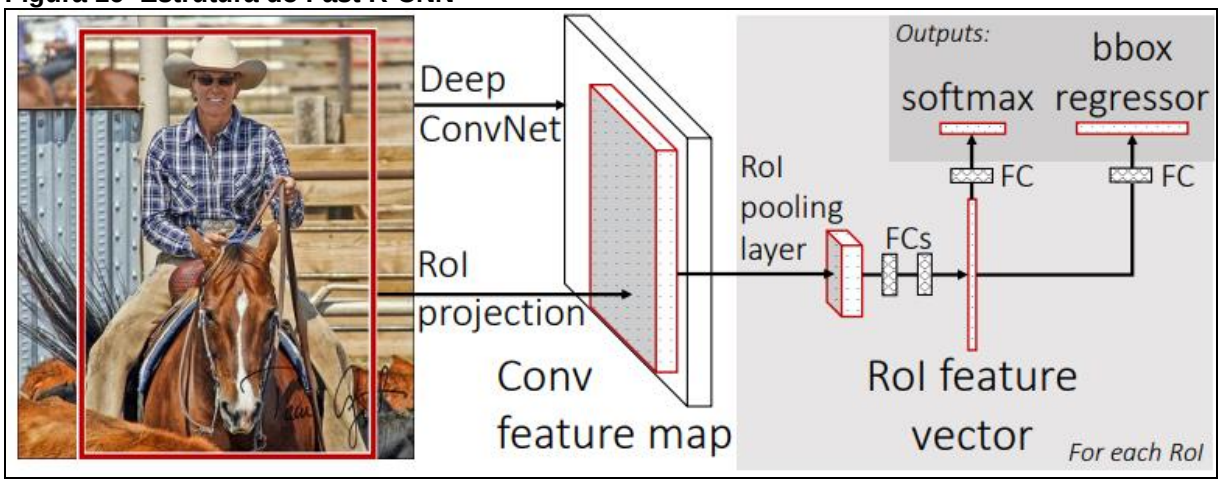
A partir disso, Girshick desenvolveu o método de Fast R-CNN, que corrige todos os pontos negativos citados anteriormente e melhora o desempenho em relação ao seu antecessor, R-CNN.

O Fast R-CNN funciona da seguinte forma: recebe como entrada uma imagem e um conjunto de possíveis objetos na cena. Então o sistema processa a imagem inteira com várias camadas de convolução para produzir um mapa de características. Em seguida, para cada possível objeto, é gerada uma camada de agrupamento de região de interesse e é extraído um vetor de recurso de tamanho fixo do mapa de característica.

Cada vetor é alimentado em uma sequência de camadas totalmente conectadas, que finalmente se ramificam em duas camadas de saída: uma que produz estimativas de classificação do objeto sobre K classes de objetos e outra

camada que gera a posição das caixas onde estão informados o objeto de cada K classe selecionada com posição em X, em Y, altura e largura. A Figura 16 demonstra de uma maneira simplificada como o algoritmo opera para o caso de uma imagem com apenas um possível objeto. (GIRSHICK, 2015).

Figura 16- Estrutura do Fast R-CNN



Fonte: Girshick (2015).

Nos testes realizados, onde as entradas eram iguais para ambos (R-CNN e Fast R-CNN), o Fast R-CNN obteve um desempenho de 4% em média, em relação a sua versão anterior.

Em testes referentes à velocidade, o Fast R-CNN se mostrou consideravelmente mais eficiente do que seu antecessor. A Tabela 3 apresenta os dados divulgados por Girshick sobre o teste relativo à velocidade do sistema novo em relação ao antigo.

Tabela 3- Comparação de desempenho entre Fast R-CNN e R-CNN

	Fast R-CNN			R-CNN		
	S	**M**	**L**	**S**	**M**	**L**
Tempo de treinamento(h)	1,2	2,0	9,5	22	28	84
Aumento na Velocidade do Treino	18,3X	14,0X	8,8X	1X	1X	1X
Taxa	0,10	0,15	0,32	9,8	12,1	47,0

Teste(s/im)						
Aumento na Velocidade do Teste	98X	80X	146X	1X	1X	1X
VOC07 mAP	57,1	59,2	66,9	58,5	60,5	66,0

Fonte: Adaptado de Girshick (2015)

3.3 YOU ONLY LOOK ONCE

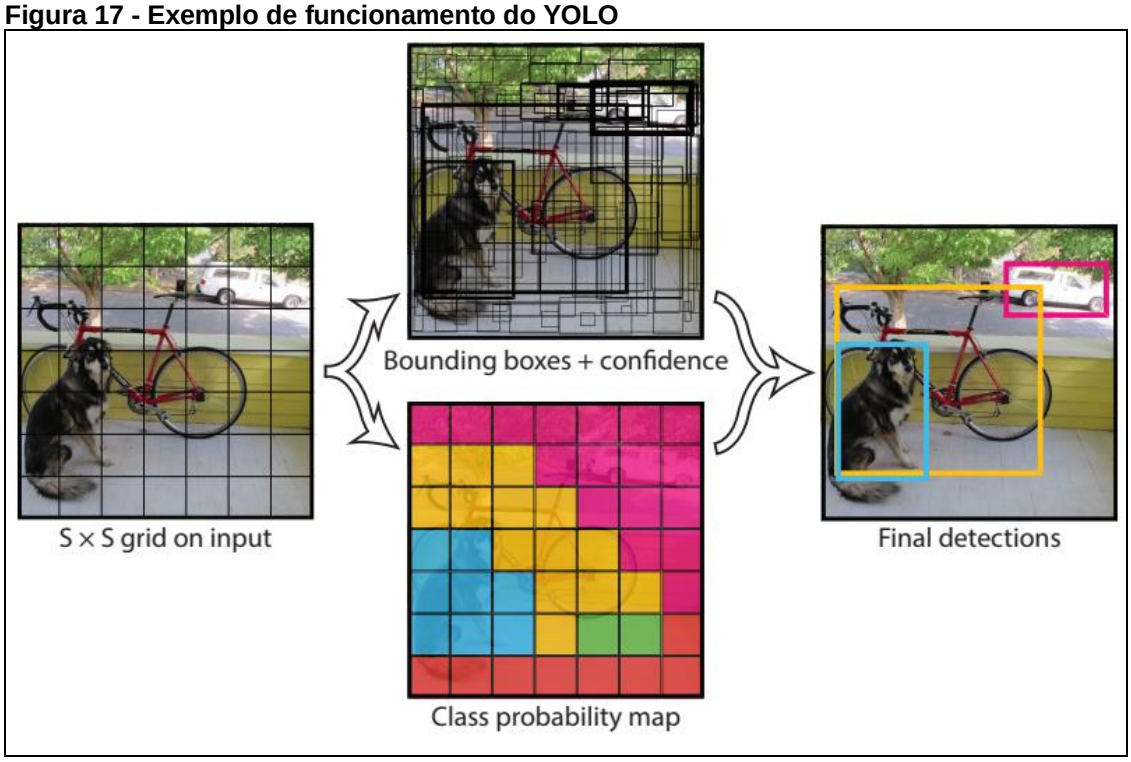
Redmon et al. (2016) desenvolveram um algoritmo capaz de fazer reconhecimento de imagem partindo por um caminho diferente de Girshick.

YOLO funciona da seguinte maneira: o primeiro passo é pegar a imagem de entrada, transformá-la em uma imagem com dimensões iguais a 448 x 448 e inserir essa imagem resultante em um *grid* S x S.

Em seguida é realizado uma operação de convolução para cada elemento desse *grid*. Desta convolução são geradas caixas que indicarão onde provavelmente haverá um objeto e será atribuído uma pontuação para aquela detecção específica. Essa pontuação será utilizada para gerar uma classe que conterá um mapa com as probabilidades de um dado objeto estar naquela cena. Caso nenhum possível objeto seja detectado em um dos elementos do *grid*, o valor atribuído a aquele *grid* será igual a zero.

O último passo é realizar uma operação matemática para definir a confiabilidade dos dados fornecidos por ambas as imagens (uma com a detecção de colisão e outra com o mapa de possíveis objetos), e inserir o quadrado nos possíveis objetos encontrados. A Figura 17 demonstra a aplicação dos passos descritos anteriormente.

Figura 17 - Exemplo de funcionamento do YOLO



Fonte: Redmon et al. (2016)

Após a etapa de localização dos objetos na imagem, a próxima é a de realizar o reconhecimento de quais objetos foram segmentados. O método convolucional utilizado por Redmon et al. (2016) é uma aplicação conjunta do trabalho de Szegedy et al. (2015) e Lin et al. (2013). No trabalho de Szegedy et al. (2015) é desenvolvido um algoritmo de nome *Inception*

3.4 RESUMO DO CAPÍTULO

Neste capítulo foram citados alguns outros trabalhos, cujo neles foram aplicadas técnicas de reconhecimento de objetos em imagens. Nele também foi detalhado sobre a progressão nos trabalhos de Girshick e Redmon que precederam as suas últimas versões (*Faster* RCNN e YOLOv3).

4 MÉTODO PROPOSTO

Para esse trabalho está sendo proposto a realização de um teste comparativo, a fim de verificar a diferença de desempenho entre os métodos apresentados, já desenvolvidos para reconhecimentos de objetos em uma cena específica.

Os métodos serão sujeitos a um teste de precisão e desempenho. O objetivo desse teste servirá para comprar a eficiência desses métodos em um determinado cenário, ainda não testado pelos autores.

A máquina utilizada para a realização dos testes possui um i5 8500 3,00GHz, 8 Gb de memória RAM Ddr4 2666mhz, placa de vídeo Nvidia GeForce GTX 1660 Ti 6gb Ddr6 192bits e placa mãe Asus TUF Z390-plus Gaming.

Os testes serão realizados em etapas. Em cada etapa será selecionando uma das bases de imagens e executado os 2 algoritmos de reconhecimento de objetos comparando seus resultados, sendo esses resultados, tempo levado para a realização de todos os reconhecimentos, quantidade de memória RAM e dedicada utilizadas durante cada processo e a precisão de reconhecimento do objeto obtido. É importante a realização dessas comparações, pois durante as etapas de teste pode ter um caso em que um algoritmo pode levar menos da metade do tempo para realizar os reconhecimentos em relação ao outro algoritmo, porém o mesmo pode apresentar uma taxa de precisão consideravelmente inferior a obtida em comparação com o outro algoritmo.

Antes de iniciar o processo de comparação entre os dois algoritmos primeiro foi necessário realizar a etapa de treinamento e validação. Para ambos os algoritmos o processo de treinamento e validação foi feito utilizando a base MS COCO (LIN, 2014), na versão do ano de 2017 e o conjunto de dados da base Pacal VOC ano de 2007.

4.1 TREINAMENTO E VALIDAÇÃO YOLOV3

Na máquina descrita anteriormente a etapa de treinamento e validação para aprendizado, levou por volta de 3 (três) dias de execução na GPU. O processo foi feito com auxílio de softwares como CAFFE (framework utilizado por Redmon e

Farhadi (2018) que possui implementações já otimizadas para *Deep Leaning* utilizando a GPU), CUDA 10.1, CUDNN 7.6 (dois softwares essências para a execução de processamento de dados utilizando placas de vídeo NVIDIA) e Anaconda (software utilizado para a compilação do PYCAFFE e de outros cujo era necessário a linguagem de programação Python). Além destes softwares foram instaladas bibliotecas para execução de código em C++ e algumas outras auxiliares para a linguagem Python.

O sistema operacional utilizado para o treinamento e para a validação da aprendizagem foi o Ubuntu versão 18.04.3 LTS. Essa informação será repetida posteriormente, no trabalho, pois os testes também foram realizados no mesmo sistema operacional com a mesma versão, incluindo suas dependências.

Algumas alterações nos arquivos de configuração original do yolov3 para a execução do treinamento e validação. O arquivo de configuração (.cfg) escolhido foi o yolov3.cfg, pois é a versão que está sendo mencionada nesse trabalho em vários momentos.

As alterações necessárias foram em relação ao número de subdivisões (*subdivisions*) que cada imagem teria, foi alterado para 16, pois ao tentar deixar com uma quantidade menor de subdivisões, o algoritmo estava exibindo uma mensagem sobre falta de memória. Não era interessante para essa etapa deixar uma quantia de 32 subdivisões, pois isso apenas faria com que o processo de treinamento e validação demorasse mais tempo. E em relação ao tamanho de cada lote (*batch*), no qual acusava o mesmo erro caso deixasse em 64.

O primeiro bloco do arquivo yolov3.cfg utilizado ficou escrito da seguinte maneira:

```
[net]
# Training
batch=32
subdivisions=16
width=608
height=608
channels=3
momentum=0.9
decay=0.0005
angle=0
```

saturation = 1.5
exposure = 1.5
hue=.1
[...]

O yolov3 possui um arquivo chamado coco.names, nesse arquivo estão escritas todas as classes que o algoritmo consegue detectar em uma imagem, são elas: *person, bicycle, car, motorbike, aeroplane, bus, train, truck, boat, traffic light, fire hydrant, stop sign, parking meter, bench, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe, backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake, chair, sofa, pottedplant, bed, diningtable, toilet, tvmonitor, laptop, mouse, remote, keyboard, cell phone, microwave, oven, toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear, hair drier e toothbrush.*

O yolov3, foi implementado com um método de pesos parciais, ou seja, caso o usuário queira encerrar o treinamento e a validação das imagens antes do fim da realização automática em todo o *conjunto de dados* selecionado, o algoritmo salva vários arquivos com os pesos escritos em momentos específicos. Apenas para fim de verificação da evolução da atualização (melhora no processo de reconhecimento de um objeto) dos pesos de cada classe em relação a quantidade de imagens processadas, foram feitos testes usando 10 imagens em cada um desses momentos. Essas imagens estão presentes na Figura 18.

Um ponto importante a mencionar é, que a única diferença entre os reconhecimentos que serão apresentados, é o valor dos pesos que o algoritmo possui para reconhecimento de um dado objeto de uma classe naquele momento específico. Não foi alterado nada além disso entre os reconhecimentos realizados pelo algoritmo yolov3.

Figura 18 - 10 imagens selecionadas para detecção de objetos

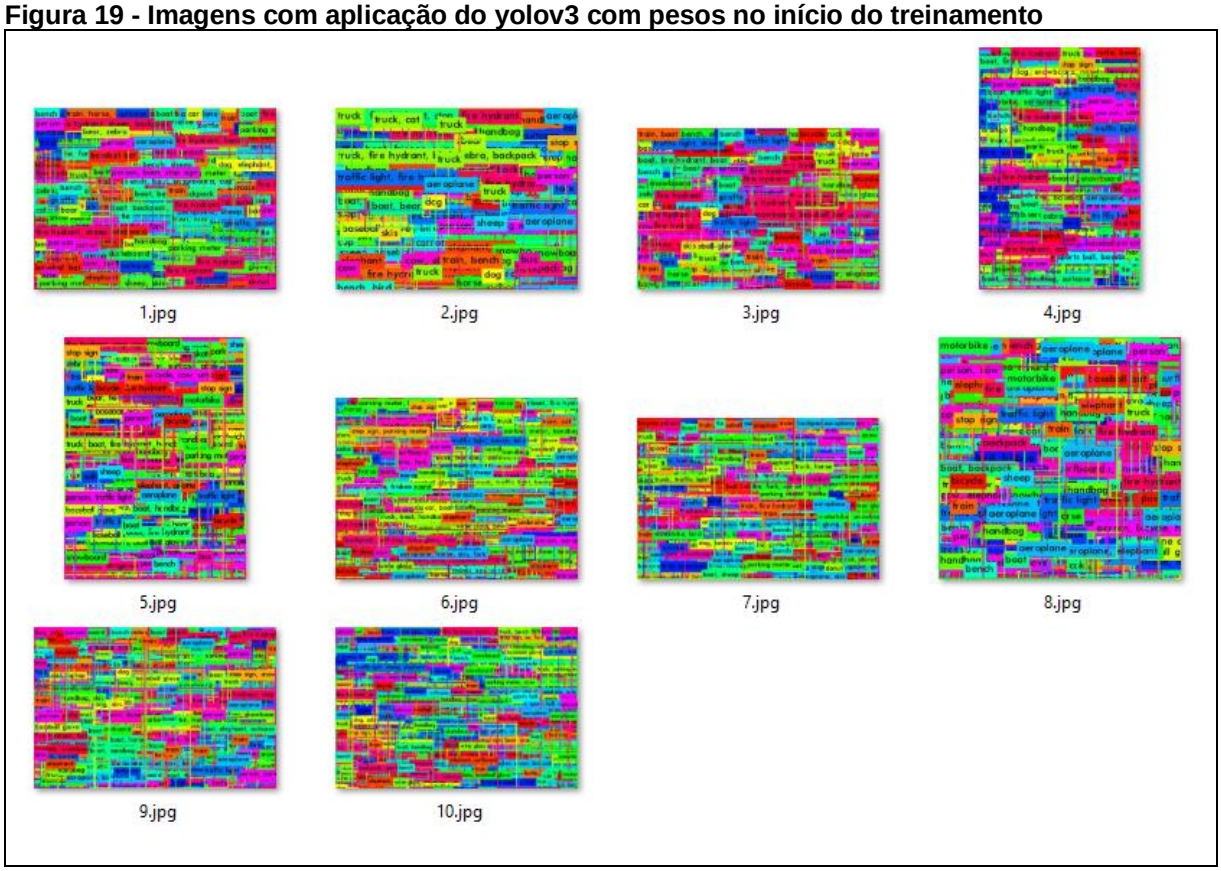
Fonte: Autoria própria.

4.1.1 Algoritmo com nenhum dos pesos treinados

Yolov3_100.weights é o primeiro arquivo criado durante a fase treinamento do YOLOv3, no momento de sua criação, o algoritmo foi treinado com menos de 200 imagens. Sabendo que são 80 classes, pode ter ocorrido um caso que uma das classes não foi apresentada a nenhuma imagem no qual essa classe aparece, então ela não possui pesos definidos, podendo assim, ser literalmente qualquer conjunto de pixels presentes nas imagens.

O resultado da aplicação do algoritmo yolov3 com o arquivo de configuração yolov3.cfg com esses pesos nas imagens da Figura 18 é a que está representada na Figura 19 a seguir.

Figura 19 - Imagens com aplicação do yolov3 com pesos no início do treinamento



Fonte: Autoria própria.

4.1.2 Algoritmo com pesos pouco treinados

Nesse ponto, o treinamento do algoritmo provavelmente exibiu várias imagens de pessoa com jaqueta a uma distância média/longa (suposição feita com base no resultado apresentado), pois no resultado apresentado pelo algoritmo ele foi capaz de reconhecer nas imagens 3 e 10, pessoas com essa aparência. A Figura 20 apresenta a imagem 3 e 10, com o reconhecimento feito pelo algoritmo. Vale notar que nas imagens 2, 4 e 5 também estão presentes pessoas (na imagem 10 existem mais pessoas, mas como o padrão reconhecido é pessoa com roupa azul, elas não foram selecionadas também), porém o algoritmo ainda não foi capaz de reconhecer esse padrão para identifica-los corretamente.

Figura 20 - Reconhecimento de pessoas pelo algoritmo yolov3 pré-treinado

Fonte: Autoria própria.

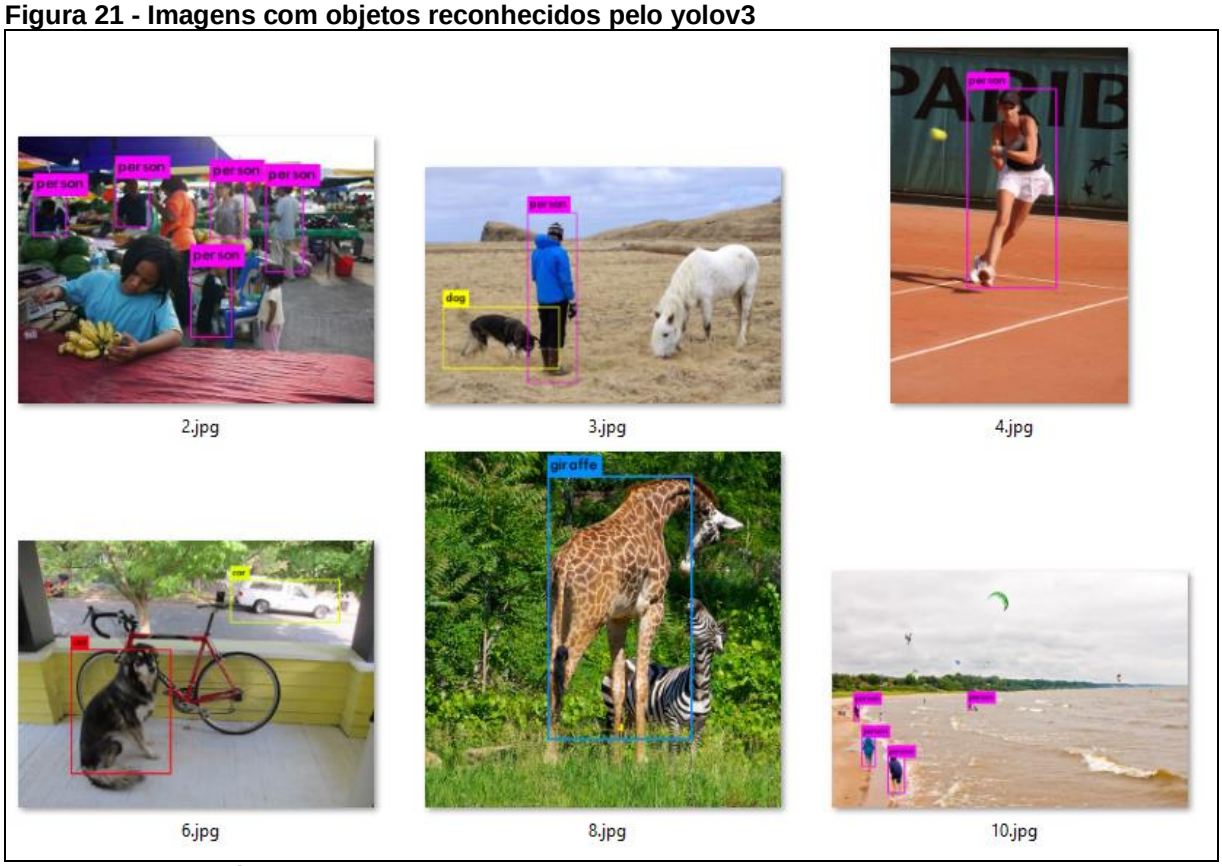
4.1.3 Algoritmo com aproximadamente 10 porcento do treinamento realizado

Aqui foi onde ocorreu a segunda maior mudança de reconhecimento. Nos pesos anteriores a esse, o algoritmo tinha parado de reconhecer a pessoa de azul na imagem 3 e 10, voltando assim a exibir um retorno semelhante a Figura 18. Nos valores para os pesos atuais é possível perceber, através da Figura 21, que algoritmo reconhece mais de um objeto em uma mesma cena.

Alguns pontos interessantes para se notar na Figura 21, são:

- Na imagem 2, mesmo a pessoa estando mais próxima (mais presente) na imagem, o algoritmo ainda não conseguiu a reconhecer;
- Na imagem 3 o mesmo animal que o algoritmo reconheceu como cachorro, na imagem 6 ele o reconheceu sendo um gato;
- Pela bicicleta estar sobreposta pelo cachorro (vulgo gato, segundo o algoritmo) na imagem 6, o yolov3 não conseguiu reconhece-la na imagem exibida.

Figura 21 - Imagens com objetos reconhecidos pelo yolov3



Fonte: Autoria própria.

4.1.4 Algoritmo com aproximadamente 30 porcento do treinamento realizado

Nesse ponto o algoritmo já está convergindo a um ponto em que os pesos estão começando a ficar bem definidos. Na Figura 22 é possível perceber que o yolov3 reconhecer ao menos um objeto em cada imagem. Como ele não reconheceu todos os objetos integralmente, se a execução do treinamento se encerrasse com esse resultado, seu mAP seria baixo, por não acertar todos os objetos corretamente na cena.

Figura 22 - Yolov3 reconhecendo quase todos os objetos presentes nas imagens

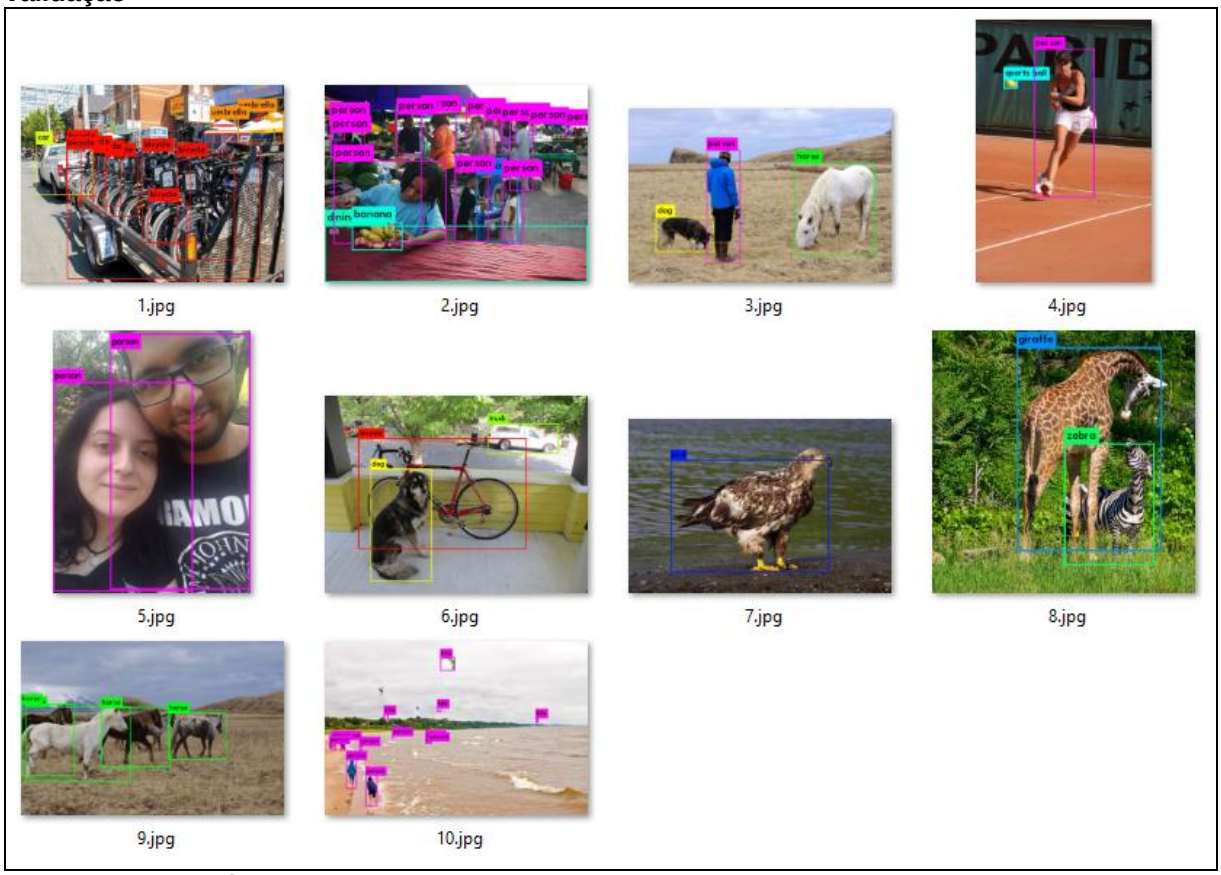
Fonte: Autoria própria.

Importante observar que o algoritmo ainda não está reconhecendo corretamente o cachorro presente na imagem 6. Em um momento anterior, o algoritmo apresentou dúvida se o objeto era um cachorro ou era um gato, porém depois ele voltou a acreditar que o objeto era somente um cachorro (segundo o próprio algoritmo ele reconhece aquele objeto como um gato com 69% de certeza, o que não é um valor alto, ou seja, ainda possui uma “dúvida” sobre tal rótulo por parte do algoritmo).

4.1.5 Pesos totalmente atribuídos após o processo de treinamento

Esse arquivo é um arquivo com o algoritmo YOLO utilizando o método de camadas convolucionais Darknet53 totalmente treinado e validado com base no conjunto de dados COCO versão do ano de 2017. A Figura 23 apresenta o resultado da execução do algoritmo nas 10 imagem apresentadas na Figura 18.

Figura 23 - Yolov3 reconhecendo objetos em uma cena após todo o processo de treinamento e validação



Fonte: Autoria própria.

É possível notar que mesmo concluído o algoritmo não apresentou corretamente todos os objetos que ele é capaz de reconhecer na cena (vide imagem 1), chamou um carro de caminhão (imagem 6) e na imagem 10, chamou os paragliders de pipas. Para esse último caso, mesmo ele não possuindo uma classe específica para paragliders, o comportamento correto para o algoritmo seria de não ter reconhecido aqueles objetos, assim como fez com os óculos na imagem 5.

Esses erros de reconhecimentos notados nessas imagens, afirma o mencionado por Redmon e Farhadi (2018), ao mostrarem o algoritmo produzido por eles não possui erros, mas sim que os erros que ele possui são poucos.

4.2 TREINAMENTO E VALIDAÇÃO FASTER RCNN

A etapa de treinamento do algoritmo *Faster* RCNN foi realizada utilizando o modelo VGG16, que segundo o autor do algoritmo é o modelo que provavelmente

apresentaria o melhor resultado (o mais "completo"). O treinamento demorou por volta de 1 dia utilizando esse modelo.

A escolha do modelo VGG16 também está relacionado com o fato de o treinamento do algoritmo YOLOv3 foi feito utilizando o modelo Darknet53 (o mais "completo") que, segundo o autor do mesmo, seria o modelo que provavelmente apresentaria o melhor resultado.

A quantidade de memória de vídeo utilizada durante o treinamento foi de aproximadamente 2Gb (metade da memória utilizada pelo YOLOv3 na mesma etapa), o processamento tanto da GPU como da CPU se manteve na média de 97 porcento nessa fase.

4.3 RESUMO DO CAPÍTULO

Neste capítulo foi demonstrado o processo de treinamento de ambos os algoritmos (tanto o *Faster* RCNN quanto o YOLOv3), exibindo, no caso do YOLOv3, a evolução no reconhecimento dos objetos nas imagens de exemplo, a dificuldade do algoritmo no processo de reconhecimento de alguns objetos e a identificação errada do algoritmo por falta de exemplos o suficiente.

5 RESULTADOS E DISCUSSÕES

Durante a realização dos testes comparativos, os processos executando no computador no momento dos testes eram os que estão informados na Tabela 4.

Tabela 4 - Processos executando antes do início dos testes

PID	%CPU	%MEM	TIME+	COMMAND
1178	0,7	0	0:08.63	irq/143-nvidia
2424	0,3	0,1	0:00.17	top
1	0	0,1	0:01.10	systemd
2	0	0	0:00.00	kthreadd
3	0	0	0:00.00	rcu_gp
4	0	0	0:00.00	rcu_par_gp
6	0	0	0:00.00	kworker/0:0H-kb
8	0	0	0:00.00	mm_percpu_wq
9	0	0	0:00.00	ksoftirqd/0
10	0	0	0:00.25	rcu_sched
11	0	0	0:00.00	migration/0
12	0	0	0:00.00	idle_inject/0
13	0	0	0:00.05	kworker/0:1-eve
14	0	0	0:00.00	cpuhp/0
15	0	0	0:00.00	cpuhp/1
16	0	0	0:00.00	idle_inject/1
17	0	0	0:00.00	migration/1
18	0	0	0:00.00	ksoftirqd/1
20	0	0	0:00.00	kworker/1:0H-kb
21	0	0	0:00.00	cpuhp/2
22	0	0	0:00.00	idle_inject/2
23	0	0	0:00.00	migration/2
24	0	0	0:00.00	ksoftirqd/2
26	0	0	0:00.00	kworker/2:0H-kb
27	0	0	0:00.00	cpuhp/3
28	0	0	0:00.00	idle_inject/3
29	0	0	0:00.00	migration/3

30	0	0	0:00.00	ksoftirqd/3
32	0	0	0:00.00	kworker/3:0H-kb
33	0	0	0:00.00	cpuhp/4
34	0	0	0:00.00	idle_inject/4
35	0	0	0:00.00	migration/4
36	0	0	0:00.00	ksoftirqd/4
38	0	0	0:00.00	kworker/4:0H-kb
39	0	0	0:00.00	cpuhp/5
40	0	0	0:00.00	idle_inject/5
41	0	0	0:00.00	migration/5
42	0	0	0:00.00	ksoftirqd/5
43	0	0	0:00.00	kworker/5:0H
44	0	0	0:00.00	kdevtmpfs
45	0	0	0:00.00	netns
46	0	0	0:00.00	rcu_tasks_kthre
47	0	0	0:00.00	kauditd
48	0	0	0:00.00	Khungtaskd

Fonte: Autoria própria.

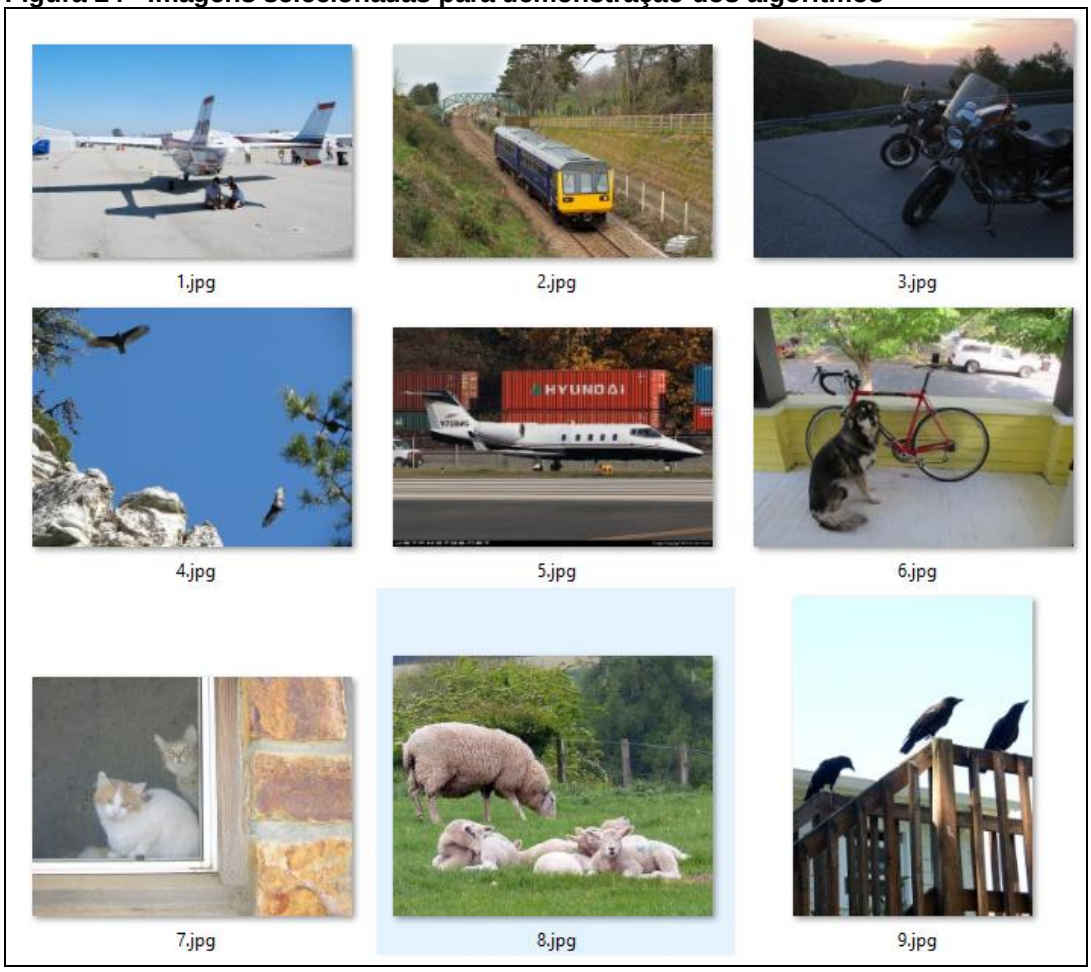
Como não foi informado na tabela por falta de espaço, as seguintes informações podem ser relevantes:

- O único processo pertencente ao usuário é o processo com PID 2424. Todos os outros pertenciam ao root;
- Coletado esses dados foi dado início a execução dos testes, sendo assim, outros processos mais pesados foram iniciados. Como esses processos fazem parte do teste, independente em qual máquina ele possa ser reproduzido, eles não foram inseridos na tabela.

Para melhor visualização da comparação entre os dois algoritmos, foi realizado o teste em uma sequência de 9 imagens escolhidas aleatoriamente da base de testes (vide Figura 24), que serão melhor descritas no decorrer desse capítulo. Os testes comparativos também foram realizados utilizando toda o conjunto de dados de validação do COCO 2014 e Pascal VOC 2007. Os resultados referentes

ao teste dos algoritmos realizados nesse conjunto de dados estão presentes na Tabela 6.

Figura 24 - Imagens selecionadas para demonstração dos algoritmos



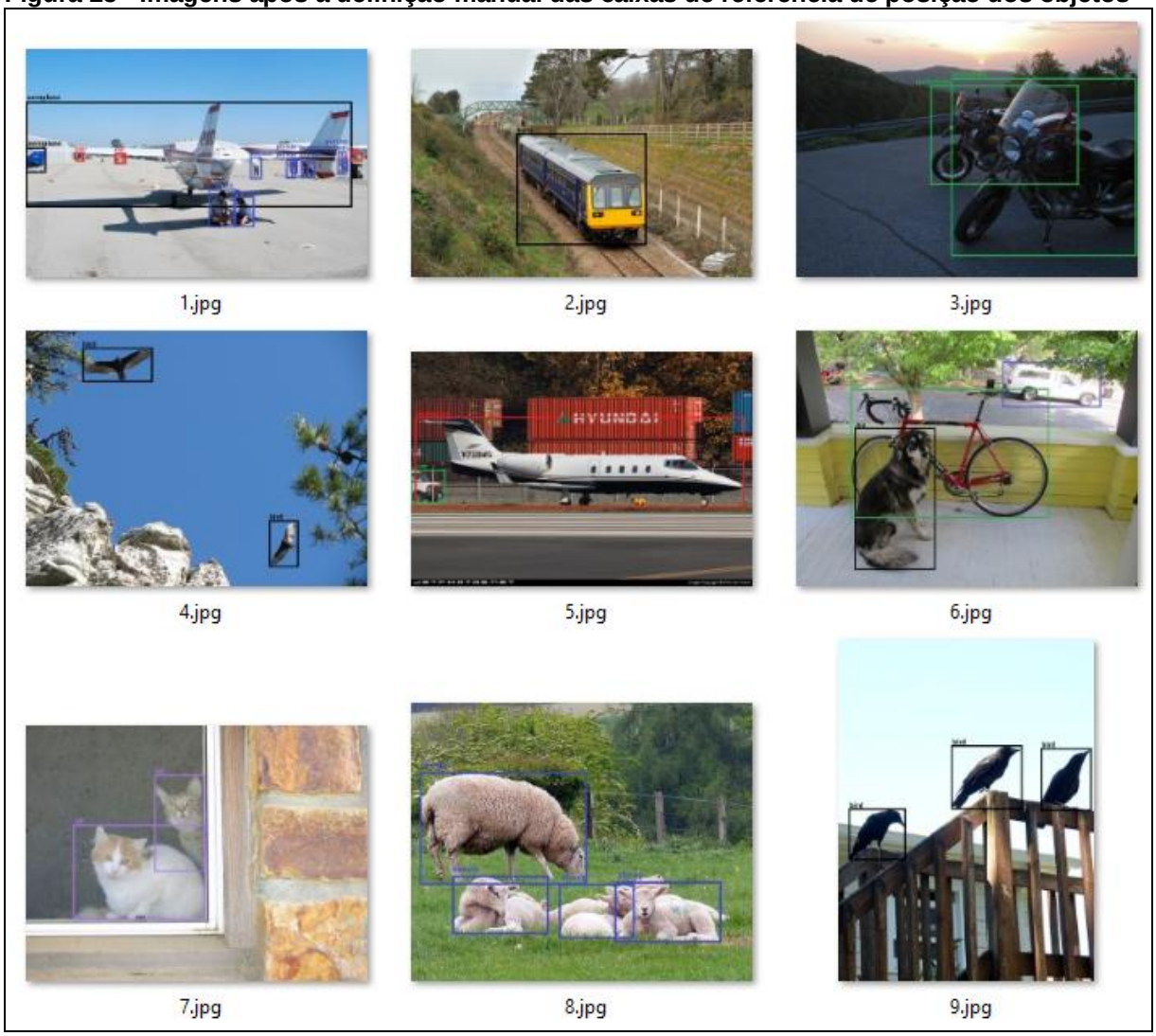
Fonte: Autoria própria.

A medida comparativa utilizada será a mAP, medida que é normalmente utilizada nas competições que utilizam o conjunto de dados COCO.

Para a realização dos testes foi definido quais seriam os critérios a serem observados em cada algoritmo. Os escolhidos para serem comparados na etapa de teste foram: Tempo de execução para cada imagem, Precisão na detecção do objeto (definido pelo mAP).

Antes de dar início aos testes foi necessário marcar manualmente nas 9 imagens escolhidas, quais seriam os objetos que, com base em olhos humanos, estariam presentes na cena para detecção. A Figura 25 demostra as 9 imagens selecionadas na Figura 24, porém com as caixas de detecção desenhadas manualmente.

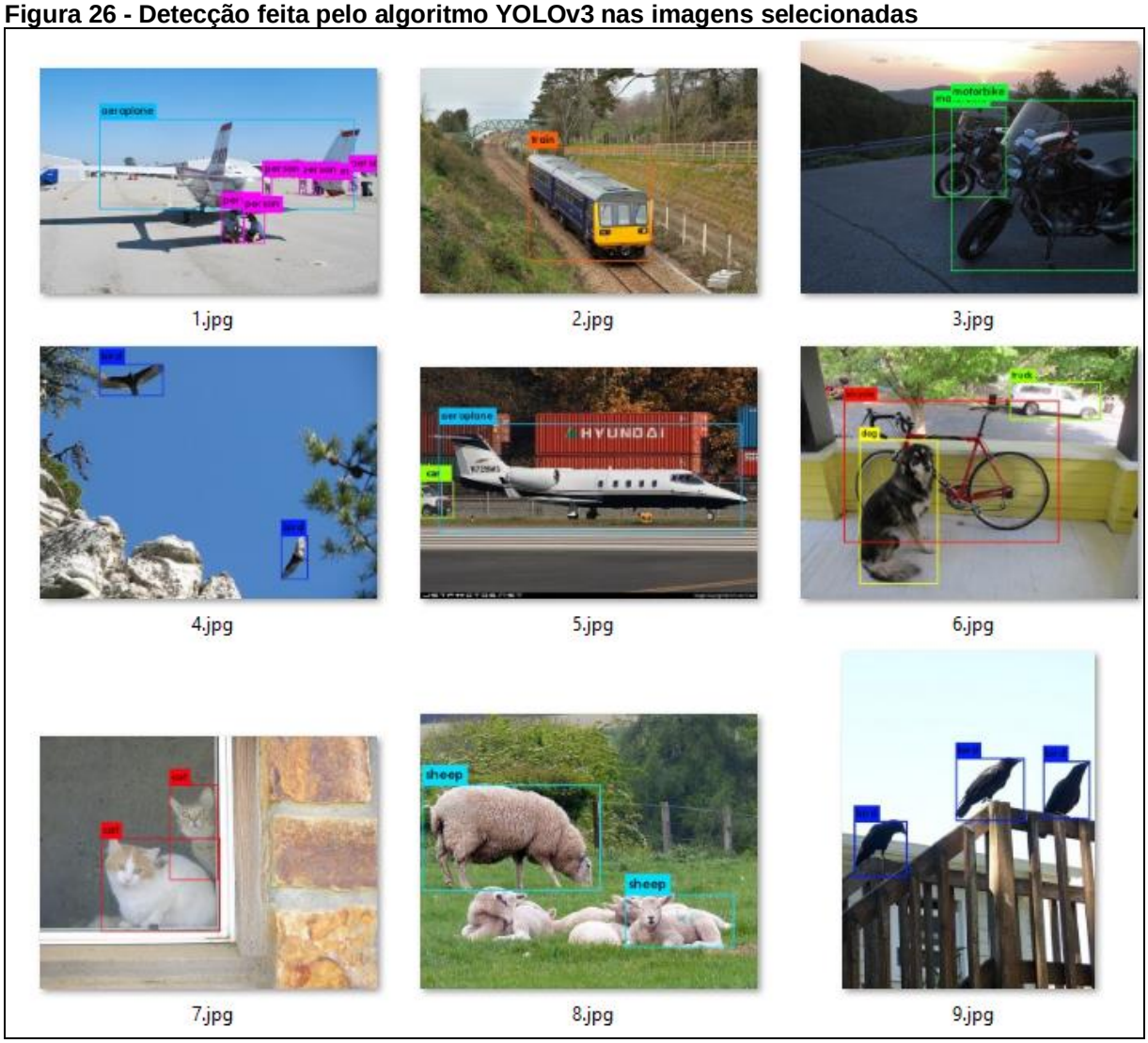
Figura 25 - Imagens após a definição manual das caixas de referência de posição dos objetos



Fonte: Autoria própria.

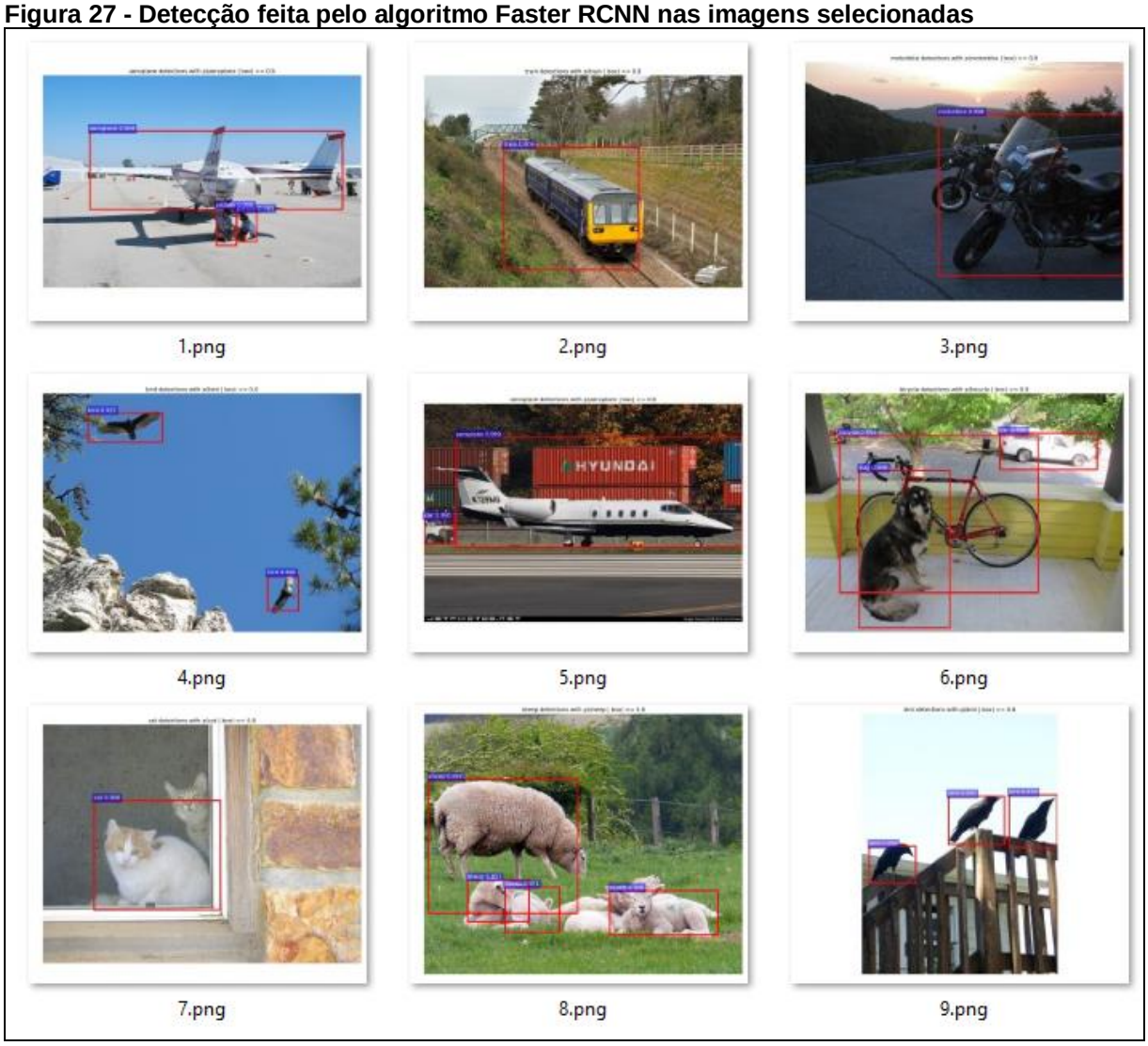
Tendo como base as imagens da Figura 24 e seus objetos devidamente marcados na Figura 25, foi dado o início na etapa de reconhecimentos automático por parte dos algoritmos YOLOv3 e *Faster* RCNN. O resultado do reconhecimento feito pelo algoritmo YOLOv3 é apresentado na Figura 26. Na Figura 27 é apresentado o resultado apresentado pelo algoritmo *Faster* RCNN.

Figura 26 - Detecção feita pelo algoritmo YOLOv3 nas imagens selecionadas



Fonte: Autoria própria.

Figura 27 - Detecção feita pelo algoritmo Faster RCNN nas imagens selecionadas



Fonte: Autoria própria.

O resultado obtido para cada algoritmo com relação a taxa de precisão (AP) por imagem, assim como o tempo de execução gasto para os reconhecimentos desses objetos nas imagens selecionadas estão presentes na Tabela 5. Ao final da Tabela 5 é apresentado o mAP e o tempo médio para a execução de cada uma das 9 imagens selecionadas.

Tabela 5 - Relação entre a precisão média e tempo de execução do YOLOv3 e Faster RCNN

	YOLOv3		Faster RCNN	
	AP%	Tempo de execução (s)	AP%	Tempo de execução (s)
1.jpg	39,78	0,290678	14,98	0,129
2.jpg	96,45	0,072362	80,96	0,120

3.jpg	71,57	0,069828	43,58	0,124
4.jpg	98,78	0,070980	91,51	0,124
5.jpg	88,08	0,070928	85,77	0,109
6.jpg	92,04	0,070191	91,74	0,111
7.jpg	92,21	0,077540	33,79	0,091
8.jpg	43,85	0,070483	67,58	0,128
9.jpg	90,31	0,072788	81,34	0,114
Média	**79,23**	**0,961975**	**65,69**	**0,1167**

Fonte: Autoria própria.

Feito essa análise mais específica em um grupo pequeno de imagens, a Tabela 6 apresenta o mAP dos algoritmos tendo como base o conjunto de dados COCO 2017 e Pascal VOC 2007.

Tabela 6 - Comparação dos dados nas bases do MS COCO 2017 e Pacal VOC 2007

	mAP% em COCO 2017	mAP% em Pascal VOC 2007
YOLOv3	42,60	74,41
Faster RCNN VGG16	21,94	60,75

Fonte: Autoria própria.

É de conhecimento que existem trabalhos como Detectron (GIRSHICK et al, 2018) e Mask R-CNN (HE et al, 2017) já desenvolvidos e que esses trabalhos seriam uma evolução dos modelos utilizados durante a elaboração desse trabalho. Porém como estes eram os únicos trabalhos desenvolvidos seguindo esse caminho, não foi possível pesquisar outros que seguissem um caminho semelhante para poder realizar a comparação de eficiência entre eles.

6 CONCLUSÕES E TRABALHOS FUTUROS

Com base em toda a etapa de treinamento, validação e testes desenvolvidos, é possível concluir que o algoritmo *You Only Look Once* versão 3 demonstrou um desempenho acima do outro algoritmo tanto no quesito precisão do objeto selecionado quanto no quesito velocidade de processamento.

Essa diferença entre eles pode e deve ser levada em consideração a data de publicação de seus trabalhos nas versões finais, pode ocorrer o caso de o YOLOv3 possuir um desempenho superior por ter sido publicado 2 anos após o outro algoritmo, ou pelo autor do mesmo ter utilizado e citado em suas pesquisas o algoritmo de *Faster* RCNN.

O Algoritmo *Faster* RCNN possui um processo de treinamento mais rápido do que seu "concorrente", isso acontece pois durante a pesquisa desenvolvida por Girshick ele dividiu o algoritmo em 2 etapas visando otimizar os dois processos (treinamento e teste). Além disso, seu maior modelo aplicado ocupa por volta de 1,5Gb a menos do que o YOLOv3.

Por fim, *Faster* RCNN é mais completo e possui uma etapa de treinamento otimizada, porém YOLOv3 apresenta melhores resultados e desempenho no quesito tempo levado para realizar o reconhecimento dos objetos na cena.

REFERÊNCIAS

AKCAY, Samet et al. Using Deep Convolucional Neural Network Architectures for Object Classification and Detection Within X-Ray Baggage Security Imagery**. IEEE Transactions on Information Forensics and Security**, v. 13, n. 9, p. 2203-2215, 2018.

AGATONOVIC-KUSTRIN, S.; BERESFORD, R. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. **Journal of pharmaceutical and biomedic /al analysis**, v. 22, n. 5, p. 717-727, 2000.

BEAR, Mark F.; CONNORS, Barry W.; PARADISO, Michael A. **NeuroCiência: desvendando o sistema nervoso**. Artmed Editora, 2008.

BOJARSKI, Mariusz et al. End to End Learning for Self-Driving Cars. **arXiv preprint arXiv**: 1604.07316, 2016.

CHEN, Liang-Chieh et al. Deeplab: Semantic image segmentation with deep Convolucional nets, atrous convolution, and fully connected crfs. **IEEE transactions on pattern analysis and machine intelligence**, v. 40, n. 4, p. 834-848, 2018.

COSER, Leandro. Filtro de difusão anisotrópico orientado por evidência de borda. 2009.

EVERINGHAM, Mark et al. The pascal visual object classes (voc) challenge. **International journal of computer vision**, v. 88, n. 2, p. 303-338, 2010.

GARDNER, Matt W.; DORLING, S. R. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. **Atmospheric environment**, v. 32, n. 14-15, p. 2627-2636, 1998.

GIRSHICK, Ross et al. Detectron. 2018.

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.

GIRSHICK, Ross. Fast r-cnn. In: **Proceedings of the IEEE international conference on computer vision**. 2015. p. 1440-1448.

GOODFELLOW, Ian et al. **Deep learning**. Cambridge: MIT press, 2016.

HAYKIN, Simon. **Redes neurais: princípios e prática**. Bookman Editora, 2007.

HE, Kaiming et al. Mask r-cnn. In: **Proceedings of the IEEE international conference on computer vision**. 2017. p. 2961-2969.

KOVÁCS, Zsolt László. **Redes neurais artificiais**. Editora Livraria da Fisica, 2002.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep Convolucional neural networks. In: **Advances in neural information processing systems**. 2012. p. 1097-1105.

LECUN, Yann et al. Backpropagation applied to handwritten zip code recognition. **Neural computation**, v. 1, n. 4, p. 541-551, 1989.

LI, Jianan et al. Scale-aware fast R-CNN for pedestrian detection. **IEEE Transactions on Multimedia**, v. 20, n. 4, p. 985-996, 2018.

LIN, Min; CHEN, Qiang; YAN, Shuicheng. Network in network. **arXiv preprint arXiv:1312.4400**, 2013.

LIN, Tsung-Yi et al. Microsoft coco: Common objects in context. In: **European conference on computer vision**. Springer, Cham, 2014. p. 740-755.

OSÓRIO, Fernando S.; BITTENCOURT, João Ricardo; OSÓRIO, Fernando Santos. Sistemas Inteligentes baseados em redes neurais artificiais aplicados ao processamento de imagens. In: **I Workshop de inteligência artificial**. 2000.

REDMON, Joseph 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, Joseph; FARHADI, Ali. Yolov3: An incremental improvement. arXiv preprint **arXiv:1804.02767**, 2018.

REN, Shaoqing et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: **Advances in neural information processing systems**. 2015. p. 91-99.

SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep Convolucional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SONKA, Milan; HLAVAC, Vaclav; BOYLE, Roger. **Image processing, analysis, and machine vision**. Cengage Learning, 2014.

SZEGEDY, Christian et al. Going deeper with convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. 2015. p. 1-9.

UIJLINGS, Jasper RR et al. Selective search for object recognition**. International journal of computer vision**, v. 104, n. 2, p. 154-171, 2013.

WIDROW, Bernard; LEHR, Michael A. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. **Proceedings of the IEEE**, v. 78, n. 9, p. 1415-1442, 1990.

YEGNANARAYANA, B. **Artificial neural networks**. PHI Learning Pvt. Ltd., 2009.

ZEILER, Matthew D.; FERGUS, Rob. Visualizing and understanding Convolucional networks. In: **European conference on computer vision**. springer, Cham, 2014. p. 818-833.

ZHANG, Qi-xing et al. Wildland Forest Fire Smoke Detection Based on Faster R-CNN using Synthetic Smoke Images. **Procedia engineering**, v. 211, p. 441-446, 2018.

ZHU, Qikui et al. Deeply-supervised CNN for prostate segmentation. In: **Neural Networks (IJCNN), 2017 International Joint Conference on**. IEEE, 2017. p. 178-184.