Guilherme Vinicius Simões Cardoso

Detecção de Armas de Fogo em Imagens Baseada em Redes Neurais Convolucionais

Brasil

Vitória, 2021

Guilherme Vinicius Simões Cardoso

Detecção de Armas de Fogo em Imagens Baseada em Redes Neurais Convolucionais

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Universidade Federal do Espírito Santo

Programa de Pós-Graduação em Engenharia Elétrica

Orientador: Prof. Dr. Patrick Marques Ciarelli

Coorientador: Prof.ª Dr.ª Raquel Frizera Vassallo

Brasil

Vitória, 2021

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Cardoso, Guilherme Vinicius Simões, 1993-
C268d Detecção de armas de fogo em imagens baseada em redes neurais convolucionais / Guilherme Vinicius Simões Cardoso. - 2021.
72 f. : il.

Orientador: Patrick Marques Ciarelli.
Coorientadora: Raquel Frizera Vassallo.
Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Inteligência artificial. 2. Redes neurais (Computação). 3. Aprendizado do computador. 4. Processamento de imagens. 5. Armas de fogo. I. Ciarelli, Patrick Marques. II. Vassallo, Raquel Frizera. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 621.3

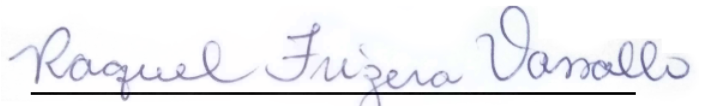
Guilherme Vinicius Simões Cardoso

Detecção de Armas de Fogo em Imagens Baseada em Redes Neurais Convolucionais

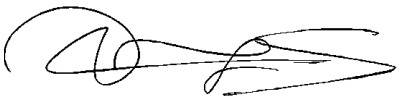
Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Engenharia Elétrica.

Trabalho aprovado. Brasil, 30 de abril de 2021:

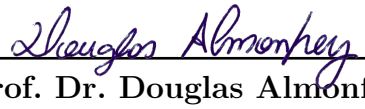
Prof. Dr. Patrick Marques Ciarelli
Orientador



Prof.ª Dr.ª Raquel Frizera Vassallo
Coorientadora



Prof. Dr. Jorge Leonid Aching Samatelo
Universidade Federal do Espírito Santo



Prof. Dr. Douglas Almonfrey
Instituto Federal do Espírito Santo

Brasil
Vitória, 2021

Com gratidão, dedico este trabalho a Deus.
Devo a Ele tudo o que sou.

Agradecimentos

Agradeço primeiramente a Deus, por todas as conquistas, por todas as bênçãos recebidas, por estar comigo nas horas mais difíceis, por nunca ter me abandonado. Agradeço por ter me orientado e sustentado ao longo da minha vida, pois sem Ele não teria chegado até aqui.

Gostaria de agradecer também aos meus professores, passados e atuais. Todos aqueles que contribuíram de alguma forma em algum momento da minha vida para o meu crescimento profissional ou pessoal. Especialmente, gostaria de agradecer aos meus orientadores, o professor Patrick Marques Ciarelli e a professora Raquel Frizera Vassallo, pela paciência, ensinamentos e tempo disposto nas reuniões que me ajudaram a trilhar esta minha fase do mestrado.

Agradeço aos professores Jorge e Douglas por terem aceitado avaliar minha dissertação. É de grande importância para mim que grandes nomes possam fazer contribuições e críticas ao meu trabalho.

Agradeço também à CAPES, pela apoio financeiro e ao Programa de Pós-Graduação em Engenharia Elétrica (PPGEE) da UFES pela oportunidade do mestrado.

Um agradecimento especial ao Bruno, Catherine, Demuth, Fernando, Guilherme, Kaique, Matheus, Thais, Thales, e para todo o pessoal do CISNE! Agradeço pelas conversas, pela amizade, pelos momentos únicos no laboratório, por todo o conhecimento e, principalmente, pelas longas conversas ao longo desses anos. Muito obrigado pelos momentos incríveis neste meio tempo, sentirei muita falta desses momentos!

À minha noiva, Thamires Rohr, pela parceria, companheirismo, amor e compreensão nos últimos anos. Ao meu irmão, Pablo Cardoso, por me acompanhar nas muitas madrugadas ao longo desses anos, muito obrigado. À toda minha igreja, pelas palavras de incentivo, pelo amor e por cada oração, vocês também fazem parte desta conquista.

Um agradecimento especial aos profissionais da área da saúde, que em meio ao caos que estamos vivendo, lutam por nós e pela vida todos os dias! Muito obrigado, pois de forma indireta, vocês contribuíram para a finalização dessa dissertação e, de forma direta, pela volta do meu pai.

Por último, mas de forma alguma menos importante, agradeço aos meus pais. Atribuo todo e qualquer sucesso profissional e pessoal que tenho ou venha a ter à eles. Agradeço pelo amor, pelo carinho, pela confiança, por serem exemplos em minha vida, tanto espiritual quanto profissional, agradeço por não medirem esforços para que eu alcançasse mais essa etapa em minha vida.

"If we knew what it was we were doing,
it would not be called research, would it?"
Albert Einstein

"Remember, when your wings are weak,
your spirits done and you've flown as far as you can,
you're halfway there!"
Legend of the Guardians: The Owls of Ga'Hoole

"Com a força que Cristo me dá,
posso enfrentar qualquer situação."
Filipenses 4:13 - NTLH

Resumo

A procura por armas tem crescido juntamente com os índices de criminalidade, sendo um problema contemporâneo que assombra diversos países. No Brasil, possíveis mudanças estão sendo discutidas para a flexibilização da posse e do porte de armas de fogo, dividindo opiniões e gerando uma enorme discussão sobre o assunto. Isto tem motivado cientistas a traçar soluções que possam auxiliar na segurança pública de maneira geral.

Em uma tentativa de buscar meios para minimizar este problema, foi realizada uma pesquisa sobre os principais trabalhos relacionados na classificação e detecção de armas de fogo, visando obter informações sobre as principais técnicas utilizadas. Com isso, neste trabalho, é proposta uma metodologia para a detecção de armas de fogo em imagens usando redes neurais convolucionais. Trabalhos recentes utilizaram detectores de objetos baseados em tais redes e apresentaram resultados relevantes. Por isso, neste trabalho foi proposta uma metodologia para detecção de armas utilizando um detector de objetos chamado *YOLO* (*You Only Look Once*) e uma arquitetura baseada em redes neurais convolucionais.

Duas abordagens foram realizadas para avaliar a metodologia proposta, levando em consideração três valores de limiar para IoU. A primeira abordagem, comparada com os resultados encontrados na literatura, aponta uma melhora nos resultados, onde foi alcançada uma precisão de $93,67\%$ e um F_1 de $93,05\%$, o que representa um crescimento superior a 10% na precisão e uma ligeira melhora de quase 2% na métrica F_1. A segunda abordagem segue a mesma metodologia, mas aplica uma etapa inicial diferente, onde o detector de objetos é modificado e utilizado para marcar um banco de dados e compor um novo conjunto de dados rotulado. Tal abordagem impactou positivamente nos resultados, onde houve um aumento na precisão e de quase 4% na métrica F_1. Nos três valores de IoU avaliados, o melhor apresenta uma precisão de $89,91\%$ e, a mesma configuração, aponta um F_1 de $94,54\%$ com uma confiança de 58%. Estes resultados mostram que a metodologia proposta é promissora para ser aplicada na detecção de armas de fogo em imagens.

Palavras-chave: Redes Neurais Convolucionais, Visão Computacional, Detecção de Objetos, YOLO.

Abstract

The demand for weapons has grown along with crime rates, being a contemporary problem that haunts several countries. In Brazil, possible changes are being discussed to make the ownership and possession of firearms more flexible, dividing opinions and generating a huge discussion on the subject. This has motivated scientists to devise solutions that can assist in public security in general.

In an attempt to find ways to minimize this problem, a research was carried out on the main work related to the classification and detection of firearms, aiming to obtain information on the main techniques used. Thus, in this work is proposed a methodology for the detection of firearms in images using convolutional neural networks. Recent work has used object detectors based on these networks and presented relevant results. Therefore, this work proposes a methodology for detecting weapons using an object detector, called YOLO (You Only Look Once), and an architecture based on convolutional neural networks.

Two approaches were taken to evaluate the proposed methodology, taking into account three threshold values for IoU. The first approach, compared with the results found in the literature, points to an improvement in the results, where an accuracy of $93,67\%$ and a F_1 of $93,05\%$ was achieved, which represents a growth greater than 10% in accuracy and a slight improvement of almost 2% in the F_1 metric. The second approach follows the same methodology, but applies a different initial step, where the object detector is modified and used to mark a database and compose a new labeled one. Such approach had a positive impact on the results, where there was an increase in accuracy and almost 4% in the F_1 metric. In the three IoU values evaluated, the best one has an accuracy of $89,91\%$ and, in the same configuration, points to a F_1 of $94,54\%$ with a confidence of 58%. These results show that the proposed methodology is promising to be applied for firearms detection in images.

Keywords: Convolutional Neural Network, Computer Vision, Object Detection, YOLO.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AUC	*Area Under The Curve*
FN	*Falsos Negativos*
FP	*Falsos Positivos*
FPS	*Frame Per Second*
FREAK	*Fast Retina Keypoint*
GPU	*Graphics Processing Unit*
HOG	*Histogram of Oriented Gradients*
IoU	*Intersection over Union*
IPEA	*Instituto de Pesquisa Econômica Aplicada*
PCA	*Principal Component Analysis*
PDI	*Processamento Digital de Imagens*
PMMW	*Passive Millimeter Wave Imaging*
RCNN	*Region-based Convolutional Neural Network*
ROC	*Receiver Operating Characteristic*
RPN	*Region Proposal Network*
SIFT	*Scale Invariant Feature Transform*
SURF	*Speeded Up Robust Features*
SVM	*Support Vector Machine*
TFP	*Taxa de Falsos Positivos*
TVP	*Taxa de Verdadeiros Positivos*
VN	*Verdadeiros Negativos*
VP	*Verdadeiros Positivos*
YOLO	*You Only Look Once*

Sumário

1 Introdução

No início do século XXI, o poder legislativo aprovou uma Lei que marcou o Brasil e dividiu opiniões. Em uma tentativa de controlar o crescimento acentuado dos índices de criminalidade, o poder legislativo aprovou, no final do ano de 2003, o Estatuto do Desarmamento (Doravante: ED) (CERQUEIRA, 2014). Responsável por definir regras e estabelecer diretrizes para o controle de armas no Brasil, a Lei nº 10.826/2003 trouxe uma série de critérios rigorosos que diz respeito ao acesso a armas, além de proibir o porte civil (BRASIL, 2003).

Segundo o Instituto de Pesquisa Econômica Aplicada - IPEA, a década de 90 apresentou um crescimento acentuado nos índices de homicídios e um aumento alarmante no índice de mortes por armas de fogo. No ano de 1990, a taxa de homicídios por cem mil habitantes era de $22,22$. Um pouco mais de 51% desses homicídios se davam pelas armas de fogo, chegando a uma taxa de $11,51$ por cem mil habitantes. No ano de 2003, ano da aprovação do ED, a taxa de homicídios era de $29,14$ por cem mil habitantes, $70,1\%$ desses homicídios se davam por armas de fogo, chegando a uma taxa de $20,42$ por cem mil habitantes. Entre 1990 e 2003, a variação na taxa de homicídios subiu 61% e a variação na taxa de homicídios por armas de fogo subiu $117,7\%$.

Os dois primeiros anos após a aprovação, foram marcados com uma redução pouco significativa dos índices de homicídios. Tais taxas caíram cerca de 10%, chegando ao número de $26,13$ homicídios por cem mil habitantes no ano de 2005, e cerca de 70% desses homicídios ainda se davam pelas armas de fogo, chegando a uma taxa de $18,14$ por cem mil habitantes (CERQUEIRA et al., 2020). E ainda, de acordo com o IPEA, entre 2003 e 2018, o ED apresentou pouca efetividade em relação ao percentual de homicídios por armas de fogo pelo total de homicídios, onde se manteve estável, com uma média de $70,4\%$.

Em 2018, a Lei completou 15 anos que estava em vigor e, até então, nenhuma mudança significativa havia ocorrido. Contudo, decretos assinados pelo presidente, Jair Messias Bolsonaro, no ano de 2019 e 2021, modificaram e flexibilizaram a posse e o porte de armas para civis no Brasil (SORANO et al., 2019; MAZZA, 2021). Dentre as exigências previstas pela Lei, pode-se citar, curso para manejar armas, ter 25 anos ou mais, não ter antecedentes criminais e comprovar a efetiva necessidade perante a Polícia Federal (BRASIL, 2003). Atualmente, existem alguns decretos que modificaram o ED, flexibilizando o registro e o porte de armas para civis. Além disso, os decretos aumentaram o limite de armas por cidadão, passando de quatro para seis armas de fogo, ampliação da validade do porte de armas para todo o território nacional, aumento na quantidade de cartuchos por ano entre outras flexibilizações para o porte de armas para civis (BRASIL, 2019; BRASIL,

2021a; BRASIL, 2021b; BRASIL, 2021c; BRASIL, 2021d).

De acordo com o último fechamento apresentado pelo IPEA, por meio do Atlas da Violência (CERQUEIRA et al., 2020), houve uma redução das taxas de homicídios e das taxas de homicídios por armas de fogo. Em 2019, após a flexibilização, o número de homicídios caiu $21,5\%$ e o número de homicídios por armas de fogo caiu $26,6\%$, implicando diretamente no percentual de homicídios por armas de fogo pelo total de homicídios, na qual, apresentou uma queda de $6,6\%$, chegando a $66,4\%$. Nesse ano (2019) a taxa por cem mil habitantes de homicídios foi de $21,65$ e a taxa por cem mil habitantes de homicídios por armas de fogo foi de $14,67$.

Pode-se observar que o ED impediu o crescimento acentuado dos índices de criminalidade, mantendo o percentual de mortes por armas de fogo estagnado, e a flexibilização, por sua vez, apresentou uma redução dos índices de criminalidade. Ambas as abordagens atuam com o objetivo de reduzir o percentual de mortes por armas de fogo e na prevenção de crimes utilizando armas, como exemplo, os massacres ocorridos no ano de 2019 (G1, 2019a; G1, 2019b).

Os dados supracitados reforçam a necessidade de propor soluções que possam contribuir na segurança pública. Um caminho para auxiliar no combate desse problema é a detecção automática de armas de fogo por meio de imagens. Recentemente, com o advento das redes neurais profundas, em especial das redes convolucionais profundas, a identificação de objetos em cenas tem passado por relevantes avanços, sendo possível até realizar a segmentação de objetos em nível de pixel com uma razoável precisão (HE et al., 2017). Várias tarefas relacionadas a visão computacional estão sendo realizadas com alto grau de sucesso graças aos avanços nos últimos anos. Sendo assim, dada a sua relevância, este trabalho tem como tema a identificação de armas em imagens a partir da utilização de redes neurais convolucionais profundas.

1.1 Método Proposto

O presente trabalho propõe utilizar técnicas computacionais que envolvam redes neurais convolucionais profundas com o intuito de detectar armas de fogo em imagens. A partir do uso do detector de objetos *YOLOv2* (*You Only Look Once*) (REDMON; FARHADI, 2017) e utilizando as bases de dados criadas e disponibilizadas por Olmos et al. (2018), uma metodologia foi elaborada e avaliada, visando a detecção de armas de fogo em imagens. A escolha do detector *YOLO* foi motivada pela sua velocidade de detecção, que é superior a de outros métodos da literatura, apresentando resultados compatíveis aos métodos de estado da arte.

O método será avaliado através de duas abordagens, sendo que em uma delas pretende-se utilizar o detector de objetos (*YOLOv2*) para auxiliar na marcação automática

de imagens sem marcação (informações referentes à localização de objetos contidos nas imagens, conhecidos como *bounding boxes*). Tal abordagem será realizada com modificações no código fonte do detector, uma vantagem do *open source*. Portanto, deve-se utilizar tal abordagem para compor um novo conjunto de dados rotulado, o qual será utilizado para treinar, testar e comparar o resultado de detecção de armas de fogo em imagens com os resultados presentes na literatura.

1.2 Objetivos

O principal objetivo deste trabalho é desenvolver um método computacional que envolva redes neurais convolucionais profundas para detecção de armas de fogo em imagens, levando em consideração a confiabilidade e a velocidade de detecção.

Os objetivos específicos do trabalho são:

- Propor uma metodologia de detecção de armas de fogo em imagens utilizando o detector de objetos *YOLO*;
- Propor uma metodologia de marcação do conjunto de dados para aprimorar os resultados da proposta;
- Comparar os resultados das metodologias propostas com os resultados encontrados na literatura.

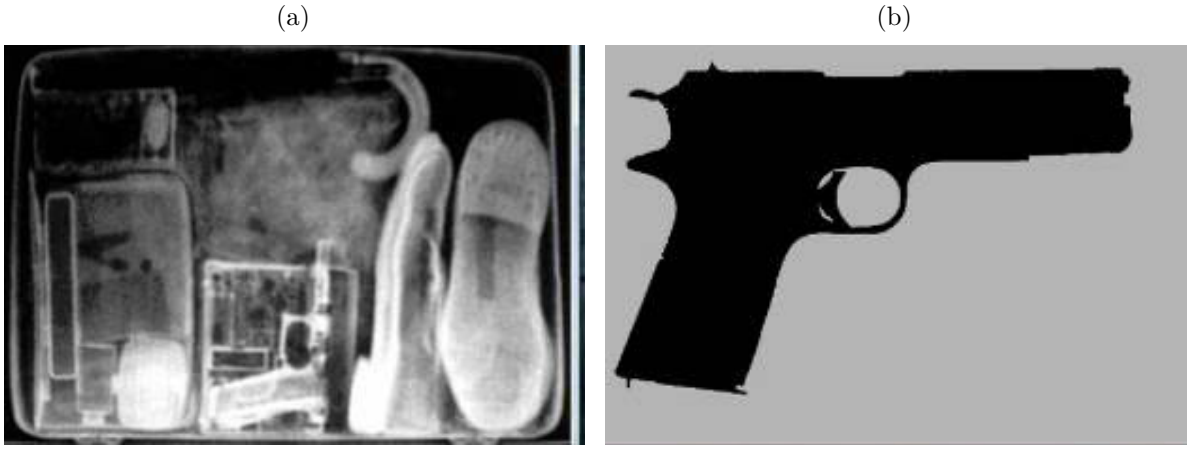
1.3 Estrutura da dissertação

Este trabalho possui a seguinte estrutura. No Capítulo 2 é apresentado um embasamento teórico, mostrando um histórico na detecção de armas de fogo. No Capítulo 3 é apresentado um breve histórico da detecção de objetos, mostrando os principais marcos e a evolução dos algoritmos de detecção de objetos e, ainda, apresenta uma descrição e explicação do detector de objetos utilizado: *YOLO*. No Capítulo 4 é explicada a metodologia utilizada para detecção de armas em imagens, onde são descritos os conjuntos de dados, as configurações utilizadas no detector de objetos e as métricas de avaliação utilizadas nas abordagens. O Capítulo 5 exibe e discute os resultados obtidos nas abordagens realizadas para a tarefa de detecção de armas de fogo em imagens. Finalmente, o Capítulo 6 consiste em uma conclusão e caminhos para trabalhos futuros.

2 Detecção de Armas de Fogo: Do Cinza ao Colorido

Os primeiros trabalhos na literatura a apresentar abordagens para detectar armas de fogo utilizavam em suas metodologias imagens em nível de cinza provenientes de ondas de raios-x. Gesick et al. (2009) avaliaram três abordagens separadas envolvendo *pattern matching*, transformada *Daubechies Wavelet* (DAUBECHIES, 1992), e uma abordagem utilizando *SIFT* (*Scale Invariant Feature Transform*) (LOWE, 2004). Os autores tinham como objetivo detectar armas escondidas em bagagens de aeroportos (Figura 1a) sem a intervenção humana, considerando a acurácia e o tempo de detecção.

Figura 1 – Exemplos do conjunto de dados de Gesick et al. (2009).



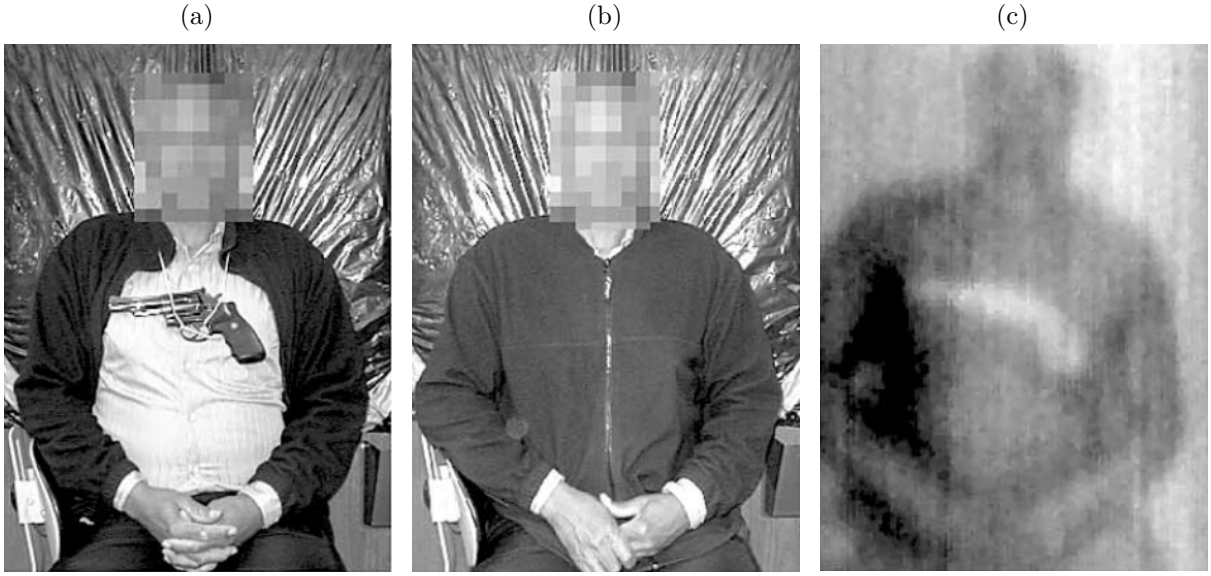
Fonte: Adaptado pelo Autor (GESICK et al., 2009).

Com um conjunto de dados pequeno, fornecida por empresas da área de segurança em aeroportos, Gesick et al. (2009) assumiram algumas condições em seu conjunto de dados. Imagens que continham armas estavam limitadas a rotações no plano horizontal e foram consideradas apenas armas do tipo "pistola" no processo de treinamento e validação. De todas as abordagens, o *pattern matching* apresentou os melhores resultados, onde detectou corretamente 10 imagens das 12 imagens de teste. Porém, para uma imagem padronizada de 1000×800 *pixels*, a abordagem levou em média 6,5 segundos para imagens consideradas "fáceis" (Figura 1b) pelos autores e 165 segundos para imagens mais "complexas" (Figura 1a), inviabilizando tais abordagens para a utilização em tempo real. Em contrapartida, a abordagem proposta foi um avanço para a detecção automática de armas, onde outrora era dependente de intervenção humana.

Xiao et al. (2015) propuseram uma metodologia para detecção automática de armas

escondidas utilizando *Passive Millimeter Wave Imaging - PMMW* - imagens geradas através de sensores passivos de radiação de ondas milimétricas que ocorrem naturalmente em uma cena, possibilitando a geração de imagens de armas ocultas de maneira não invasiva (Figura 2) (YUJIRI et al., 2003).

Figura 2 – Exemplos de (2a) uma pessoa portando uma arma, (2b) uma pessoa escondendo uma arma com uma jaqueta e (2c) uma imagem gerada de (2b) com PMMW.



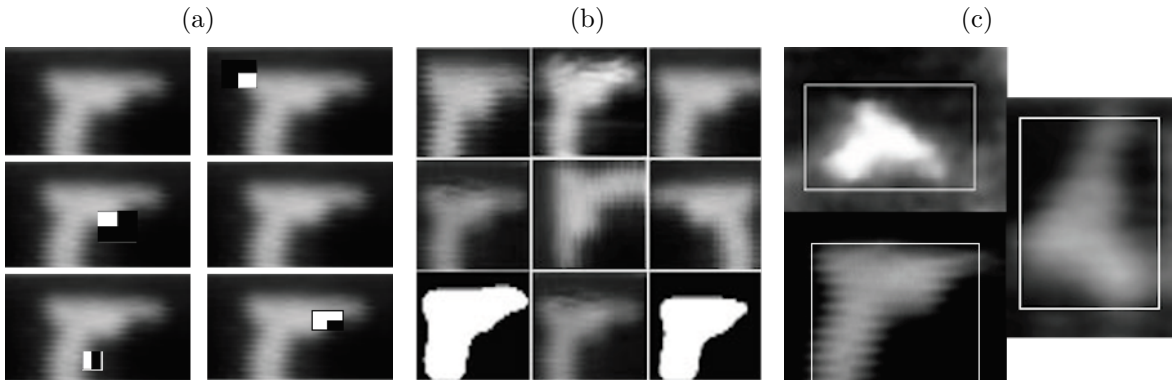
Fonte: Adaptado pelo Autor (YUJIRI et al., 2003).

Com uma metodologia não trivial e um pouco extensa, Xiao et al. (2015) utilizam métodos de reconhecimento de objetos e de aprendizado de máquina, utilizando o extrator de características *Haar-Like* e o classificador *AdaBoost* em cascata. Xiao et al. (2015) não apresentam dados quantitativos em relação ao seu conjunto de dados e sobre os resultados, mas apresentam algumas imagens referentes à metodologia proposta (Figura 3a, 3b) e de alguns resultados (Figura 3c). Eles ainda concluíram que a metodologia pode ser utilizada para detectar armas metálicas automaticamente, mas que estudos na área de PMMW são necessários para aumentar as possibilidades dessa metodologia.

Os trabalhos a seguir diferem dos objetivos apresentados até aqui, que eram buscar formas de detectar armas de fogo escondidas através de imagens de raios-X e PMMW. Os próximos trabalhos discutidos buscam auxiliar sistemas de vídeo monitoramento na detecção automática de armas, onde a arma está aparente e não mais escondida. Vale lembrar que as metodologias a seguir se baseiam em imagens RGB.

Alguns trabalhos utilizam métodos clássicos de processamento de imagem para detectar armas de fogo em imagens RGB. Tiwari e Verma (2015a) desenvolveram um *framework* que é dividido em duas etapas: a primeira consiste em segmentar uma imagem baseada em suas cores, utilizando *K-means* (Figura 4a, 4b), em seguida é utilizado o

Figura 3 – Imagens referentes à metodologia e os resultados de (XIAO et al., 2015).



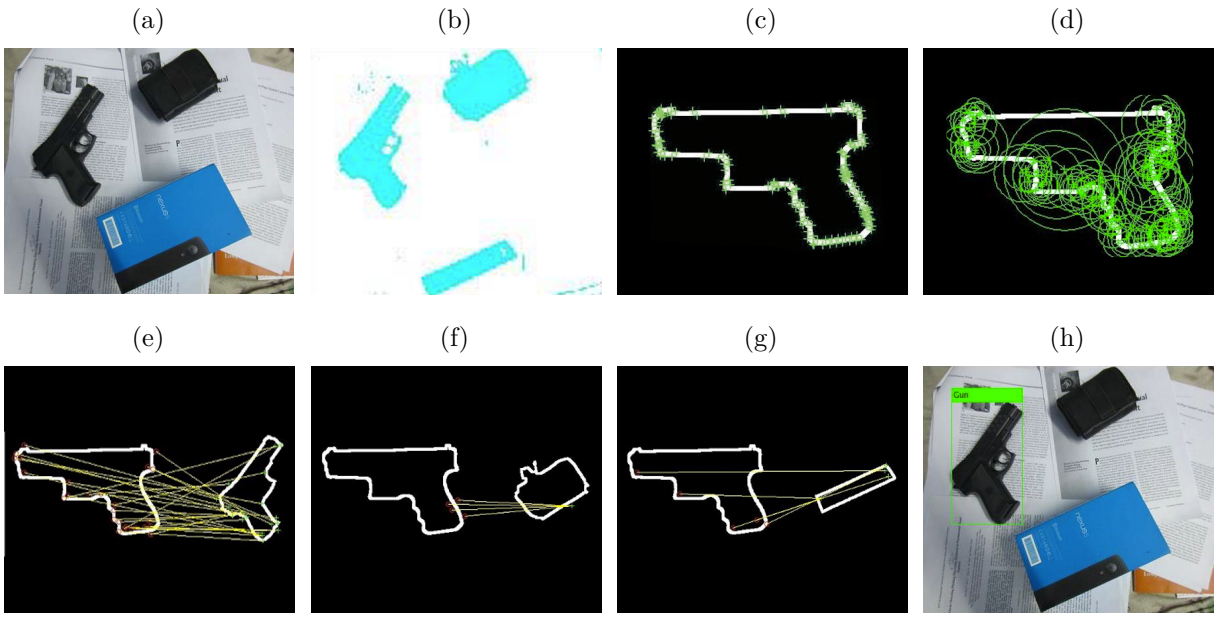
Fonte: Adaptado pelo Autor (XIAO et al., 2015).

detector de cantos *Harris* (HARRIS et al., 1988) e um extrator de características - *Fast Retina Keypoint - FREAK* (Figura 4c) (ALAHI et al., 2012). Tiwari e Verma (2015b) desenvolveram outro *framework* similar, dividido em duas etapas, sendo a primeira etapa idêntica ao trabalho anterior (TIWARI; VERMA, 2015a), e na segunda etapa foi utilizado um extrator de características mais rápido e robusto - *Speeded Up Robust Features - SURF* (Figura 4d) (BAY et al., 2006). Em ambos os *frameworks*, a segunda etapa foi responsável por determinar os pontos de interesse e, em seguida, era verificada a similaridade entre uma pistola e a imagem verificada, como mostra as Figuras 4e-4h.

O primeiro trabalho, (TIWARI; VERMA, 2015a), obteve uma acurácia de $84,26\%$, utilizando um conjunto de dados criado pelos autores, contendo 65 imagens com armas e 24 imagens sem armas, com resoluções iguais a 400×300 *pixels*. O segundo trabalho, (TIWARI; VERMA, 2015b), obteve uma acurácia de $88,67\%$, utilizando um conjunto de dados inferior, contendo 15 imagens com armas e 10 imagens sem armas, com resoluções iguais a 400×300 *pixels*. Ambos os trabalhos obtiveram uma alta acurácia, mas não apresentam informações sobre o tempo de detecção, e afirmam que seus *frameworks* não apresentam o melhor desempenho possível para processamento em tempo real, mas que são capazes de detectar mais de uma arma em uma imagem automaticamente.

Halima e Hosam (2016) propuseram um classificador de imagens para sistemas de vigilância utilizando técnicas combinadas de aprendizado de máquina e processamento de imagem. Em uma metodologia dividida em 4 etapas, foi utilizado *SIFT* para extrair pontos de interesse das imagens, *K-means* para agrupar estes pontos, um algoritmo para calcular um histograma espacial baseado nos *clusters* encontrados e, por fim, as informações obtidas nos histogramas espaciais foram utilizadas como entrada para o treinamento do classificador *SVM* (*Support Vector Machine*). Halima e Hosam (2016) não quantizaram ao certo o conjunto de dados usado, mas mencionaram que era uma conjunto de dados grande, que continha imagens retiradas da *Internet* e de câmeras digitais, com imagens de

Figura 4 – Imagem de entrada (4a), imagem segmentada (4b) de 4a, pontos de interesse de uma imagem de controle utilizando *Harris* e *FREAK* (4c), pontos de interesse de uma imagem de controle utilizando *SURF* (4d), verificação de similaridade com uma imagem de controle (4e-4g) e uma imagem com uma arma detectada (4h).



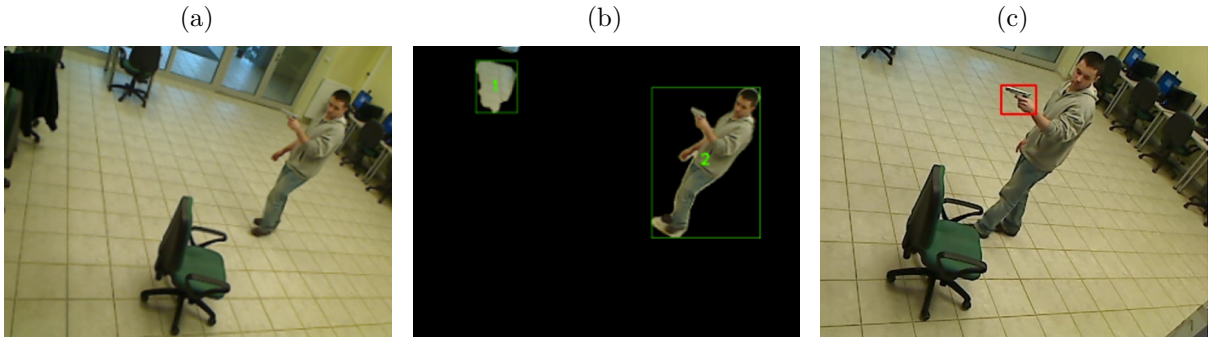
Fonte: Adaptado pelo Autor (TIWARI; VERMA, 2015a; TIWARI; VERMA, 2015b).

pistolas, rifles e imagens que não continham nenhum tipo de arma (amostras negativas para o treinamento do classificador). Este trabalho não apresenta nenhuma métrica de avaliação dos seus resultados, mas conclui que seu trabalho necessita de melhorias para ser utilizado em aplicações em tempo real, levando em consideração um classificador mais eficiente e uma metodologia que não envolva cálculos de *clusters* e histogramas.

O trabalho de Grega et al. (2013) propõe um reconhecimento automatizado de armas em sistemas de vídeo monitoramento, utilizando processamento digital de imagens e redes neurais. Sua metodologia começa com um *background detection* (Figura 5b) responsável por detectar movimento na cena, onde o algoritmo subtrai a cena atual com a anterior e, diante de alguns processos morfológicos e um determinado limiar, identifica os novos objetos presentes na cena. Ocorre uma varredura para localizar estes objetos, que são redimensionados e transformados em vetores, sendo reduzido novamente com a utilização do PCA (*Principal Component Analysis*). A saída do PCA é a entrada de uma rede neural artificial com 560 neurônios na camada de entrada, 200 neurônios na camada oculta e 9 neurônios na camada de saída. Um descritor de imagem é utilizado para reduzir os falsos positivos, comparando os candidatos provenientes da rede neural com uma arma de controle. Um verdadeiro positivo pode ser observado na Figura 5c.

A metodologia do trabalho de Grega et al. (2013) parte do princípio que as câmeras

Figura 5 – Imagem de entrada (5a), *background detection* (5b) e um exemplo de verdadeiro positivo (5c).



Fonte: Adaptado pelo Autor (GREGA et al., 2013).

de vídeo monitoramento são estáticas, de tal forma que foi possível construir um conjunto de dados específico. Com vídeos de baixa qualidade e resolução igual a 640×480 *pixels*, os autores simularam câmeras de vídeo monitoramento e utilizaram duas métricas para avaliar sua proposta: sensibilidade (taxa de verdadeiros positivos) e especificidade (taxa de verdadeiros negativos). A metodologia proposta resultou em uma sensibilidade de $35,98\%$ e uma especificidade de $96,69\%$, sendo avaliadas em 4425 *frames*, correspondente a um vídeo contendo armas. Grega et al. (2013) apresentam resultados expressivos em suas métricas, mas não informam o tempo de detecção de sua proposta, assim como não disponibilizam, de maneira direta, seu conjunto de dados. Grega et al. (2013) concluíram que sua proposta necessitava de otimizações e consideraram substituir sua rede neural de três camadas por uma CNN em trabalhos futuros.

Em um trabalho posterior, Grega et al. (2016) apresentaram uma metodologia para duas abordagens - detecção de armas e facas em sistemas de vídeo monitoramento - na qual a abordagem e os resultados para a detecção de armas não apresentaram nenhuma modificação proposta por Grega et al. (2013), apenas um diferencial, os autores divulgaram e compartilharam o conjunto de dados completo, embora a descrição apresentada não condissesse com o conjunto de dados compartilhado.

Olmos et al. (2018) propuseram um "alarme de detecção automático de armas em vídeos usando *deep learning*", que, segundo os autores e confirmado pela literatura, é o primeiro trabalho que utiliza CNN para detecção de armas de fogo em imagens. Os autores avaliaram duas abordagens antes de propor o sistema de detecção automática, o *sliding window* e o *region proposal*. Foram criados 6 bancos de dados, 4 para avaliar o *sliding windows*, 1 para o *region proposals* e 1 banco de teste para avaliar as duas abordagens. As métricas utilizadas pelos autores para avaliar as duas abordagens foram: o tempo de detecção, a precisão, a sensibilidade e o *F1 Score*.

O *sliding windows* não apresentou resultados relevantes, além de o processo de

detecção demorar $1,5$ segundos para imagens de resolução igual a 640×360 *pixels*. Por outro lado, a *region proposals* demora $0,19$ segundos no processo de detecção em imagens de resolução igual a 1000×1000 *pixels*, além de apresentar uma precisão de $84,21\%$, sensibilidade de 100% e *F1* igual a $91,43\%$. Vale lembrar que os autores utilizaram o *Faster R-CNN* (REN et al., 2015), pois este modelo implementa o *Region Proposal Network - RPN*. Os autores utilizaram esta abordagem para avaliar as detecções de armas em alguns vídeos retirados da *Internet* e definiram uma nova métrica. Os autores ainda compartilharam 3 das 6 bases de dados criadas em um repositório público[1]: a base de teste; uma base referente a abordagem *sliding windows*; e a base utilizada para o *region proposal* com as suas respectivas anotações.

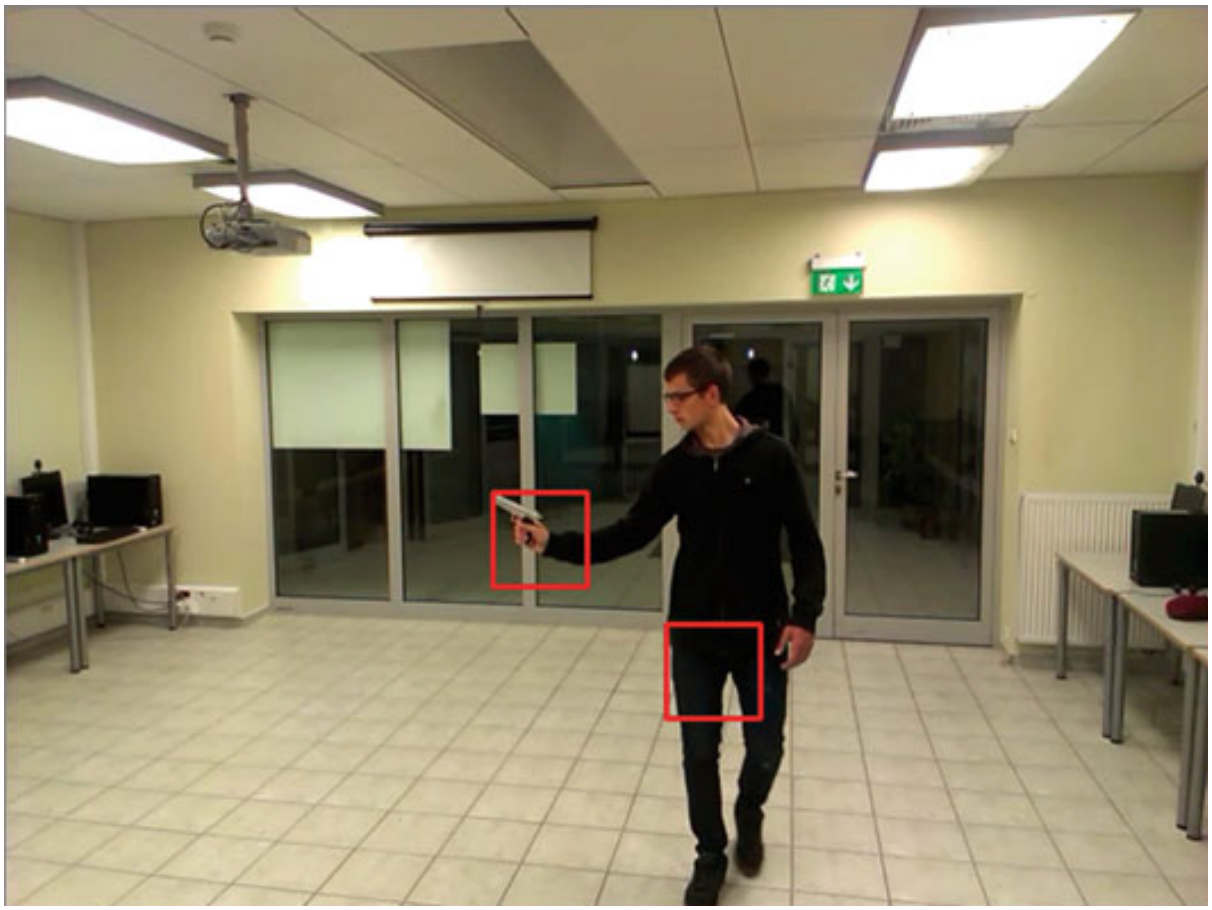
Baseado no trabalho de Olmos et al. (2018), diferentes trabalhos começaram a utilizar CNN em seus trabalhos, por exemplo, Gelana e Yadav (2019) propuseram uma alteração na metodologia de Grega et al. (2013), onde optaram por trabalhar com um método para detecção de armas em vídeos envolvendo um detector de movimento e classificação com uma CNN. Com um total de 6 passos, a proposta de Gelana e Yadav (2019) envolve um passo para converter os *frames* do vídeo em imagens, outro passo envolvendo um conversor de imagens RGB para imagens em níveis de cinza, outro passo utilizando uma subtração de fundo para ativar e verificar mudanças na cena, ou seja, um detector de movimento, e algumas operações de filtragem e segmentação. Além disso, seguindo as propostas de trabalhos futuros de Grega et al. (2013), uma CNN foi adicionada no processo de classificação ao invés de uma simples rede neural, baseado nos resultados satisfatórios apresentados por Olmos et al. (2018). Gelana e Yadav (2019) utilizaram um conjunto de dados criado por Grega et al. (2016) (a base é formada por 7 vídeos), que é composta por 4000 amostras positivas e 1869 amostras negativas, sendo dividida em 70% para treinamento e 30% para teste, aproximadamente. Essa abordagem apresentou uma acurácia de $97,78\%$, uma sensibilidade de $93,84\%$ e uma especificidade de $99,73\%$, resultados promissores e superiores aos trabalhos de Grega et al. (2013). Entretanto, não foi realizada uma comparação direta, uma vez que Grega et al. (2013) e Gelana e Yadav (2019) não especificaram quais dos 7 vídeos foram avaliados.

Embora a metodologia proposta por Gelana e Yadav (2019) seja capaz de detectar armas em imagens, sua metodologia é avaliada por uma classificação binária, sendo suficiente apenas uma detecção positiva de uma arma (verdadeiro positivo) para considerar uma classificação correta, como pode ser observado na Figura 6.

Elmir et al. (2019) propuseram outro alarme automático de armas, dividido em duas etapas: um detector de movimento e um detector de armas. O detector de movimento apresenta uma metodologia similar ao apresentado por Gelana e Yadav (2019), onde os autores utilizaram um conversor de imagens RGB para imagens em níveis de cinza, somados

[1] https://github.com/SihamTabik/Pistol-Detection-in-Videos

Figura 6 – Exemplo de uma classificação positiva para armas contendo duas detecções, um verdadeiro positivo e um falso positivo.



Fonte: Adaptado pelo Autor (GELANA; YADAV, 2019).

com outras técnicas de processamento digital de imagens. Elmir et al. (2019) avaliaram dois detectores de objetos em sua metodologia, o *Fast R-CNN* e o *MobileNet-CNN*, utilizando as bases de dados criadas por Olmos et al. (2018). Um dos objetivos dos autores era a detecção em tempo real, onde os autores afirmam que a metodologia proposta apresenta um desempenho aceitável em tempo real, porém, sem apresentar dados quantitativos referentes ao tempo de detecção. Os detectores avaliados não apresentaram resultados satisfatórios, comparado aos trabalhos de Olmos et al. (2018), mas contribuem de maneira positiva na avaliação de detectores de objetos em um mesmo conjunto de dados. O modelo que apresentou melhores resultados foi o *Fast R-CNN*.

Diante dos trabalhos apresentados, um dos caminhos para a evolução na detecção de armas em imagens é a utilização de redes neurais profundas convolucionais. Avanços apresentados por Olmos et al. (2018), e trabalhos subsequentes utilizando metodologias diferentes, mas com a mesma técnica (CNN), apresentaram grandes avanços e possibilidades para melhoria (OLMOS et al., 2018; GELANA; YADAV, 2019; ELMIR et al., 2019). Uma possibilidade abordada no presente trabalho é a utilização de um detector de objetos

mais robusto e veloz, o *YOLOv2* (REDMON; FARHADI, 2017). Olmos et al. (2018) foi o primeiro a utilizar CNN, além disso, a disponibilização das bases de dados, contribui de forma positiva para a reprodutibilidade de trabalhos futuros.

O objetivo deste trabalho é desenvolver um método computacional que envolva redes neurais convolucionais profundas para detecção de armas de fogo em imagens, levando em consideração a confiabilidade e a velocidade de detecção. Dentre os trabalhos supracitados, Olmos et al. (2018) e Elmir et al. (2019) apresentam semelhanças com o objetivo proposto, além disso, utilizam em seus experimentos a mesma fonte de dados. Logo, o presente trabalho visa a utilização do detector de objetos *YOLOv2* juntamente com as bases de dados de Olmos et al. (2018) para avaliar a metodologia proposta para com os trabalhos citados.

3 Detecção de Objetos

Neste capítulo será realizada uma breve revisão da tarefa de detecção de objetos, assim como, apresentar as técnicas e ferramentas utilizadas ao longo deste trabalho.

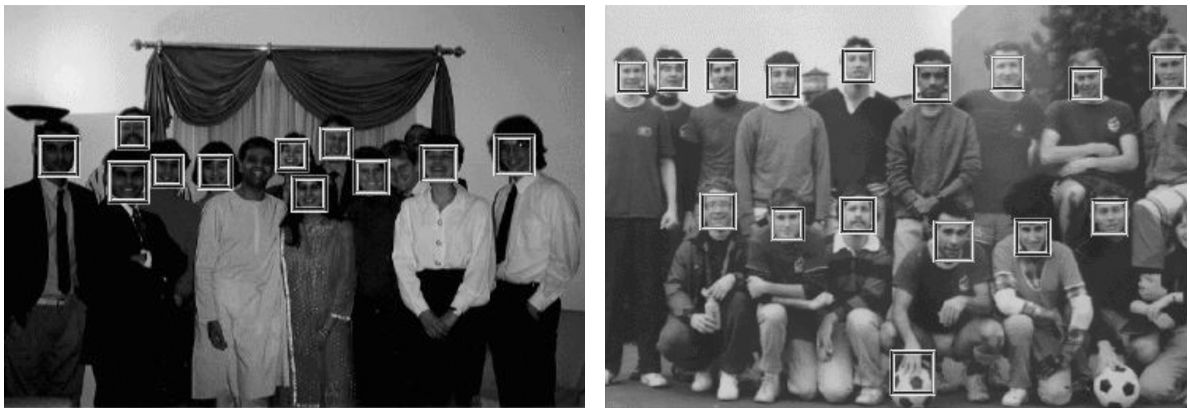
3.1 História da Detecção de Objetos

A década de 60 não apenas marcou a humanidade com a chegada do homem à Lua (LOFF, 2019), mas com avanços científicos importantes referentes ao processamento digital de imagens (PDI) (GONZALEZ; WOODS, 2009; MACHEMER, 2019). Segundo Gonzalez e Woods (2009), o nascimento de PDI ocorreu graças à disponibilidade de máquinas potentes (para a época) e a necessidade do tratamento das imagens obtidas por sondas espaciais.

Desde então, muitos avanços ocorreram, assim como no início do século XXI com o algoritmo *Viola-Jones* (VIOLA; JONES, 2001). Viola e Jones (2001) apresentaram o primeiro algoritmo prático para detecção de faces, um algoritmo simples e que poderia ser utilizado em tempo real, algo bastante notável para o *hardware* da época. O algoritmo é dividido em 4 etapas: o extrator de características *Haar-Like*; a imagem integral; o treinamento do classificador *AdaBoost*; e o classificador em cascata. Etapas simples, mas que permitiram um detector rápido e preciso. Contudo, mudanças na iluminação e ligeiras inclinações no rosto podem afetar as detecções. A Figura 7 apresenta alguns exemplos das detecções com o algoritmo *Viola-Jones*.

Figura 7 – Exemplos de detecções faciais utilizando o algoritmo *Viola-Jones*.

(a) (b)

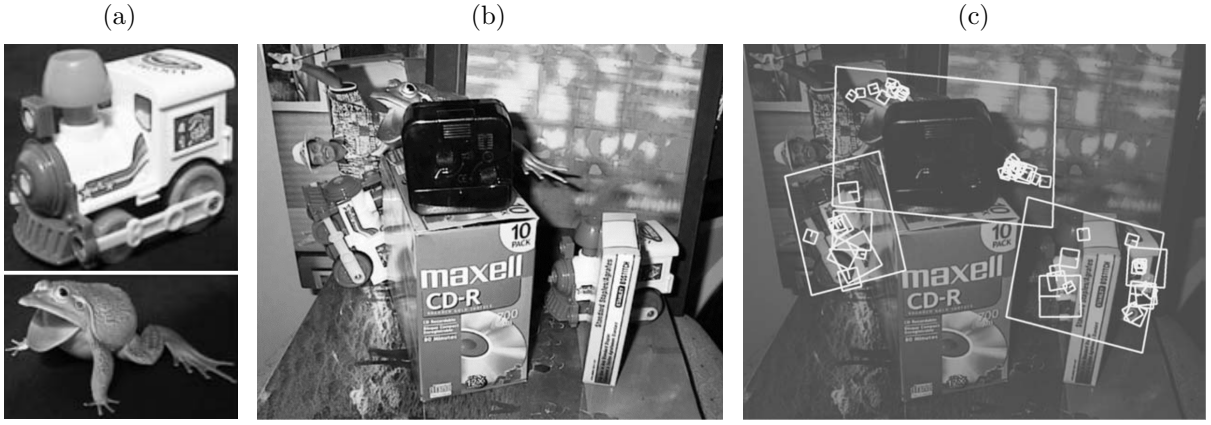


Fonte: Adaptado pelo Autor (VIOLA; JONES, 2001).

Alguns anos depois, Lowe (2004) apresentou um algoritmo para correspondência

de características chamado *SIFT* (*Scale Invariant Feature Transform*, em português, transformação de características invariante de escala). Basicamente, o algoritmo apresenta 4 etapas principais: a detecção extrema de espaço de escala; a localização de *keypoints* (pontos relevantes na imagem); atribuição de orientação; e a criação dos descritores dos *keypoints*. Em resumo, Lowe (2004) apresentou uma maneira de extrair *keypoints* de imagens e uma maneira de calcular seus descritores, na qual, são invariantes à translação, rotação, escala e outros parâmetros de imagem. Dessa forma, é possível combinar os *keypoints* entre duas imagens, podendo realizar a detecção de objetos em diferentes visualizações em uma cena e até mesmo com oclusão parcial e variação de iluminação. A Figura 8 apresenta um exemplo de detecção com oclusão parcial utilizando o *SIFT*. No entanto, o algoritmo proposto por Lowe (2004) apresenta problemas na detecção em imagens com padrões repetidos e imagens com pouca textura, além de não ser tão rápido para uso em tempo real. Existem algumas alternativas ao *SIFT*, como o *SURF* (*Speeded Up Robust Features* - recursos robustos acelerados) proposto por Bay et al. (2006), que é uma versão acelerada do *SIFT*. Bay et al. (2006) demonstram que ele é três vezes mais rápido que o *SIFT*, apresenta um desempenho comparável, mas com problemas para lidar com mudanças de ponto de vista e iluminação.

Figura 8 – Exemplos de detecções de objetos utilizando o *SIFT* com oclusão de imagem. Imagem com os objetos de interesse (8b), imagem de entrada (8c) e imagem com as detecções dos objetos (8c).

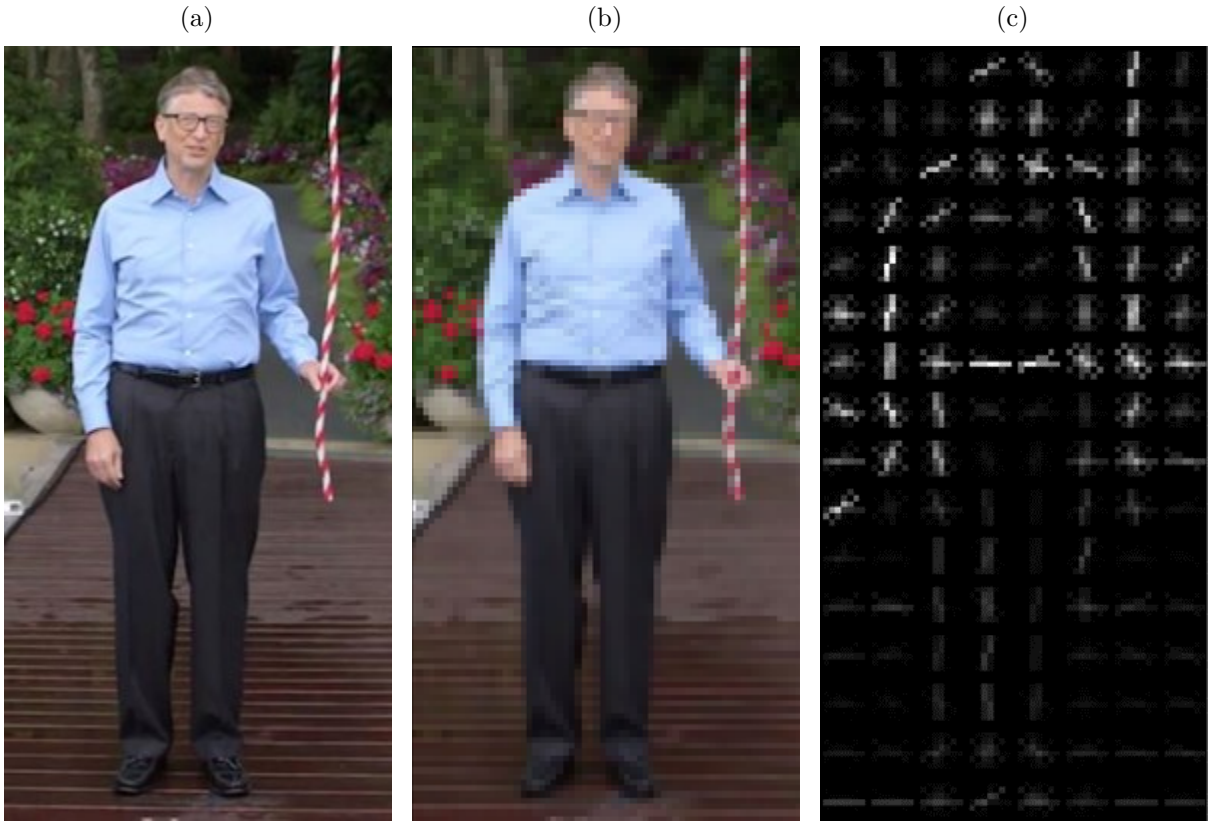


Fonte: Adaptado pelo Autor (LOWE, 2004).

Dalal e Triggs (2005) apresentaram um extrator de características denominado *HOG* (*Histogram of Oriented Gradients* - histograma de gradientes orientados) e foi bastante utilizado para detectar objetos e pessoas em imagens. *HOG* tem como base o trabalho de Lowe (2004), porém, diferente de *SIFT*, o descritor calcula os histogramas de gradientes em uma densa e sobreposta grade de células uniformemente espaçadas na imagem. Em resumo, a ideia básica por trás da utilização do descritor *HOG* é que a aparência e a forma de objetos podem ser caracterizadas pela distribuição da intensidade dos gradientes locais

ou as direções das bordas. O descritor divide a imagem em pequenas regiões espaciais, denominadas células, e acumulando, para cada célula, um histograma 1D das direções dos gradientes ou as orientações das bordas. A combinação destes histogramas forma a representação dos contornos e aparências locais da imagem. O descritor *HOG* é formado pela concatenação dos diversos blocos presentes em uma janela de detecção varrida sobre a imagem. A Figura 9 apresenta um exemplo do descritor *HOG*. Vale lembrar que Dalal e Triggs (2005) utilizaram os descritores *HOG* para treinar um classificador (*SVM*), sendo possível classificar humanos e não humanos, dado que o conjunto de dados utilizado para treino era de humanos.

Figura 9 – Exemplo do descritor *HOG*. (9a) Imagem de entrada, (9b) imagem pré-processada (imagem redimensionada para o tamanho 64×128 *pixels* (DALAL; TRIGGS, 2005)) e a (9c) imagem dos histogramas combinados.



Fonte: o Autor.

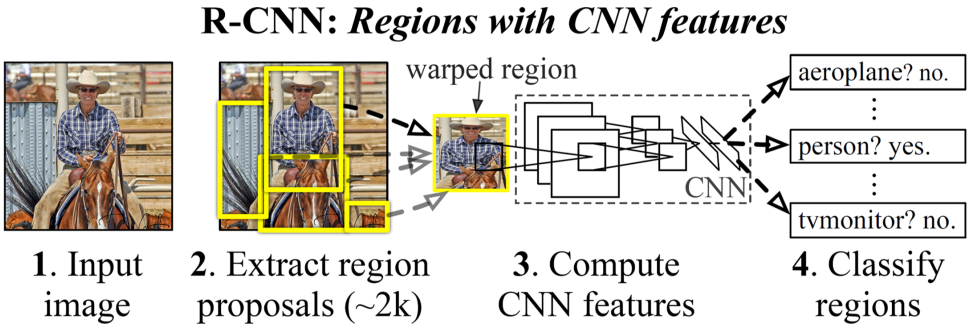
O ano de 2012 foi marcado com uma das contribuições mais relevantes do século XXI para a área de processamento digital de imagens e visão computacional. Neste ano, a famosa competição anual em tarefas de reconhecimento visual, chamada *ImageNet*[1] (DENG et al., 2009), foi surpreendida com o trabalho de Krizhevsky et al. (2012) ao utilizar redes neurais convolucionais e superar todos os outros[2] com a rede *AlexNet*: uma arquitetura

[1] *ImageNet: Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)*
[2] Resultados *ILSVRC2012* (O time de Krizhevsky et al. (2012) é intitulado como *SuperVision*)

considerada simples atualmente, pois apresenta apenas 5 camadas convolucionais e 3 camadas totalmente conectadas. Vale lembrar que as *CNN*, do inglês, *Convolutional Neural Networks* (Redes Neurais Convolucionais), existem desde a década de 80 (FUKUSHIMA; MIYAKE, 1982; LECUN et al., 1989), porém, o *hardware* da época inviabilizava sua utilização. Em uma comparação rápida, pode-se destacar que o processo de treinamento demorou de 5 a 6 dias para treinar a rede *AlexNet*, utilizando uma *GPU* (*Graphics Processing Unit*, em português, Unidade de Processamento Gráfico) de alta desempenho para a época (*GTX580*) (KRIZHEVSKY et al., 2012).

Desde a rede *AlexNet*, muitas outras arquiteturas utilizando *CNN* surgiram aplicadas no *ImageNet*, como exemplo a *VGGNet* (SIMONYAN; ZISSERMAN, 2014) e a *Inception* (SZEGEDY et al., 2015), apresentando arquiteturas robustas e complexas para a tarefa de classificação. Por sua vez, Girshick et al. (2014) apresentaram uma arquitetura denominada *R-CNN*, que delimitava objetos de interesse em uma imagem utilizando *CNN*. A ideia por trás das "redes neurais convolucionais baseadas em regiões" (do inglês, *Region-based Convolutional Neural Network - R-CNN*) é utilizar um processo de busca seletiva (UIJLINGS et al., 2013) para extrair candidatos, utilizar essas informações em uma *CNN* inspirada na *AlexNet*, a fim de extrair características e treinar um classificador *SVM*. A Figura 10 apresenta o sistema de detecção proposto por (GIRSHICK et al., 2014).

Figura 10 – Sistema de detecção de objetos do *R-CNN*.



Fonte: Adaptado pelo Autor (GIRSHICK et al., 2014).

A abordagem de Girshick et al. (2014) se mostrou eficaz na detecção de objetos. A rede *R-CNN* evoluiu para a *Fast R-CNN* (GIRSHICK, 2015), *Faster R-CNN* (REN et al., 2015) e para o *Mask R-CNN* (HE et al., 2017). No entanto, as abordagens supracitadas, utilizando como base a arquitetura *R-CNN*, tem como um primeiro passo gerar as possíveis *bounding boxes*, em seguida, executar o processo de extração de características e o processo de classificação. Por fim, um processo de pós-processamento é executado para melhorar o encaixe das *bounding boxes* e eliminar detecções duplicadas. Estes *pipelines* complexos são lentos e difíceis de otimizar, pois cada componente individual deve ser treinado separadamente (REDMON et al., 2016).

Estes métodos examinam cada imagem milhares ou centenas de milhares de vezes para realizar a detecção, isso envolve muitas avaliações desses classificadores repetidamente em diferentes partes da imagem e em diferentes escalas. Porém, um detector de objetos conhecido como *YOLO* (*You Only Look Once*), adotou uma abordagem completamente diferente e, em alguns casos, superou os métodos supracitados.

3.2 Uma visão geral do *YOLO*

YOLO é um dos detectores de objetos mais populares e mais recentes para detecção em tempo real. Atualmente, existem 5 versões do detector, 4 versões oficiais (REDMON et al., 2016; REDMON; FARHADI, 2017; REDMON; FARHADI, 2018; BOCHKOVSKIY et al., 2020) e uma *branch* não oficial do detector de objetos (JOCHER et al., 2021). Nesta seção é apresentado como este detector de objetos funciona e suas principais vantagens, com foco na versão 2, utilizada neste trabalho (que era a versão disponível no momento em que a maior parte dos experimentos foram realizados).

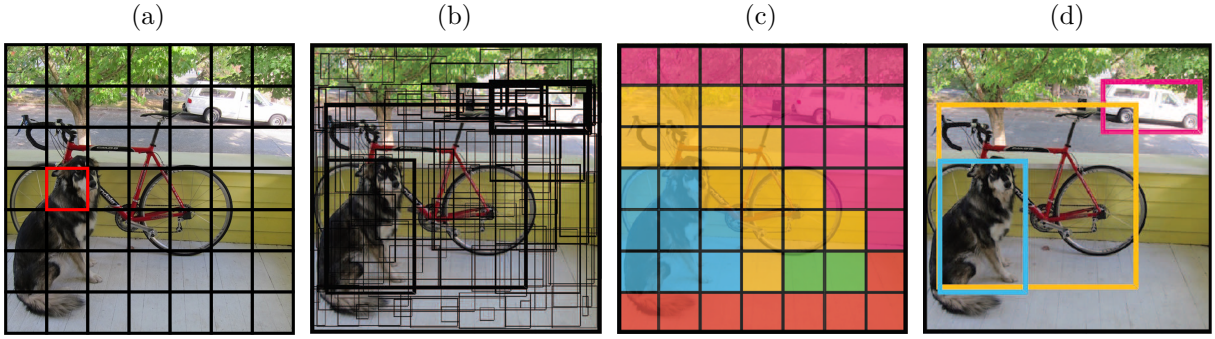
3.2.1 Funcionamento

O detector de objetos *YOLO*, utiliza uma única *CNN* para detectar os objetos e apresentar o valor de confiança das classes detectadas, dessa forma, não sendo necessário recorrência ou redes paralelas, impactando diretamente na velocidade do detector. Dada uma imagem de entrada, o *YOLO* divide a imagem em uma grade de tamanho $S \times S$, onde cada subdivisão da imagem é chamada de célula (Figura 11a). O termo S representa a "sensibilidade" do detector, onde quanto maior o valor de S, mais sensível será o detector de objetos, em contrapartida, valores elevados impactam diretamente na velocidade de detecção.

Cada célula é responsável por prever a localização das B *bounding boxes*, o valor de confiança de cada detecção e uma probabilidade de cada um das C classes condicionada à existência de um objeto na célula. Se o centro de um objeto cai em uma célula, essa célula é "responsável" por detectar a existência desse objeto. A Figura 11b apresenta uma representação gráfica das B *bounding boxes* e o valor de confiança (ilustrado pela espessura da marcação). A Figura 11c apresenta um mapa de probabilidades das C classes (cada célula é responsável por detectar uma classe), e por fim, a pontuação de confiança calculada anteriormente para cada *bounding box* e a predição condicional da classe são combinadas em uma pontuação final que nos diz a probabilidade de que cada *bounding box* contenha um tipo específico de objeto, conforme mostra a Figura 11d. Além disso, detecções duplicadas prevendo o mesmo objeto são removidas usando *non-max suppression* (supressão não máxima).

Em resumo, o detector de objetos *YOLO* utiliza uma *CNN* para treinar e prever

Figura 11 – Visão geral do modelo utilizado no detector de objetos *YOLO*: (11a) exemplo de uma imagem de entrada, dividida em uma grade de tamanho $S = 7$, em destaque vermelho, um exemplo de célula; (11b) exemplo de B *bounding boxes* por célula e seu respectivo valor de confiança, representado pela espessura das linhas; (11c) exemplo de um mapa de probabilidades de classes, onde cada célula apresenta a probabilidade de apenas uma classe; e (11d) as detecções finais, resultantes da combinação de (11b) e (11c).



Fonte: Adaptado pelo Autor (REDMON et al., 2016).

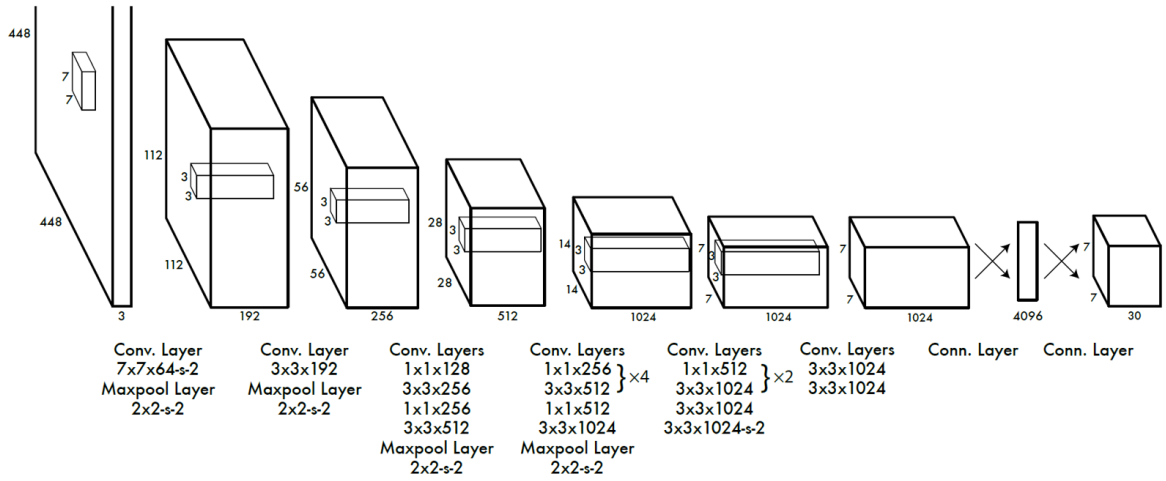
um tensor de saída para que, de uma só vez, calcule todas as detecções de uma imagem. Um dos principais segredos do porquê *YOLO* ser tão rápida é que as previsões de saída são todas feitas ao mesmo tempo olhando apenas uma vez para a imagem, e é por isso que é chamada de *YOLO* (*You Only Look Once* - "você só olha uma vez").

3.2.2 Arquitetura

A arquitetura proposta por Redmon et al. (2016) é inspirada no modelo para classificação de imagens, chamado *Inception* (SZEGEDY et al., 2015), conhecido também como *GoogLeNet*. A arquitetura apresenta 24 camadas convolucionais seguidas de duas camadas totalmente conectadas. A Figura 12 apresenta o esquemático da arquitetura utilizada no detector de objetos *YOLO*.

Pode-se destacar a última camada, o tensor final de predição, que carrega informações importantes de cada célula. A dimensão deste tensor é uma relação entre o tamanho da grade (S), o número de propostas de *bounding boxes* (B) e a quantidade de classes (C). Dessa forma, a camada final da arquitetura do *YOLO* é um tensor de predição de tamanho $S \times S \times (5B + C)$. O valor referente às classes (C) é um conjunto de valores referente a quantidade de classes e o termo $5B$ apresenta a multiplicação da quantidade de propostas de *bounding boxes* (B) com a quantidade de termos presentes em uma *bounding box* (5): p_c - confiança da *bounding box*; b_x - coordenadas x representando o centro da *bounding box*; b_y - coordenadas y representando o centro da *bounding box*; b_w - dimensão referente a largura da *bounding box* relativa a imagem; e, b_h - dimensão referente a altura da *bounding box* relativa a imagem. Para exemplificar, Redmon et al. (2016) validam o detector *YOLO*

Figura 12 – Visão geral do modelo utilizado no detector de objetos *YOLO*. A arquitetura apresenta 24 camadas convolucionais seguida de duas camadas totalmente conectadas.



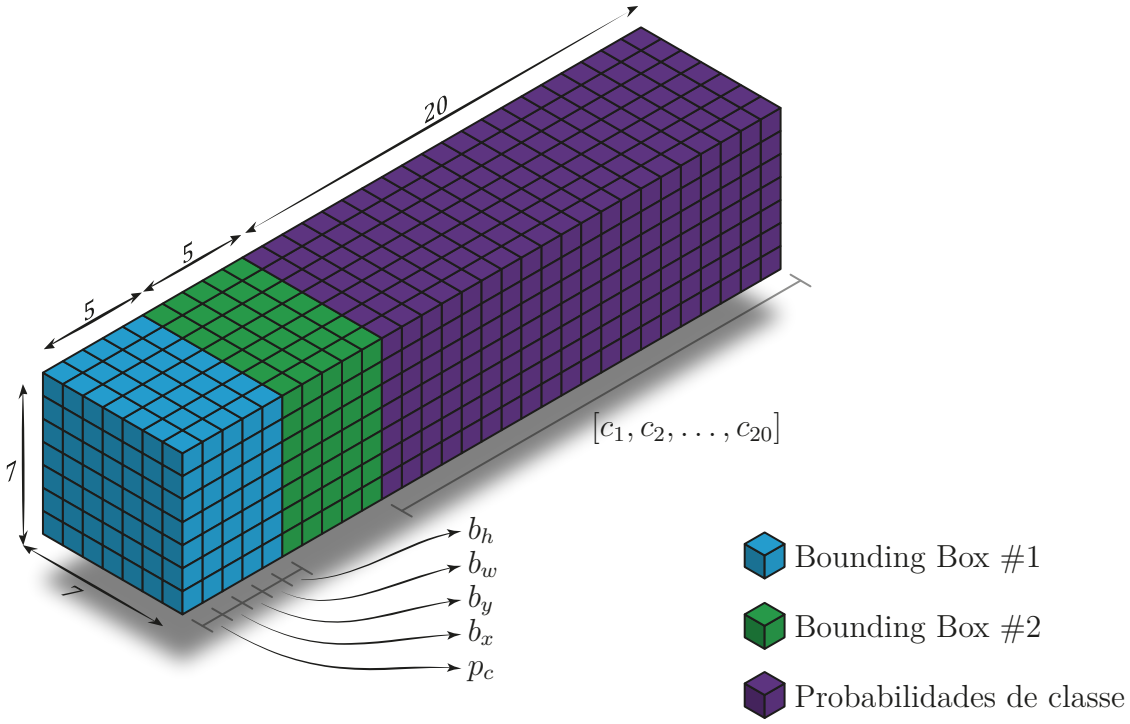
Fonte: (REDMON et al., 2016).

no conjunto de dados *PASCAL VOC* (EVERINGHAM et al., 2010), utilizando uma grade 7×7 ($S = 7$), duas propostas de *bounding box* ($B = 2$) e utilizando as 20 classes rotuladas do conjunto de dados ($C = 20$). A Figura 13 apresenta uma interpretação visual dessa última camada, resultando um tensor de tamanho $7 \times 7 \times (5 \times 2 + 20) = 7 \times 7 \times 30$.

Para exemplificar a codificação presente no tensor final de predição, a Figura 14 exemplifica a detecção em uma imagem com $S = 3$, $B = 2$ e $C = 2$. Cada célula é responsável por conter as informações referentes as *bounding boxes*, o valor de confiança das detecções e um conjunto de valores referente à probabilidade de conter uma das classes. O valor de confiança (p_c) indica a probabilidade de uma detecção conter um objeto. As *bounding boxes* apresentam 4 números referentes ao posicionamento e ao tamanho da detecção na imagem, dois referentes ao centro da marcação em relação à célula (b_x, b_y) e os outros dois representam o comprimento e a largura da marcação (b_w, b_h) em relação ao tamanho da imagem. Portanto, todos os 4 valores (b_x, b_y, b_w, b_h) estão entre 0 e 1. O último conjunto de dados apresenta um vetor de tamanho C, onde indica apenas uma classe por célula (Figura 11c).

A arquitetura dessa rede não é simples, porém robusta, onde, dada uma imagem de entrada, é necessário uma única passagem pela rede neural convolucional, gerando um tensor descrevendo todas as informações previstas nas células da grade, obtendo, depois de uma combinação dos valores encontrados e alguns filtros, as devidas *bounding boxes*.

Figura 13 – Codificação do tensor final de predição do detector de objetos *YOLO*. A figura ilustra um tensor com $S = 7$, $B = 2$ e $C = 20$, resultando em um tensor de tamanho $7 \times 7 \times 30$.



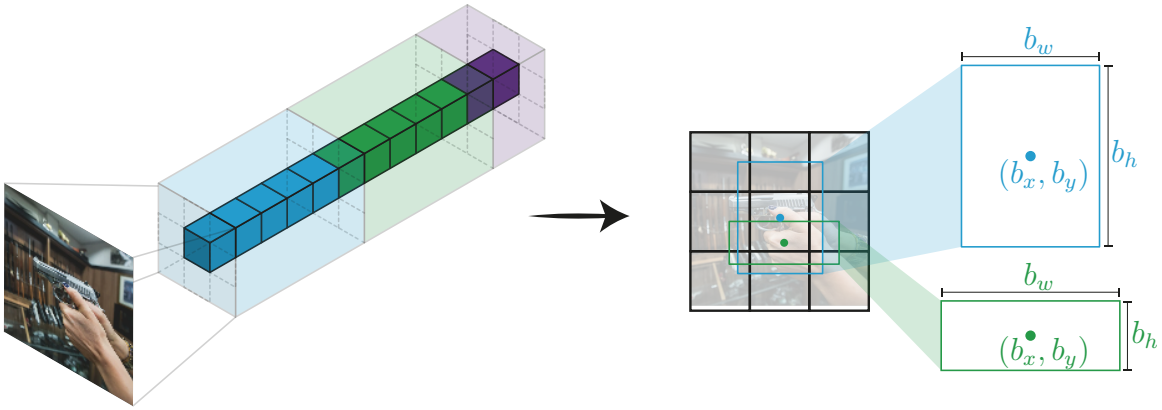
Fonte: o Autor.

3.2.3 Treinamento

Como o detector *YOLO* prevê todas as instâncias existentes de objetos a partir de uma única imagem completa, ele também deve ser treinado com imagens completas. Dessa forma, dada uma imagem de treinamento juntamente com os rótulos de *ground truth*, o *YOLO* combina esses rótulos com as células de grade apropriadas. Isso é feito calculando o centro de cada *bounding box* e onde quer que a célula da grade esteja nesse centro, será a responsável por prever essa detecção. Em seguida, o *YOLO* ajusta as previsões de classe dessa célula e suas propostas de *bounding box*. A *bounding box* que mais se sobrepor ao rótulo de *ground truth* terá sua pontuação de confiança aumentada e suas coordenadas ajustadas enquanto as detecções restantes simplesmente terão sua pontuação de confiança diminuída, pois elas não se sobrepõem corretamente ao objeto.

Algumas células restantes não têm detecções verdadeiras atribuídas a elas. Neste caso, a confiança de todas as *bounding boxes* previstas é diminuída, uma vez que elas não são responsáveis por prever qualquer objeto. Uma coisa importante a se notar é que, neste caso, o *YOLO* não ajusta as probabilidades de classe ou coordenadas para as propostas

Figura 14 – Exemplo da codificação presente em um tensor final de predição com $S = 3$, $B = 2$ e $C = 2$. A figura exemplifica a visualização das *bounding boxes* a partir do tensor.



Fonte: o Autor.

de *bounding boxes*, uma vez que não há nenhum objeto de verdade real atribuído a essas células.

Em resumo, o treinamento do detector de objetos *YOLO* é bastante direto e corresponde a padrões na comunidade de Visão Computacional: *YOLO* é pré-treinado no *ImageNet*; usa descida de gradiente estocástica; e usa *data augmentation* (aumento de dados).

3.2.4 Vantagens do detector de objetos *YOLO*

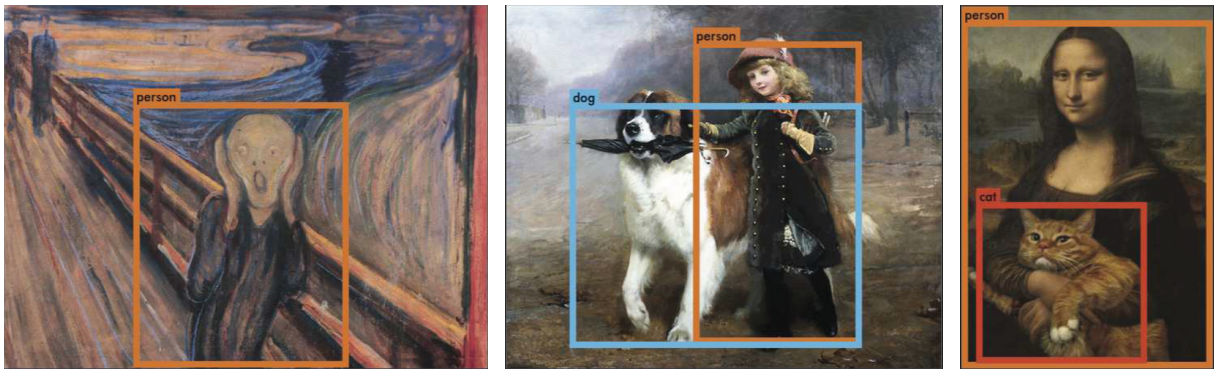
Um dos principais benefícios de usar o *YOLO* é que ele é consideravelmente rápido, permitindo o processamento em tempo real. Além disso, as previsões (localizações e classes de objetos) são feitas a partir de uma única rede, podendo ser treinada de ponta a ponta para melhorar a precisão.

O *YOLO* supera outros métodos ao generalizar imagens naturais para outros "domínios", como exemplo, obras de arte. A Figura 15 apresenta alguns exemplos de generalização do *YOLO*, sendo capaz de detectar objetos em representações abstratas.

Outra grande vantagem é que o *YOLO* analisa globalmente a imagem ao fazer previsões. Ao contrário de janela deslizante e técnicas baseadas em proposta de região, o *YOLO* vê a imagem inteira durante o treinamento e o teste, portanto, codifica implicitamente as informações contextuais sobre as classes, bem como sua aparência. Com o contexto adicional, o *YOLO* demonstra menos falsos positivos nas áreas de fundo (REDMON et al., 2016). Em contrapartida, o detector pode ter dificuldades em detectar pequenos objetos na imagem, isso se deve às restrições espaciais do algoritmo (GANDHI, 2018).

Figura 15 – Generalização do detector de objetos *YOLO* para imagens abstratas (WESTLAKE et al., 2016): (15a) apresenta uma detecção da classe *"person"* (pessoa) na obra "O Grito" de Edvard Munch; (15b) apresenta duas detecções, uma da classe *"dog"* (cachorro) e outra da classe *"person"* (pessoa), ambas na obra "Fora da Escola" de Charles Burton Barber; por fim, (15c) apresenta duas detecções, uma da classe *"cat"* (gato) e outra da classe *"person"* (pessoa), ambas na releitura da obra de Leonardo da Vinci, intitulada *"Mona Lisa, True version"* (em português: Mona Lisa, versão verdadeira) de Svetlana Petrova.

(a) (b) (c)



Fonte: (REDMON et al., 2016).

3.2.5 *YOLOv2*

Desde a criação do detector de objetos *YOLO*, algumas versões foram apresentadas, sendo supracitadas no começo dessa seção. Porém, neste trabalho, todo o desenvolvimento foi realizado com a versão 2 da *YOLO*, a *YOLOv2* (REDMON; FARHADI, 2017). Doravante, para evitar erros de interpretação pelas diversas siglas ao longo deste trabalho, é de suma importância reforçar que a nomenclatura referente à versão 2 do detector de objetos será: *YOLOv2*. Da versão inicial ao *YOLOv2*, algumas melhorias incrementais foram aplicadas, aumentando a precisão significativamente e, ao mesmo tempo, tornando-o mais rápido.

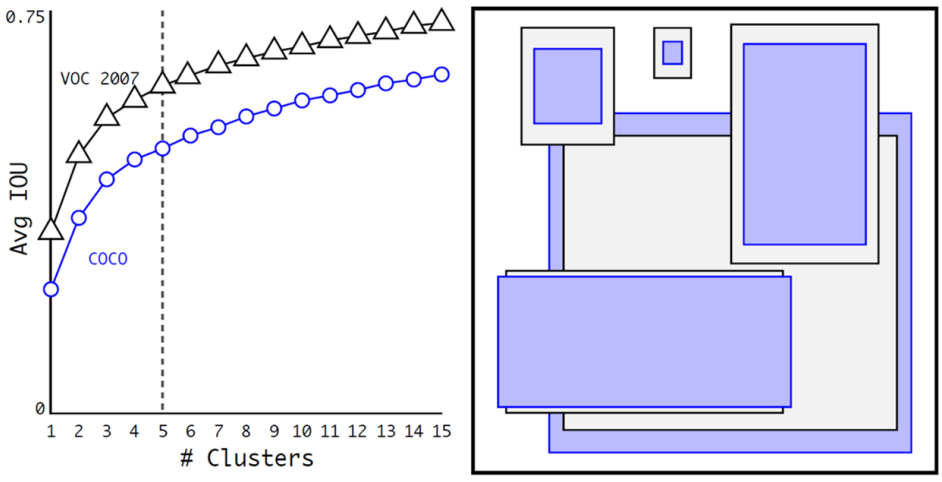
Redmon e Farhadi (2017) destacam 7 melhorias no *YOLOv2*, sendo elas: *batch normalization* (normalização de lote); *high resolution classifier* (classificador de alta resolução); *convolutional with anchor boxes* (convolução com "caixas âncoras"); *dimension clusters* (*clusters* de dimensão); *direct location prediction* (previsão de localização direta); *fine-grained features* (características refinadas); e *multi-scale training* (treinamento multi-escala). Dentre elas, pode-se destacar a convolução com "caixas âncoras" e *clusters* de dimensão.

Convolução com "caixas âncoras", ou em inglês, *convolutional with anchor boxes*, é uma das melhorias presentes no *YOLOv2*. A primeira versão do *YOLO* apresenta uma codificação na camada final, na qual camadas totalmente conectadas (*fully connected*) são

utilizadas para prever *bounding boxes*. Além disso, na primeira versão, embora possa existir B propostas de *bounding boxes*, apenas um objeto é previsto por cada célula, limitando o número de previsões. Por exemplo, utilizando uma arquitetura com $S = 7$ e $B = 2$, *YOLO* (versão 1) apresenta 49 células ($S \times S = 49$), onde cada célula apresenta duas propostas de *bounding box* ($B = 2$), logo, o detector de objetos pode predizer 98 "marcações". Na versão 2 do detector de objetos *YOLO*, visando a predição de mais objetos por célula, as camadas totalmente conectadas foram substituídas por camadas convolucionais, assim como é utilizado o conceito de "caixas âncoras" (*anchor boxes*) utilizado em outros sistemas de detecção, como exemplo o *Faster R-CNN*.

O *cluster* de dimensão (em inglês, *dimension clusters*), outra melhoria presente no *YOLOv2*, aparece como uma etapa em paralelo das convoluções com "caixas âncoras". Basicamente, no conjunto de treinamento, é executado o algoritmo *k-means clustering* para vários valores de k (Figura 16), dessa forma, obtendo os melhores valores das dimensões dos *bounding boxes*, sendo utilizados como "caixas âncoras". Redmon e Farhadi (2017) afirmam que $k = 5$ apresenta uma boa relação entre o *Recall* (Seção 4.3.1) e a complexidade do modelo.

Figura 16 – Resultados obtidos utilizando o *dimension clusters*. A imagem da esquerda apresenta a média de IoU (*intersection over union*) pela quantidade de *clusters* avaliados. Segundo Redmon e Farhadi (2017), $k = 5$ representa um equilíbrio entre o *Recall* e a complexidade do modelo. A imagem da direita apresenta o encaixe de detecções para testes executados com $k = 5$ em algumas bases de dados.



Fonte: (REDMON; FARHADI, 2017).

Em resumo, o detector de objetos *YOLOv2* utiliza os *clusters* de dimensão para tentar encontrar as melhores formas de "caixas âncoras", auxiliando no treinamento da rede neural e predizendo, de uma só vez, as *bounding boxes* e seus respectivos valores de

confiança.

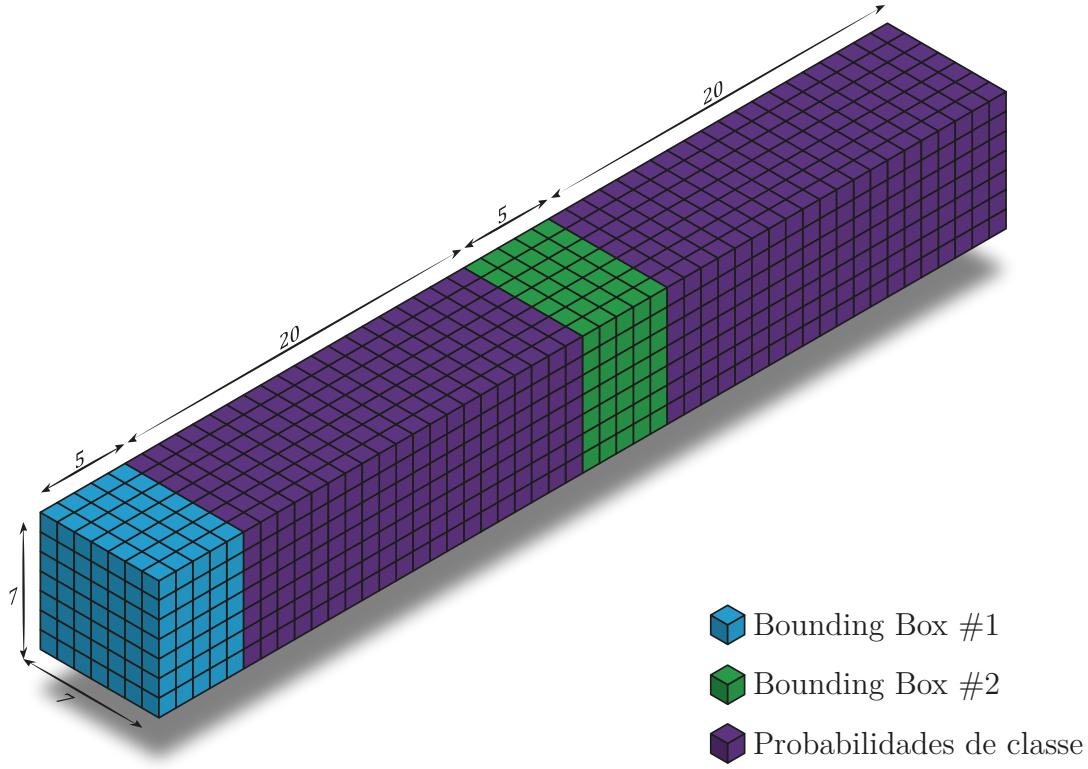
A primeira versão do *YOLO* é limitada à uma detecção por célula, onde, as B propostas de *bounding boxes* são avaliadas com o vetor de probabilidades de umas das C classes, retornando apenas uma marcação, caso o valor de confiança ultrapasse um limiar. O detector de objetos *YOLOv2* utiliza a convolução com "caixas âncoras" e o *cluster* de dimensão para prever mais de uma detecção por célula. Além disso, uma nova arquitetura foi introduzida para este detector - *Darknet-19* - uma rede neural com 19 camadas convolucionais e 5 camadas de *maxpooling*. Tais modificações implicaram diretamente no tensor final de predição, uma vez que as camadas totalmente conectadas foram substituídas por camadas convolucionais.

Mais uma vez, pode-se destacar a última camada, o tensor final de predição, que carrega informações das predições de cada célula. O padrão utilizado na codificação é igual à versão 1, ou seja, cada proposta de detecção apresenta um conjunto de valores: a probabilidade de um objeto existir (p_c); as coordenadas da detecção (b_x, b_y, b_w, b_h); e, por fim, um vetor com a probabilidade de uma das C classes estar presente na célula.

No *YOLOv2*, a dimensão D do tensor final é semelhante a versão 1, mas considerando apenas uma proposta de *bounding box* ($B = 1$), sendo multiplicado pela quantidade k de *bounding boxes* que uma única célula da grade é responsável por detectar. Dessa forma, a dimensão do tensor final de predição fica variável e condicionada ao valor de k e não mais a B ($B = 1$): $S \times S \times D$, onde, $D = (5B + C) \times k = (5 + C) \times k$. A Figura 17 ilustra uma representação gráfica do tensor final com $k = 2$.

De maneira geral, o *YOLOv2* é um detector de objetos poderoso, aliado com o *framework open source Darknet* (REDMON, 2013–2016), desenvolvido pelos próprios autores do *YOLO*, possibilita inúmeras modificações e aplicações. A Seção 4.2 aborda com detalhes as configurações utilizadas ao longo deste trabalho, abordando a arquitetura utilizada e o ambiente de desenvolvimento.

Figura 17 – Codificação do tensor final de predição do detector de objetos *YOLOv2*. A figura ilustra um tensor com $S = 7$, $C = 20$ e $k = 2$, resultando em um tensor de tamanho $7 \times 7 \times 50$.



Fonte: o Autor.

4 Detecção de Armas de Fogo em Imagens

A proposta deste trabalho é utilizar técnicas computacionais que envolvam Redes Neurais Convolucionais Profundas com o intuito de detectar armas de fogo em imagens. Para tal, este capítulo tem por objetivo apresentar as etapas utilizadas ao longo deste trabalho. A Figura 18 apresenta um procedimento que pode ser resumido em três etapas: conjunto de dados, detector de objetos e testes e avaliação dos resultados. A Seção 4.1 apresenta os conjuntos de dados utilizados para treinamento, validação e avaliação dos resultados; a Seção 4.2 apresenta o detector de objetos utilizado junto com as suas respectivas configurações; por fim, a Seção 4.3 apresenta métricas que possibilitam avaliar o desempenho do detector de objetos.

Figura 18 – Diagrama dos procedimentos utilizados neste trabalho.

Fonte: o Autor.

4.1 Conjunto de Dados

Alguns dos trabalhos apresentados no Capítulo 2 utilizam meios manuais ou automáticos para a construção de seus bancos, onde utilizam, em sua grande maioria, dados do *site* IMFDb[1]. Este *site* é uma espécie de enciclopédia *on-line* que reúne informações de todas as armas de fogo exibidas em filmes, programas de televisão, desenhos e jogos. Porém, estes trabalhos transgridem no que diz respeito a sua reprodutibilidade, onde apresentam seus resultados mas não compartilham seus respectivos conjuntos de dados.

Por isso, os conjuntos de dados utilizados neste trabalho foram construídas e avaliadas por Olmos et al. (2018), sendo disponibilizados em um repositório público[2]. A Tabela 1 descreve cada conjunto de dados contendo informações do número total de

[1] http://www.imfdb.org/wiki/Main_Page
[2] https://github.com/SihamTabik/Pistol-Detection-in-Videos

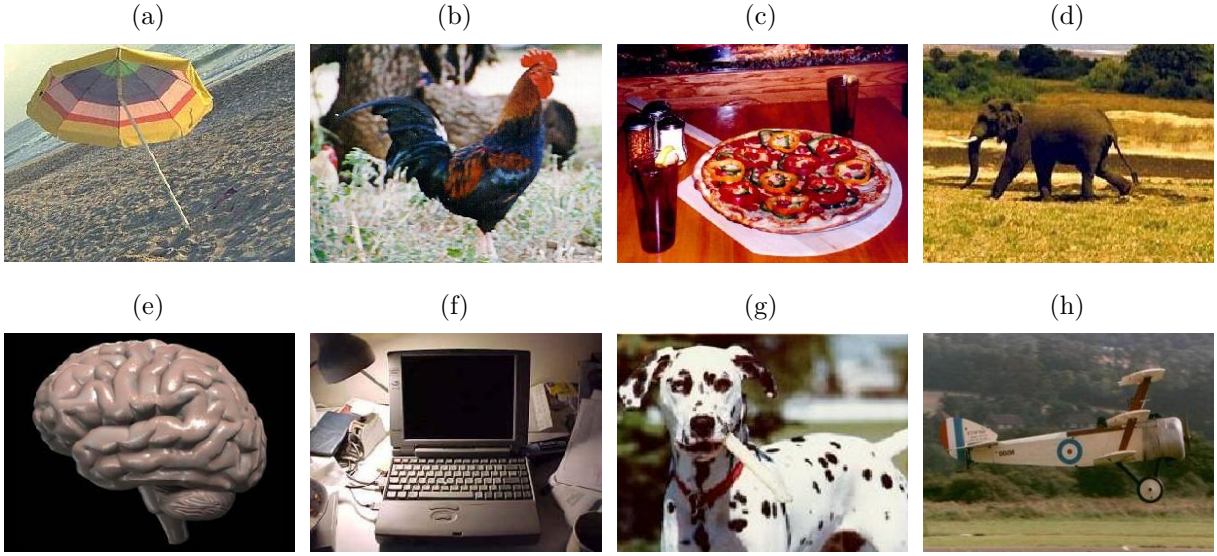
imagens, imagens com armas e imagens anotadas. Os conjuntos originais *BK4*, *WeaponS* e *Test Set* foram renomeadas como DB-1, DB-2 e DB-T, respectivamente. Alguns termos utilizados ao longo do texto necessitam de atenção: "anotações" de uma imagem se refere ao arquivo referente à uma imagem; *bounding boxes* (em português, caixas delimitadoras) se refere às informações dos objetos contidos em uma imagem (localização e rótulo) presentes na anotação de uma imagem.

Tabela 1 – Características dos bancos de dados.

Conjunto de Dados	Total img	# img com armas	# anot.
DB-1	9857	795	0
DB-2	3000	3000	3000
DB-T	608	304	0

O primeiro conjunto de dados, renomeado neste trabalho como DB-1, é a maior conjunto de dados dentre as três, e foi construída para avaliar a abordagem de *sliding window* de Olmos et al. (2018). Ela possui imagens de resoluções variadas, divididas em 102 classes distintas, sendo uma delas de armas. A base DB-1 é constituída por diversos tipos de imagens. A Figura 19 apresenta algumas imagens das 102 classes desta base.

Figura 19 – Exemplos de imagens do conjunto de dados DB-1.

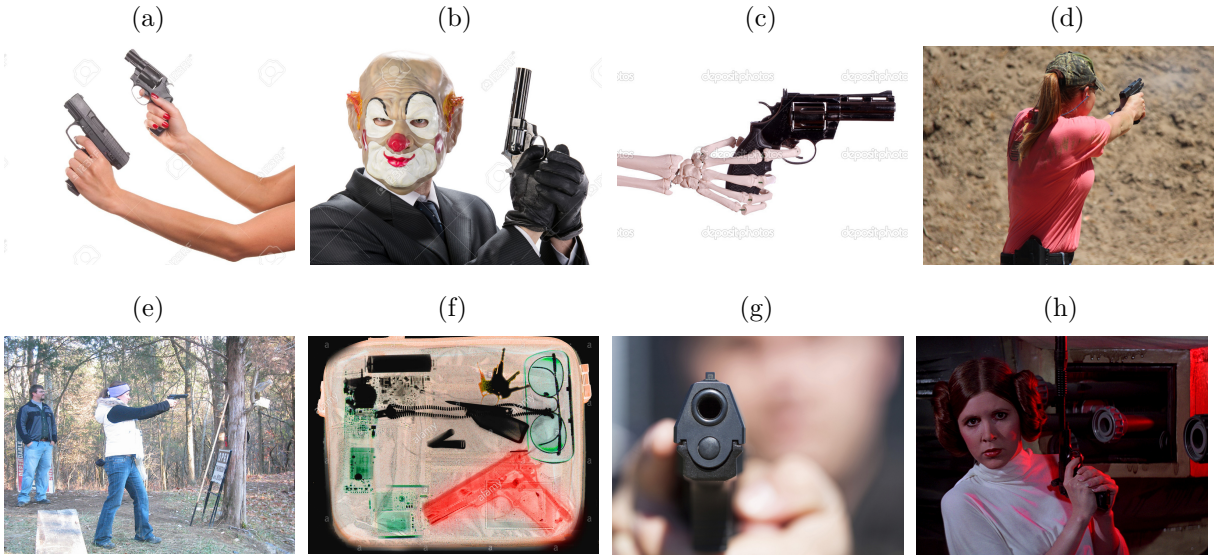


Fonte: Olmos et al. (2018).

O conjunto de dados DB-2 foi construído especificamente para o método *Region Proposal*, um conjunto de dados que contém apenas armas e as suas devidas localizações nas imagens, os *bounding boxes*. Essa base foi construída do zero, sem reaproveitar nenhuma das 795 imagens com armas da base DB-1. Vale lembrar que ela possui imagens de tamanhos

variados e que em cada uma das imagens contém pelo menos uma arma visível, conforme pode ser visto na Figura 20.

Figura 20 – Exemplos de imagens do conjunto de dados DB-2.



Fonte: Olmos et al. (2018).

A base de testes (DB-T) foi construída de tal maneira a criar um equilíbrio entre imagens com armas e não armas. Contém 608 imagens de resoluções fixas de 160×120 *pixels*, onde metade dessas imagens são de armas. Ela foi construída para se tornar um problema de classificação binária, e por isto não possui as *bounding boxes* das localizações das armas. As 304 primeiras imagens (1-304) são amostras negativas e as outras 304 imagens (305-608) são de armas. Algumas imagens desse banco podem ser observadas na Figura 21.

4.2 Detector de Objetos

O objetivo dos trabalhos apresentados no Capítulo 2, em sua grande maioria, é detectar armas de fogo em imagens ou vídeos de maneira automática em tempo real, porém, o tempo de detecção é um fator que limita tais abordagens. Diante disso, o detector de objetos utilizado neste trabalho será o *YOLOv2*, apresentado na Subseção 3.2.5, um detector de objetos voltado para aplicações em tempo real, sendo implementado pelo *framework open source*, desenvolvido em linguagem C, chamado de *Darknet* (REDMON, 2013–2016).

Redmon e Farhadi (2017) desenvolveram uma arquitetura para ser usada como base no detector de objetos *YOLOv2*. A arquitetura recebeu o nome de *Darknet19*[3] pois conta

[3] https://github.com/pjreddie/darknet/blob/master/cfg/darknet19.cfg

Figura 21 – Exemplos de imagens do conjunto de dados DB-T.

Fonte: Olmos et al. (2018).

com 19 camadas convolucionais e 5 camadas de *max-pooling*. Semelhante aos modelos VGG (SIMONYAN; ZISSERMAN, 2014), essa arquitetura utiliza filtros de tamanho 3×3 e dobrando o número de filtros/canais após cada camada de *max-pooling*.

A arquitetura *Darknet19* foi desenvolvida para a tarefa de detecção e classificação, utilizando algumas abordagens para treinamento e validação em algumas bases de dados: *ImageNet* (DENG et al., 2009); *Pascal VOC* (EVERINGHAM et al., 2010); e *COCO Dataset* (LIN et al., 2014). A abordagem de Redmon e Farhadi (2017) se mostrou eficaz nas tarefas de detecção e classificação e, ainda, apresentou superioridade na detecção em tempo real - comparada com outros detectores de objetos - onde o tempo do detector de objetos *YOLOv2* apresenta resultados entre 40-90 *fps* no conjunto de dados *Pascal VOC*.

A arquitetura *Darknet19* não só foi utilizada para avaliação dos resultados perante os apresentados na literatura, como proporcionou a construção de outras arquiteturas. Uma delas, chamada simplesmente de (arquitetura) `yolov2`, apresenta a estrutura da *Darknet19*, mas com o acréscimo de 4 camadas convolucionais. Redmon (2013–2016) utiliza essa arquitetura (`yolov2`) para demonstrar sua utilização em seu *site*[4], porém, em momento algum explica o motivo da utilização da arquitetura `yolov2` ao invés da *Darknet19*. Devido à arquitetura (`yolov2`) ter o mesmo nome do detector de objetos (*YOLOv2*), vale reforçar que ambos os nomes são conceitualmente diferentes: *YOLOv2* - detector de objetos; `yolov2` - arquitetura.

A arquitetura proposta nesta metodologia é baseada na arquitetura `yolov2`, onde sua saída padrão foi modificada para atender as especificações previstas neste trabalho.

[4] https://pjreddie.com/darknet/yolov2/

A Tabela 2 apresenta a arquitetura utilizada pelo detector de objetos (*YOLOv2*) neste trabalho e a Equação 4.1 define o tamanho do tensor final de predição na última camada convolucional. Conforme explicado na Seção 3.2.5, C é a quantidade de classes a ser detectada e, seguindo os resultados experimentais de Redmon e Farhadi (2017), o fator multiplicador k, referente as propostas de *bounding boxes*, será igual a 5.

$$D_{yolov2} = (5 + C) \times k = (5 + C) \times 5 \tag{4.1}$$

Tabela 2 – Arquitetura `yolov2`: uma arquitetura baseada na *Darknet19*. A coluna "Camada" apresenta a abreviação de algumas camadas, sendo elas: ***conv***: camada convolucional; ***max***: camada de *max-pooling*; ***route***: rotas; e ***reorg***: camada de reorganização

#	Camada	Filtro	Tamanho/Passo	Entrada	Saída
00	conv	32	3 × 3 / 1	608 × 608 × 3	608 × 608 × 32
01	max		2 × 2 / 2	608 × 608 × 32	304 × 304 × 32
02	conv	64	3 × 3 / 1	304 × 304 × 32	304 × 304 × 64
03	max		2 × 2 / 2	304 × 304 × 64	152 × 152 × 64
04	conv	128	3 × 3 / 1	152 × 152 × 64	152 × 152 × 128
05	conv	64	1 × 1 / 1	152 × 152 × 128	152 × 152 × 64
06	conv	128	3 × 3 / 1	152 × 152 × 64	152 × 152 × 128
07	max		2 × 2 / 2	152 × 152 × 128	76 × 76 × 128
08	conv	256	3 × 3 / 1	76 × 76 × 128	76 × 76 × 256
09	conv	128	1 × 1 / 1	76 × 76 × 256	76 × 76 × 128
10	conv	256	3 × 3 / 1	76 × 76 × 128	76 × 76 × 256
11	max		2 × 2 / 2	76 × 76 × 256	38 × 38 × 256
12	conv	512	3 × 3 / 1	38 × 38 × 256	38 × 38 × 512
13	conv	256	1 × 1 / 1	38 × 38 × 512	38 × 38 × 256
14	conv	512	3 × 3 / 1	38 × 38 × 256	38 × 38 × 512
15	conv	256	1 × 1 / 1	38 × 38 × 512	38 × 38 × 256
16	conv	512	3 × 3 / 1	38 × 38 × 256	38 × 38 × 512
17	max		2 × 2 / 2	38 × 38 × 512	19 × 19 × 512
18	conv	1024	3 × 3 / 1	19 × 19 × 512	19 × 19 × 1024
19	conv	512	1 × 1 / 1	19 × 19 × 1024	19 × 19 × 512
20	conv	1024	3 × 3 / 1	19 × 19 × 512	19 × 19 × 1024
21	conv	512	1 × 1 / 1	19 × 19 × 1024	19 × 19 × 512
22	conv	1024	3 × 3 / 1	19 × 19 × 512	19 × 19 × 1024
23	conv	1024	3 × 3 / 1	19 × 19 × 1024	19 × 19 × 1024
24	conv	1024	3 × 3 / 1	19 × 19 × 1024	19 × 19 × 1024
25	route	#16			
26	conv	64	1 × 1 / 1	38 × 38 × 512	38 × 38 × 64
27	reorg		/ 2	38 × 38 × 64	19 × 19 × 256
28	route	#27 #24			
29	conv	1024	3 × 3 / 1	19 × 19 × 1280	19 × 19 × 1024
30	conv	D_{yolov2}	1 × 1 / 1	19 × 19 × 1024	19 × 19 × D_{yolov2}

A Tabela 2 apresenta dois tipos de camadas não convencionais, a camada *route*

e a camada *reorg*. A camada *route* (camada de rotas) tem por objetivo criar rotas entre camadas e também concatenar a saída de duas camadas, ficando condicionada à quantidade de argumentos na coluna "Camada". Considerando um argumento, uma camada de rota utiliza a saída da camada especificada como entrada para a camada subsequente à camada rotas. No caso de dois argumentos na coluna "Camada", ou seja, o número de duas camadas, a camada rotas concatena as duas saídas em apenas uma única saída. Por fim, a camada *reorg* (camada de reorganização) tem o simples objetivo de reestruturar uma camada, modificando suas dimensões de saída sem perder informação ou alterar elementos. Em resumo, visando o contexto do YOLOv2, uma camada *route* é uma camada de redirecionamento (rotas) e uma camada *reorg* é uma camada de reorganização ou *reshape*.

Assim, é possível compreender, de maneira mais clara, a proposta da arquitetura `yolov2` de Redmon e Farhadi (2017). Em resumo, a camada da linha #25 da Tabela 2 cria uma rota com a camada #16, dessa forma, a camada #26 recebe como entrada a saída da rota especificada (#16). A camada da linha #27 reestrutura a saída da camada convolucional (#26), pois a camada de rotas da linha #28 é responsável por concatenar as saídas #24 e #27, e utilizar como entrada para a penúltima camada. A camada final (#30), também chamada de tensor final de predição, representa a codificação das predições, onde sua dimensão é condicionada ao número de classes (C), presente na Equação 4.1, e representa a codificação das predições, conforme explicado na Subseção 3.2.5. Para exemplificar, a Figura 22 apresenta um diagrama da arquitetura apresentada na Tabela 2, demonstrando, de maneira visual, as camadas convolucionais, as camadas de *max-pooling*, a camada de reorganização e as rotas.

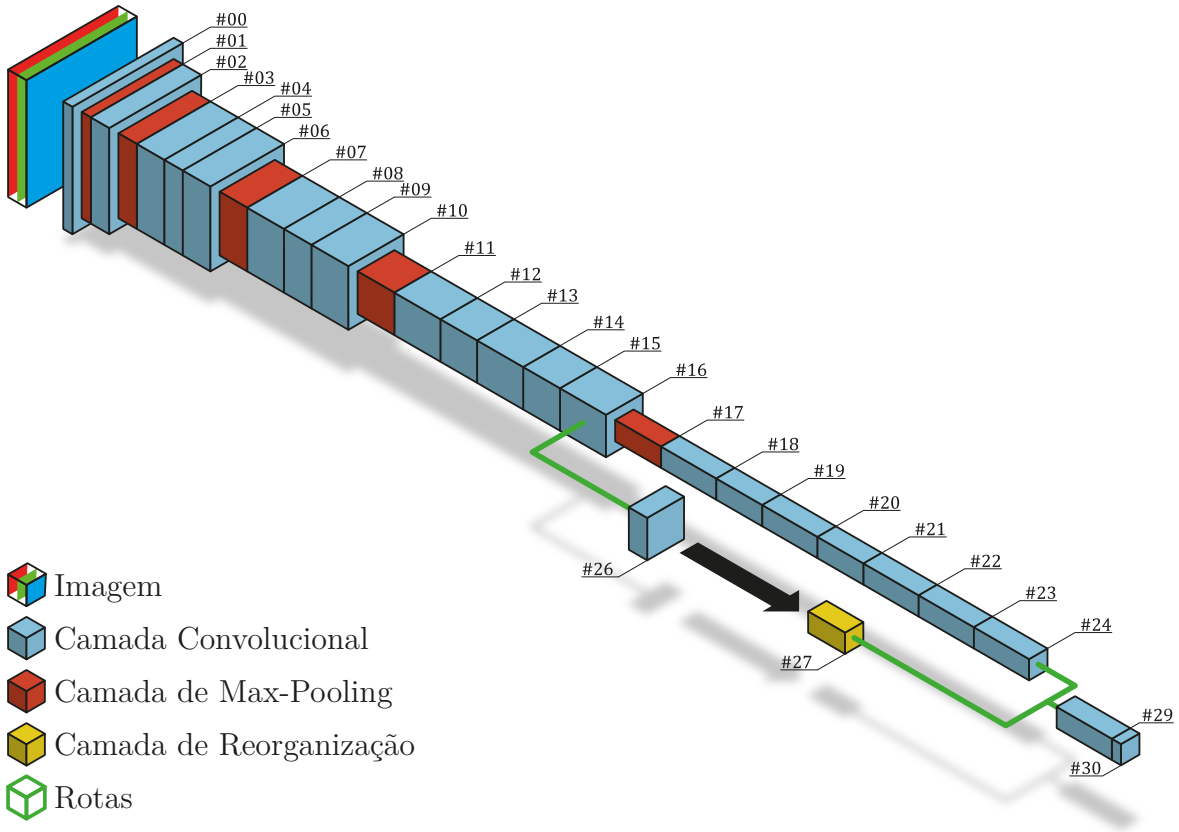
4.3 Testes e Avaliação dos Resultados

A presente seção apresenta as métricas de avaliação utilizadas neste trabalho, sendo responsável por mensurar o desempenho da metodologia proposta segundo diferentes critérios. Baseado nos resultados apresentados na literatura sobre o conjunto de dados DB-T, as métricas utilizadas neste trabalho são similares aos trabalhos de Olmos et al. (2018) e Elmir et al. (2019) - precisão, sensibilidade e F_1 *Score* - além disso, outras métricas vão ser utilizadas, com a finalidade de avaliar o modelo proposto.

4.3.1 Classificação Binária

Considerando que todas as avaliações serão realizadas sobre o banco de testes DB-T e que cada imagem contém ou não uma arma, pode-se considerar o problema de detecção de armas como uma classificação binária. Para avaliar o desempenho da abordagem, foram utilizadas as seguintes métricas: Precisão (Pr), Sensibilidade (*Recall*) (Se) e F_1 *Score* (F_1).

Figura 22 – Diagrama visual da arquitetura *yolov2*, apresentada na Tabela 2.



Fonte: o Autor.

A precisão avalia a capacidade do modelo em detectar corretamente as imagens que realmente possuam armas; a sensibilidade, também conhecida como *Recall*, avalia a capacidade do modelo em detectar todas as armas de fogo do conjunto de dados; e, finalmente, a medida F_1 *Score*, que é a média harmônica entre a precisão e a sensibilidade.

Todas as métricas supracitadas são apresentadas nas Equações 4.2 a 4.4, onde: VP são os verdadeiros positivos (número de imagens com armas que o modelo detectou corretamente ao menos uma arma de fogo), VN são os verdadeiros negativos (número de imagens sem armas nas quais o modelo corretamente não detectou armas de fogo), FP são os falsos positivos (número de imagens sem armas que o modelo erroneamente detectou ao menos uma arma de fogo), e FN são os falsos negativos (número de imagens com armas nas quais o modelo erroneamente não detectou armas de fogo).

$$Pr = \frac{VP}{VP + FP} \times 100\% \tag{4.2}$$

$$Se = \frac{VP}{VP + FN} \times 100\% \tag{4.3}$$

$$F_1 = 2 \times \frac{Pr \times Se}{Pr + Se} \times 100\% \tag{4.4}$$

4.3.2 Curva ROC

A curva ROC, do inglês, *Receiver Operating Characteristic Curve* (Curva Característica de Operação do Receptor), é uma forma gráfica de representar o desempenho de um modelo para problemas de classificação binária, onde o seu limiar de discriminação pode variar (PRATI et al., 2008). Em outras palavras, a curva ROC avalia a dispersão da taxa de verdadeiros positivos × taxa de falsos positivos em função de *thresholds* aplicados na classificação. Um exemplo de curva ROC é apresentado na Figura 23.

Essa representação gráfica é construída a partir das Equações 4.5 e 4.6, onde: a taxa de verdadeiros positivos (TVP) (também conhecida como sensibilidade ou *Recall* - Equação 4.3), avalia a capacidade do modelo em detectar todas as amostras positivas do conjunto de dados; e a taxa de falsos positivos (TFP), por outro lado, avalia as detecções erradas do modelo. Dessa forma, um modelo ideal deve apresentar um alto número de TVP e um baixo número de TFP.

Alguns pontos devem ser levados em consideração no espaço ROC: o ponto `(0,0)` representa um modelo que não erra nenhuma classificação, pois não classifica nada; o ponto `(1,1)`, por sua vez, classifica toda e qualquer amostra, eliminando os falsos negativos e verdadeiros negativos, o que resulta em uma alta taxa de verdadeiros positivos e falsos positivos; o ponto `(1,0)` representa um modelo que nunca acerta suas classificações; por fim, o ponto `(0,1)` representa um modelo perfeito, onde todas as amostras positivas são classificadas corretamente (PRATI et al., 2008).

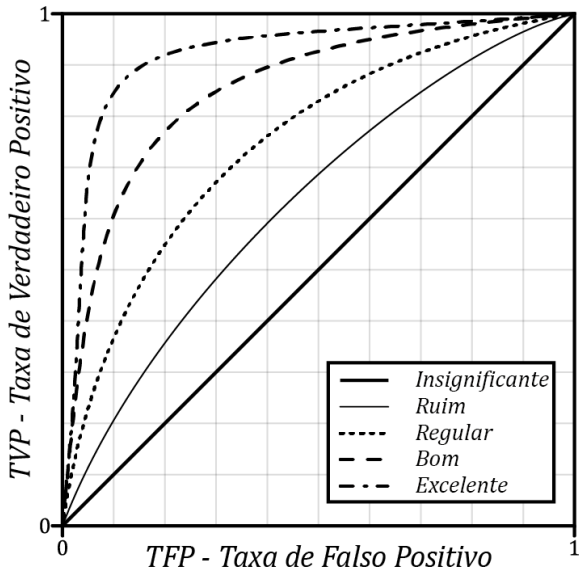
$$TVP = Se = \frac{VP}{VP + FN} \times 100\% \tag{4.5}$$

$$TFP = \frac{FP}{FP + VN} \times 100\% \tag{4.6}$$

4.3.3 Interseção sobre União

As métricas apresentadas anteriormente avaliam o desempenho do detector de objetos em relação à precisão, sensibilidade e à medida F_1, além de permitir avaliar o desempenho de detecção da solução através da variação de um limiar a partir da curva ROC. Porém, nenhuma dessas métricas possui a capacidade de avaliar as *bounding boxes*

Figura 23 – Exemplo de 5 curvas ROC's diferentes.



Fonte: o Autor.

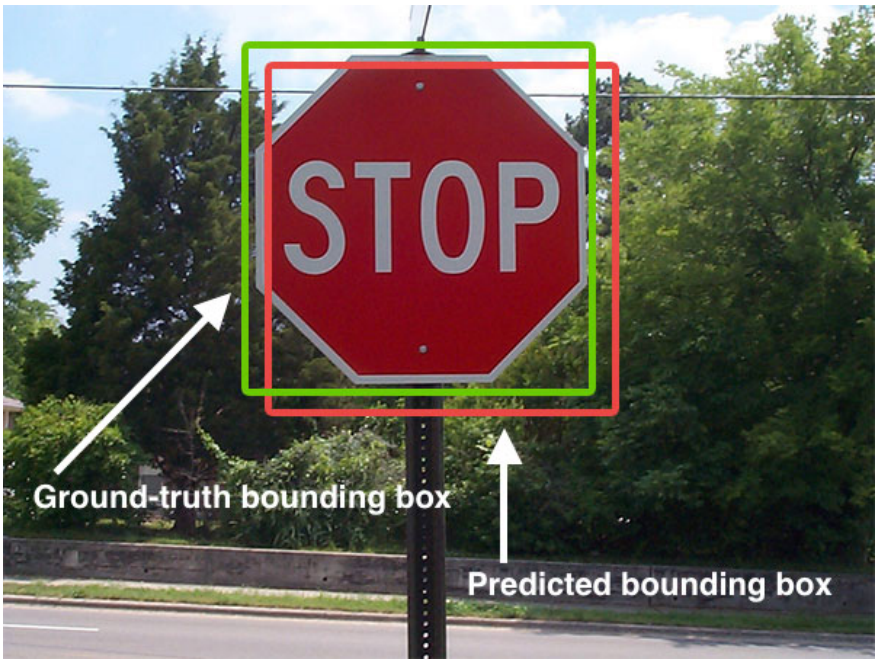
geradas pelo detector. O índice de Jaccard (JACCARD, 1901), mais conhecido como Interseção sobre União (IoU), do inglês *Intersection over Union*, calcula a interseção da *bounding box* detectada com o *ground truth* da imagem, em outras palavras, o IoU avalia o encaixe da *bounding box* com a sua respectiva *ground truth*. A Figura 24 demonstra uma predição e seu respectivo *ground truth*.

A Interseção sobre União é bastante utilizada em tarefas de detecção de imagens, como exemplo, a competição PASCAL VOC (EVERINGHAM et al., 2010) define e utiliza a métrica IoU como uma das métricas para avaliação dos resultados. Nesse caso, a Interseção sobre União é chamada de *Bounding Box Evaluation*. A Equação 4.7 descreve o cálculo da métrica IoU, onde, B_p é a *bounding box* prevista/detectada e B_{gt} é o *ground truth*. A Figura 25 demonstra de forma visual a Equação 4.7.

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{4.7}$$

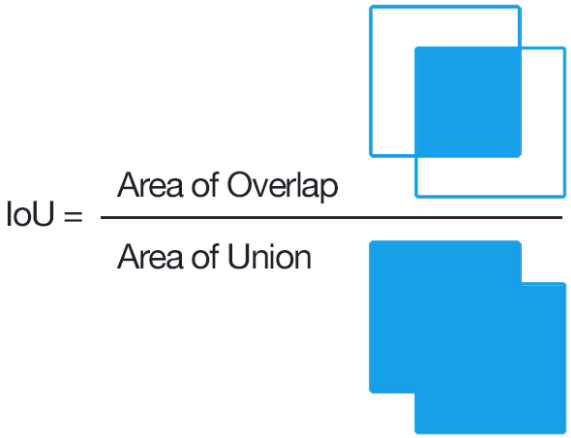
A Equação 4.7 e a Figura 25 demonstram que o valor de IoU é a razão entre a interseção de B_p e B_{gt} pela união de B_p e B_{gt}. O cálculo da métrica IoU pode ser realizado por qualquer detector de objetos, desde que as imagens possuam seus respectivos *ground truths* e que o detector retorne uma *bounding box*. Vale lembrar que o valor de IoU é um número entre 0 e 1, e que Everingham et al. (2010) definem um limiar de 50%, ou seja, para uma detecção ser validada, ela deve ter o mesmo *label* do *ground truth* e um valor de $IoU \geqslant 0.5$.

Figura 24 – Exemplo de uma detecção de placa de trânsito, onde o *bounding box* vermelho representa a predição e o *bounding box* verde representa o *ground truth*.



Fonte: (ROSEBROCK, 2016).

Figura 25 – Representação visual da métrica IoU.



Fonte: (ROSEBROCK, 2016).

5 Resultados e Discussões

Neste capítulo são descritos e discutidos os resultados obtidos através da metodologia proposta para a tarefa de detecção de armas de fogo em imagens. Este capítulo está dividido em duas seções, sendo a primeira responsável por descrever as etapas realizadas ao longo deste trabalho, assim como apresentar os resultados obtidos através da metodologia proposta no Capítulo 4. Logo a seguir, a segunda seção avalia e discute os resultados obtidos das propostas.

5.1 Detecção de Armas De Fogo

Esta seção descreve os procedimentos realizados baseados na metodologia proposta com o objetivo de viabilizar sua reprodutibilidade.

Diante disso, o presente trabalho utiliza o conceito de *containers*, abstraído pelo *software Docker* (MERKEL, 2014), responsável pela abstração e virtualização do sistema operacional, além de encapsular o *Darknet*[1] (*framework*) juntamente com o detector de objetos *YOLOv2*. A máquina utilizada no desenvolvimento deste trabalho apresenta as seguintes configurações: sistema operacional Linux, distribuição Ubuntu 18.04 LTS; processador Intel i7-8700K @3,70GHz; 32GB DDR4 de memória RAM; e uma placa de vídeo Nvidia Titan V, com 12GB de memória dedicada.

Ao longo desta seção, dois treinamentos foram realizados, ambos utilizando as mesmas configurações supracitadas. O primeiro treinamento, apresentado na Seção 5.1.1, foi realizado com o conjunto de dados DB-2, que contém apenas armas e suas respectivas anotações, conforme descrita na Seção 4.1. O segundo treinamento, apresentado na Seção 5.1.2, foi realizado com o conjunto de dados composto por dois conjuntos de dados: DB-1 e DB-2. Este treinamento necessitou de algumas etapas iniciais, uma vez que o conjunto de dados DB-1 não contém anotações. Tais etapas são explicadas ao longo da Seção 5.1.2. Os testes realizados em ambos os treinamentos foram realizados com a base de teste DB-T.

Vale ressaltar que a finalidade do primeiro treinamento (5.1.1) é uma das propostas deste trabalho, de forma a utilizar a metodologia proposta (Capítulo 4) para detectar armas de fogo em imagens, utilizando o detector de objetos *YOLOv2* juntamente com a arquitetura `yolov2`. O segundo treinamento (5.1.2) faz parte de uma outra proposta deste trabalho, utilizar o detector de objetos *YOLOv2* para marcar e aprimorar os resultados da proposta. Dessa forma, etapas inicias são efetuadas, tanto no detector como no conjunto de dados, para aprimorar os resultados da proposta.

[1] https://github.com/loretoparisi/docker/tree/master/darknet

Alguns pontos importantes devem ser destacados, o *framework Darknet* de Redmon (2013–2016) não apresenta uma interface para desenvolvimento, assim como não há uma documentação sobre a utilização do *framework*. Muitas das informações obtidas sobre a utilização do detector de objetos se dão por alguns exemplos dispostos no site oficial e pelas informações obtidas na central de discussão do repositório, onde o *framework* está hospedado. Para suprir ou amenizar esta falta de informação e de ferramentas que contribuem para a utilização do detector de objetos *YOLOv2*, algumas tarefas foram realizadas utilizando *Python*: o processo de divisão aleatória do conjunto de dados; a criação dos arquivos de configuração; e os códigos para mensurar e avaliar o detector de objetos. Todos estes códigos desenvolvidos para este trabalho estão disponibilizados em um repositório público[2].

5.1.1 Conjunto de Dados de Armas

A metodologia proposta neste trabalho, descrita no Capítulo 4, apresenta três pontos principais: o conjunto de dados, o detector de objetos e uma avaliação dos resultados.

A primeira etapa desta abordagem visa utilizar o conjunto de dados DB-2, descrito na Seção 4.1, pois é um conjunto de dados apenas com armas e apresenta suas respectivas *bounding boxes*. O objetivo de utilizar um conjunto de dados com apenas um tipo de objetos (armas) é simples, verificar se o *YOLOv2* poderia ser utilizado para detectar armas utilizando os mesmos conjuntos de dados de Olmos et al. (2018). Além disso, o conjunto de dados DB-2 foi dividido aleatoriamente em 80% para treino e 20% para validação, o que representa 2400 imagens para treino e 600 imagens para validação. Para teste, foi utilizado a base de testes DB-T, contendo 608 imagens, 304 imagens de armas e 304 imagens de não armas.

Após a divisão do conjunto de dados e seguindo a metodologia proposta, foi necessário configurar o detector de objetos para o processo de treinamento e validação. Utilizando um *container* contendo o detector de objetos *YOLOv2*, a arquitetura `yolov2` foi inserida, conforme a Tabela 2 mostrada na Seção 4.2, assim como os hiperparâmetros. Neste trabalho foram utilizados os mesmos hiperparâmetros para o pré-treinamento que Redmon e Farhadi (2017) e alguns trabalhos que utilizam o *YOLOv2* (BARRETO, 2018; WU et al., 2019; LOEY et al., 2021): *learning rate* de $0,001$; *momentum* de $0,9$; *decay* de $0,0005$; e um *batch size* de 64 imagens. Além disso, as âncoras foram ajustadas utilizando as anotações do conjunto de treinamento e o algoritmo *k-means clustering*.

A segunda etapa da metodologia é responsável por treinar e validar o modelo proposto. Porém, antes do processo de treinamento começar de fato, algumas configurações foram realizadas. Toda a utilização do detector de objetos *YOLOv2* é realizada via terminal, dessa forma, alguns arquivos de texto são utilizados para apontar os conjuntos

[2] https://github.com/GuiCardosooo/yolo-gun-detection

de dados, os arquivos referentes à arquitetura utilizada e sobre a utilização de possíveis pesos pré-treinados.

O detector de objetos *YOLOv2* gera *checkpoints* seguindo um padrão próprio: *checkpoints* são salvos a cada 100 épocas até a época de número 1000, depois disso, um *checkpoint* é salvo a cada 10000 épocas. Essa limitação pode ser compreendida como uma forma de segurança feita por Redmon (2013–2016), uma vez que os arquivos gerados ocupam um espaço considerável de memória (utilizando a arquitetura `yolov2` os arquivos ocupam cerca de 200MB). Devido ao detector de objetos *YOLOv2* não apresentar um critério de parada bem definido (uma maneira de parar o treinamento é apresentada em uma das discussões em um repositório de um dos mantenedores, porém, não leva em consideração nenhuma métrica, apenas a quantidade máxima de *batches*[3]), toda e qualquer pausa é feita pelo usuário, sendo possível reiniciar o treinamento de qualquer *checkpoint* gerado. Dito isso, o processo de treinamento durou cerca de 3 horas, sendo interrompido pelo autor quando o processo apresentou um erro constante via terminal.

Todos os arquivos gerados são de suma importância, visto que o processo de validação é feito a partir destes arquivos. Basicamente, os *checkpoints* são avaliados sobre o conjunto de validação, visando selecionar os melhores resultados para o processo de treinamento. O processo de validação, disposto no detector de objetos *YOLOv2*, avalia as detecções utilizando duas métricas: *IoU* e *Recall* (ou Sensibilidade), ambas descritas na Seção 4.3. Além disso, a média harmônica ($\overline{h}$) entre as duas métricas apresentadas, *IoU* e *Recall* (*Se*), foi calculada para auxiliar a avaliação dos *checkpoints*. A Tabela 3 apresenta os resultados obtidos no processo de validação ao longo dos *checkpoints*.

Baseado no valores obtidos através da média harmônica no processo de validação, foi selecionado o melhor *checkpoint*. Quando há métricas com relação inversa, neste caso, *IoU* e a *Sensibilidade*, a melhor média para ser usada é a média harmônica. A média harmônica foi usada como referência por ser um número único que indica a qualidade do modelo, nesse caso, a qualidade dos *checkpoints*. Com isso, foi selecionado o arquivo com a maior média harmônica ($\overline{h}$): *checkpoint* #23 (época = 140000).

Em seguida, a terceira e última etapa da metodologia é iniciada: testes e avaliação dos resultados. Toda a etapa de testes e avaliação foi feita com a base de teste (DB-T), onde foram avaliados 3 valores de confiança de saída da rede. Os três valores definidos foram usados para representar cenários onde o modelo seja capaz de detectar com um alto, médio e baixo valor de confiança. Neste caso foram usados os valores de 25%, 50% e 75%. A Tabela 4 apresenta os resultados obtidos com os 3 valores de confiança sendo comparados com trabalhos encontrados na literatura sobre a mesma base de teste (DB-T).

Sabe-se que a precisão (Pr) avalia a capacidade do modelo em classificar correta-

[3] https://github.com/AlexeyAB/Yolo_mark/issues/11

Tabela 3 – Resultados obtidos no processo de validação utilizando os arquivos gerados (*checkpoints*) no processo de treinamento (o processo de validação foi efetuado com 20% do conjunto de dados DB-2). A coluna $\overline{h}$ destaca o maior valor de média harmônica (#23).

#	época	*IoU* (%)	*Se* (%)	$\overline{h}$ (%)	#	época	*IoU* (%)	*Se* (%)	$\overline{h}$ (%)
01	100	44,57	32,95	37,89	14	50000	54,25	59,09	56,57
02	200	48,03	42,05	44,84	15	60000	54,94	62,50	58,48
03	300	46,63	33,38	38,91	16	70000	59,68	72,30	65,39
04	400	47,37	37,78	42,03	17	80000	60,86	76,85	67,93
05	500	47,47	37,07	41,63	18	90000	60,89	77,56	68,22
06	600	44,38	28,55	34,75	19	100000	62,37	79,97	70,08
07	700	45,47	29,83	36,03	20	110000	62,04	74,29	67,61
08	800	44,19	25,43	32,28	21	120000	64,94	78,98	71,28
09	900	42,99	25,28	31,84	22	130000	58,83	72,87	65,10
10	10000	47,25	41,05	43,93	23	140000	63,78	82,67	**72,01**
11	20000	58,01	70,17	63,51	24	150000	61,49	76,70	68,26
12	30000	55,51	66,19	60,38	25	160000	63,21	79,97	70,61
13	40000	60,20	75,71	67,07					

Tabela 4 – Comparação entre o modelo proposto e modelos encontrados na literatura, utilizando o detector de objetos *YOLOv2* treinado e validado com um conjunto de dados de armas (DB-2). Todos os resultados aqui apresentados foram obtidos sobre a mesma base de teste: DB-T.

Modelo	Conf. (%)	VP	FN	VN	FP	*Pr* (%)	*Se* (%)	F_1 (%)
Fast R-CNN: (ELMIR et al., 2019)	-	232	72	248	56	80,76	76,31	78,37
Faster R-CNN: (OLMOS et al., 2018)	-	304	0	247	57	84,21	**100,00**	91,43
Detector de Objetos: *YOLOv2*	25	302	2	144	160	65,37	99,34	78,85
	50	296	8	234	70	80,87	97,37	88,36
	75	281	23	285	19	**93,67**	92,43	**93,05**

mente detecções que correspondem a amostras positivas, e que a sensibilidade (*Se*) avalia a capacidade do modelo em detectar todas as amostras positivas em um conjunto de dados. Dito isso, pode-se observar padrões na Tabela 4, onde valores maiores de confiança resultam em valores de precisão (*Pr*) superiores aos valores de sensibilidade (*Se*), e de forma inversa, valores menores de confiança resultam em valores de sensibilidade (*Se*) superiores, ou bem próximos, aos valores de precisão (*Pr*). Vale ressaltar que a métrica F_1 apresenta um número único, que indica a qualidade geral do modelo, em outras palavras, a média harmônica entre a precisão (*Pr*) e a sensibilidade (*Se*).

De imediato, pode-se destacar que o modelo *Fast R-CNN*, utilizado por Elmir et al. (2019), não alcançou resultados expressivos em relação ao modelo *Faster R-CNN*, utilizado

anteriormente por Olmos et al. (2018), mas mesmo assim, apresenta uma quantidade similar de falsos positivos. Por sua vez, o modelo *Faster R-CNN* apresenta uma redução significativa de falsos negativos em relação ao modelo *Fast R-CNN*, alcançando uma sensibilidade (Se) de 100%. Porém, a quantidade considerável de falsos positivos impacta diretamente na precisão (Pr) e, em contrapartida, na métrica F_1.

Os resultados obtidos com o detector de objetos *YOLOv2*, utilizando o banco DB-2 para treino e validação, e sendo avaliado sobre a base de teste DB-T, apresenta alguns pontos importantes. Inicialmente, é possível observar que com uma confiança de 75%, a métrica F_1 é superior aos modelos apresentados. Mesmo não alcançando uma sensibilidade (Se) igual a de Olmos et al. (2018), o modelo proposto neste trabalho compensa com o aumento da precisão (Pr), sendo explicado pela redução de falsos positivos. As duas últimas linhas apresentam uma redução de falsos positivos, mas apresentam uma quantidade considerável de falsos negativos.

Os resultados apresentados na primeira abordagem, utilizando a arquitetura `yolov2`, uma arquitetura baseada na *Darknet19* de Redmon e Farhadi (2017), apresenta uma melhora na tarefa de detecção de armas de fogo, onde foi possível reduzir a quantidade de falsos positivos. Porém, com apenas uma classe, o detector de objetos *YOLOv2* não foi capaz de generalizar e aprender totalmente, sendo observado pela quantidade de falsos negativos. De maneira geral, o modelo proposto foi superior ao *Faster R-CNN* e ao *Fast R-CNN*, com uma confiança de 75%, alcançou uma precisão (Pr) de $93,67\%$ e uma sensibilidade (Se) de $92,43\%$, resultando em um F_1 de $93,05\%$. Vale ressaltar que o tempo de detecção médio da base de teste DB-T foi de $0,0112$ segundos ($\cong$ 90 *fps*), considerando imagens de tamanho fixo de 160×120 (conforme descrito na Seção 4.1).

5.1.2 Conjunto de Dados com Armas e Não Armas

O conjunto de dados utilizada na primeira abordagem (DB-2) é disposta de suas respectivas *bounding boxes*, ou seja, arquivos de texto contendo informações referentes à localização das armas nas respectivas imagens. Um conjunto dados relativamente grande, contendo 3000 imagens, mas que apresenta apenas uma classe: "*pistol*". Para a segunda abordagem, é proposto uma etapa inicial, visando utilizar o detector de objetos (*YOLOv2*) e o conjunto de dados DB-1 (conjunto de dados sem anotações criada para avaliar uma das propostas de (OLMOS et al., 2018)) para compor uma novo conjunto, juntamente com a conjunto de dados DB-2.

A etapa inicial, uma das propostas deste trabalho, visa utilizar o detector de objetos *YOLOv2* para marcação de base dados. Conforme descrito na Seção 4.2, o *framework Darknet* possibilita a alteração e modificação de seus arquivos fontes, uma vez que ele é *open source*. Dessa maneira, foi realizada uma modificação na saída do detector de objetos *YOLOv2* pós-detecção. A saída padrão é composta pela configuração da arquitetura

utilizada, o tempo de detecção e as detecções, sendo elas apresentadas pela classe seguida de sua respectiva confiança, como mostra a Figura 26a. A alteração feita no código fonte focou na representação das detecções, modificando a saída padrão para uma saída mais detalhada, apresentando de maneira numérica as classes, as informações referentes à localização do objeto detectado e, por fim, o valor de confiança, como mostra a Figura 26b. Essa modificação foi realizada para averiguar as métricas presentes neste trabalho e para auxiliar em uma abordagem de marcação de imagens com o detector de objetos.

Figura 26 – Exemplo de uma saída padrão (a) e uma saída modificada (b) utilizando o detector de objetos *YOLOv2*.

(a)

```
   30 conv    425  1 x 1 / 1    19 x  19 x1024   →    19 x  19 x 425
   31 detection
mask_scale: Using default '1.000000'
Loading weights from yolov2.weights ... Done!
data/dog.jpg: Predicted in 0.011568 seconds.
dog: 82%
truck: 64%
bicycle: 85%
```

(b)

```
   30 conv    425  1 x 1 / 1    19 x  19 x1024   →    19 x  19 x 425
   31 detection
mask_scale: Using default '1.000000'
Loading weights from /darknet/simples/yolov2.weights ... Done!
data/dog.jpg: Predicted in 0.011572 seconds.
16 0.295195 0.653756 0.246263 0.501944 82
7 0.747563 0.218985 0.276749 0.145329 64
1 0.445103 0.496250 0.641791 0.561731 85
```

Fonte: o Autor.

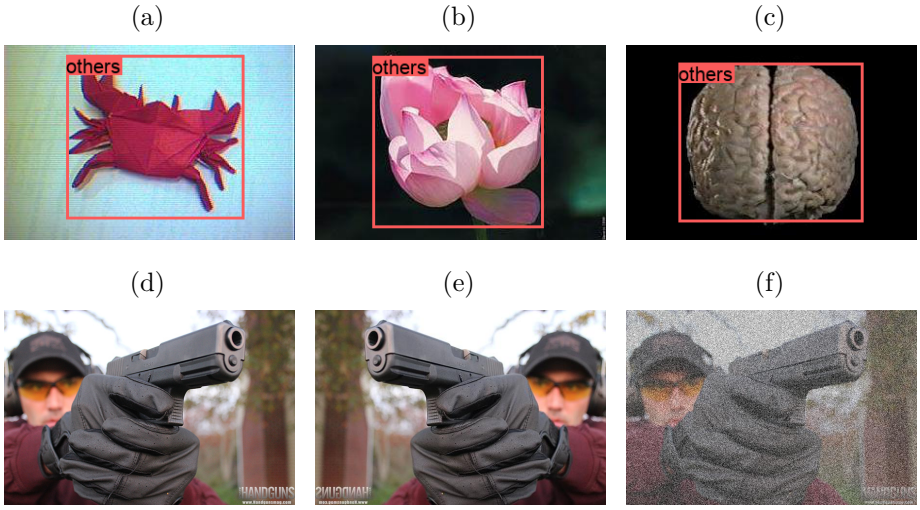
Dessa maneira, a modificação na saída padrão permitiu que uma nova abordagem fosse usada com conjunto de dados sem *bounding boxes*. Utilizando pesos pré-treinados sobre o conjunto de dados COCO *Dataset* (LIN et al., 2014) e utilizando o detector de objetos *YOLOv2*, foi feita uma detecção dos objetos contidos no conjunto de dados DB-1. Alguns exemplos podem ser observados nas Figuras 27a - 27c. Conforme descrito na Tabela 1, o conjunto de dados DB-1 não apresenta nenhuma marcação e/ou anotação referente aos objetos contidos nas imagens. Vale ressaltar que todas as 795 imagens com armas do conjunto de dados DB-1 foram retiradas neste processo (imagens de baixa resolução). Dessa forma, tal abordagem foi uma maneira de detectar e marcar automaticamente 9062 imagens.

Com o auxílio de um algoritmo em *Python*, todas as 80 possíveis classes detectadas (pois os pesos pré-treinados são sobre o conjunto de dados COCO *Dataset*, a qual possui 80 classes) foram convertidas para apenas uma classe: "*others*" (em português, outros).

Uma inspeção visual das marcações foi realizada utilizando um *software*[4] para validar as marcações. O conjunto de dados DB-1 marcado (Doravante: DB-1*) irá compor um novo conjunto de dados, contribuindo com amostras negativas para o detector de objetos.

Duas técnicas de *data augmentation* foram utilizadas no conjunto de dados DB-2 para tentar igualar o número do conjunto de dados DB-1*. As Figuras 27d - 27f apresentam alguns exemplos da técnica de *data augmentation* (*flip vertical* e ruído Gaussiano). O ruído Gaussiano foi aplicado para cada imagem com desvio padrão igual a média da respectiva imagem e com média igual a 0.

Figura 27 – (a - c) - Exemplos de imagens marcadas utilizando a abordagem inicial; (d) - Exemplo de uma imagem do conjunto de dados DB-2; (e) - *flip vertical* de *d*; e (f) - ruído Gaussiano de *d*.



Fonte: o Autor.

As duas técnicas de *data augmentation* resultaram em um aumento na quantidade de imagens, triplicando o tamanho do conjunto de dados DB-2, saindo de 3000 imagens e chegando a 9000 imagens (Doravante: DB-2*), gerando um equilíbrio entre as amostras positivas (classe "*pistol*") e as amostras negativas (classe "*othres*"). Dessa forma, foi formado um novo conjunto de dados, composta pelos conjuntos DB-1* e DB-2*, sendo nomeada DB-3 (DB-3 = DB-1* + DB-2*), totalizando 18062 imagens com suas devidas *bounding boxes* (anotações).

Toda essa etapa inicial foi realizada para ser usada como entrada na primeira etapa da metodologia: o conjunto de dados. Assim como na primeira abordagem, o conjunto de dados DB-3 foi dividido aleatoriamente em 80% para treino e 20% para validação, o que representa 14449 imagens para treino (7200 armas e 7249 não armas) e 3613 imagens para validação (1800 armas e 1813 não armas). Para testes e avaliação dos resultados, foi utilizado a base de teste DB-T.

[4] https://github.com/tzutalin/labelImg

Em seguida, a segunda etapa foi realizada com a configuração dos hiperparâmetros. De maneira geral, toda segunda abordagem foi idêntica à primeira, com exceção do conjunto de dados utilizado para treinamento e validação. Logo, a divisão do conjunto de dados e as configurações do detector de objetos permaneceram idênticas. Sendo assim, o detector de objetos foi configurado com um *learning rate* de $0,001$, um *momentum* de $0,9$, um *decay* de $0,0005$ e um *batch size* de 64 imagens para a etapa de treinamento.

Assim como explicado, devido ao detector de objetos *YOLOv2* não apresentar um critério de parada bem definido, o detector de objetos gera *checkpoints* seguindo um padrão próprio, mas que possibilita a pausa no treinamento seja ela proposital ou não, e possibilita reiniciar o treinamento de qualquer *checkpoint* gerado. Dito isso, e baseado na única saída no detector de objetos *YOLOv2* referente ao treinamento, o processo de treinamento durou cerca de 5 horas, sendo interrompido pelo autor quando o processo apresentou um erro constante via terminal.

Depois do processo de treinamento, os *checkpoints* gerados foram submetidos ao processo de validação disposto no detector de objetos *YOLOv2*. Esse processo avalia os *checkpoints* sobre duas métricas, *IoU* e *Recall* (ou Sensibilidade) e, ainda, a média harmônica ($\overline{h}$) entre as duas métricas apresentadas (*IoU* e *Recall (Se)*) foi calculada para auxiliar a avaliação dos *checkpoints*. A Tabela 5 apresenta os resultados obtidos no processo de validação.

Tabela 5 – Resultados obtidos no processo de validação utilizando os arquivos gerados (*checkpoints*) no processo de treinamento (O processo de validação foi efetuado com 20% do conjunto de dados DB-3). A coluna $\overline{h}$ destaca os quatro maiores valores de média harmônica.

#	época	*IoU* (%)	Se (%)	$\overline{h}$ (%)	#	época	*IoU* (%)	Se (%)	$\overline{h}$ (%)
01	100	46,69	35,73	40,48	14	50000	71,11	87,79	78,57
02	200	47,76	37,95	42,29	15	60000	70,32	86,07	77,40
03	300	47,47	36,84	41,48	16	70000	70,83	87,00	78,09
04	400	47,65	36,94	41,62	17	80000	71,03	87,20	78,29
05	500	44,99	28,19	34,66	18	90000	71,96	88,82	**79,51**
06	600	47,93	39,76	43,46	19	100000	70,09	86,66	77,50
07	700	47,63	39,67	43,29	20	110000	70,50	86,66	77,75
08	800	48,23	42,66	45,27	21	120000	71,68	88,18	**79,08**
09	900	48,95	43,94	46,31	22	130000	71,96	88,23	**79,27**
10	10000	64,45	81,64	72,03	23	140000	70,69	87,96	78,39
11	20000	69,11	85,82	76,56	24	150000	71,81	87,59	78,92
12	30000	70,33	86,73	77,67	25	160000	72,10	88,13	**79,31**
13	40000	70,29	86,29	77,47	26	170000	69,33	86,24	76,87

Diferente da primeira abordagem, ao invés de selecionar apenas uma época com a maior média harmônica, quatro épocas foram selecionadas, baseadas em sua média harmônica ($\overline{h}$). Com isso, os quatro arquivos com maior $\overline{h}$ foram: *checkpoint* #18 (época =

90000), *checkpoint* #25 (época = 160000), *checkpoint* #22 (época = 130000) e *checkpoint* #21 (época = 120000), respectivamente. Os quatro *checkpoints* apresentam um $\overline{h}$ superior a 79%, assim como valores de $IoU \geqslant 71\%$ e $Se \geqslant 88\%$.

O conjunto de validação é composto por 3613 imagens, sendo 1800 imagens de armas e 1813 de não armas, todas com as suas devidas anotações (bounding boxes). Com isso, para selecionar a melhor época para avaliação, a curva ROC foi calculada sobre o conjunto de validação para cada uma das quatro épocas e, ainda, foram definidos 3 valores de limiar para IoU para auxiliar na seleção da melhor época e do melhor ponto (confiança). A Figura 28 apresenta as curvas ROC das quatro épocas selecionadas.

Visualmente, todas as figuras são semelhantes e apresentam resultados próximos ao ponto ótimo de uma curva ROC (`0,100%`), descrito na Seção 4.3.2. Dessa forma, a área sobre a curva (AUC) foi utilizada para selecionar a melhor época e, assim, calcular o melhor valor de confiança. Todas as curvas ROC foram criadas sobre o conjunto de validação DB-3 e apresentam, em sua maioria, áreas superiores $0,9$. As épocas 130000 e 160000 apresentam resultados semelhantes, porém, na média, a época 160000 apresenta a maior área sobre a curva.

Baseado na variação da confiança, a distância Euclidiana bidimensional entre o ponto ótimo (`0,100%`) e o melhor ponto de cada curva (`TFP,TVP`) foi calculada, resultando no melhor valor de confiança para com o conjunto de validação DB-3. Dessa forma, os valores de confiança, obtidos para cada um dos limites de IoU, foram utilizados para avaliar o detector de objetos *YOLOv2*, utilizando a arquitetura `yolov2`, na base de testes DB-T. A Tabela 6 apresenta os resultados obtidos com o melhor ponto de cada uma das curvas (IoU) referente a época 160000, sendo comparados com trabalhos encontrados na literatura sobre a mesma base de teste (DB-T).

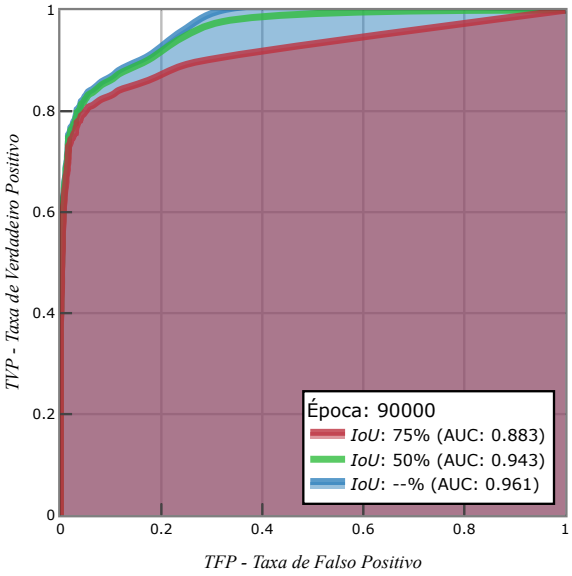
Tabela 6 – Comparação entre o modelo proposto, utilizando o detector de objetos *YOLOv2* com um conjunto de dados de armas e não armas (DB-3), e modelos encontrados na literatura. Todos os resultados aqui apresentados foram obtidos sobre a mesma base de teste: DB-T.

Modelo	Conf, (%)	IoU (%)	VP	FN	VN	FP	Pr (%)	Se (%)	F_1 (%)
Fast R-CNN: (ELMIR et al., 2019)	-	-	232	72	248	56	80,76	76,31	78,37
Faster R-CNN: (OLMOS et al., 2018)	-	-	304	0	247	57	84,21	**100,00**	91,43
Detector de Objetos: *YOLOv2*	58	-	303	1	270	34	**89,91**	99,67	**94,54**
	57	50	290	14	270	34	**89,51**	95,39	**92,36**
	52	75	290	18	268	36	**88,96**	95,39	**92,06**

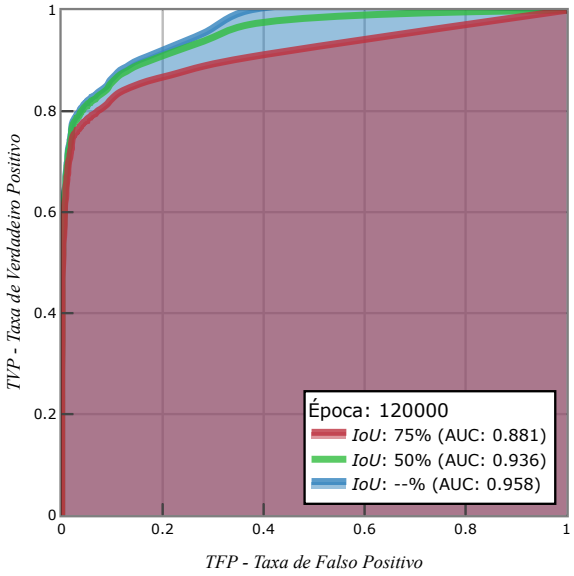
Assim como observado na Subseção 5.1.1, o modelo *Fast R-CNN* não alcançou

Figura 28 – Curvas ROC referente às épocas selecionadas, calculadas sobre o conjunto de validação de DB-3. Cada figura apresenta 3 curvas ROC para uma mesma época, com 3 valores de limiar para IoU: 0% (curva azul); 50% (curva verde); e 75% (curva vermelha). Todas as curvas apresentam o seu respectivo valor da área sobre a curva (AUC).

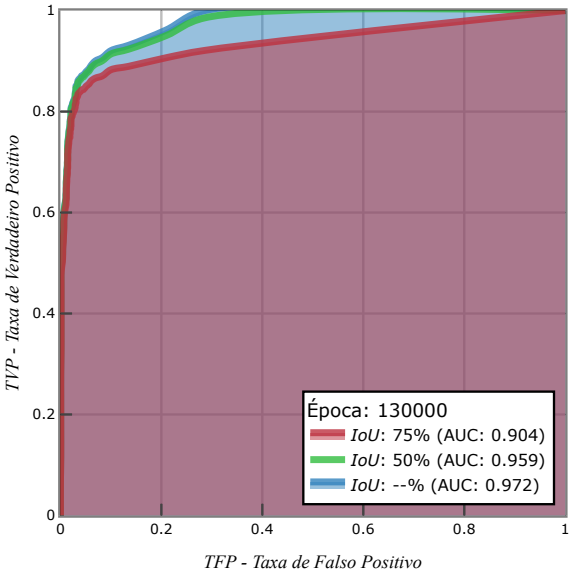
(a) Época = 90000



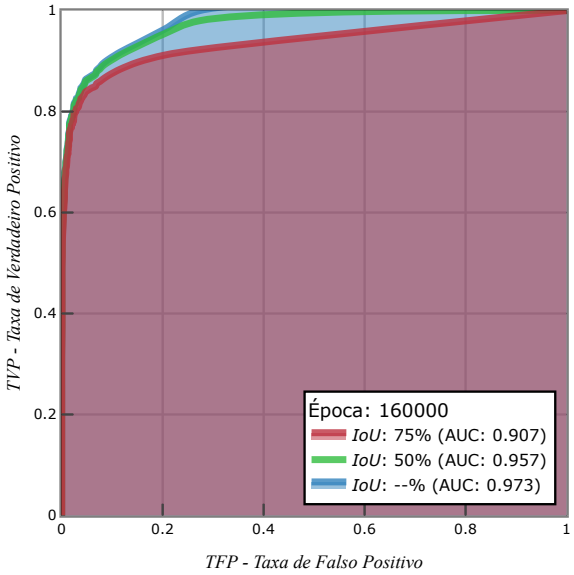
(b) Época = 120000



(c) Época = 130000



(d) Época = 160000



Fonte: o Autor.

resultados expressivos em relação ao modelo *Faster R-CNN*, mas apresenta uma quantidade similar de falsos positivos. Por sua vez, o modelo implementado por Olmos et al. (2018) para detecção de armas de fogo não apresenta nenhum falso negativo, resultando em uma sensibilidade (Se) de 100%, diferentemente do modelo implementado por Elmir et al. (2019). Porém, uma quantidade considerável de armas foram detectadas erroneamente, resultando

em 57 falsos positivos, impactando diretamente na precisão (Pr) e, em contrapartida, na métrica F_1. Vale lembrar que os trabalhos de Elmir et al. (2019) e Olmos et al. (2018) não utilizam a métrica IoU em sua avaliação sobre a base de teste DB-T, assim como não informam os valores de confiança para com a base de teste.

Importante destacar que o objetivo proposto deste trabalho é desenvolver um método computacional que envolva redes neurais convolucionais profundas para detecção de armas de fogo em imagens. Dito isso é importante levantar que a precisão e a sensibilidade apresentam suas importâncias, mas com objetivos distintos. Por exemplo, em alguns casos é mais importante ter certeza que uma arma foi detectada, com o risco de deixar armas sem detecção (um modelo preciso), em outros casos é mais importante tentar detectar o maior número possível de armas, mesmo que algumas detecções sejam falsas (modelo sensível).

Os resultados obtidos com o detector de objetos *YOLOv2*, utilizando o conjunto de dados DB-3 para treino e validação, apresentam uma melhora em relação aos trabalhos apresentados. Em uma comparação direta, o modelo configurado com uma confiança de 58 e nenhum valor para IoU apresenta uma redução do número de falsos positivos, ou seja, armas detectadas erroneamente, quando comparado aos outros valores de confiança. Tal redução é expressa pelo aumento da precisão ($Pr = 89,91\%$), porém, 1 arma não foi detectada, o que resultou em uma sensibilidade inferior ao modelo implementado por Olmos et al. (2018) ($Se = 99,67\%$). Os valores de limiar para IoU, resultaram em detecções mais precisas para ambos os casos (50% e 75%), em contrapartida, o refinamento das detecções tornou o modelo um pouco menos sensível em relação ao limiar de IoU igual a 0. Os três valores de IoU apresentam resultados mais precisos e detecções mais sensíveis, o que impactou diretamente na métrica F_1, onde todas as configurações resultaram em valores superiores aos encontrados na literatura.

Todos os três resultados apresentados na segunda abordagem, utilizando o conjunto de dados DB-3, foram selecionados com a ajuda da curva ROC e da obtenção do melhor ponto para cada limiar de IoU, onde todos os três resultados apresentam uma considerável melhora na tarefa de detecções de armas de fogo. A segunda abordagem deste trabalho apresentou uma nova etapa inicial, a qual, utiliza o próprio detector de objetos para marcar um conjunto de dados de não armas e compor um novo conjunto de dados com amostras negativas. Isto possibilitou uma melhora na generalização do modelo, podendo ser observada pela redução de falsos positivos e impactando diretamente no aumento da precisão. Em resumo, o detector de objetos *YOLOv2*, com a arquitetura `yolov2`, proporcionou resultados superiores aos detectores implementados com o *Faster R-CNN* e ao *Fast R-CNN*, apresentando valores de confiança relativamente altos, na qual, a melhor configuração (valor de confiança igual a 57% e nenhum limite para IoU) apresentou uma precisão de $89,91\%$ e uma sensibilidade de $99,67\%$, resultando em um F_1 de $94,54\%$. Vale

ressaltar que o tempo de detecção médio da base de teste DB-T foi de $0,0111$ segundos ($\cong 90$ *fps*), considerando imagens de tamanho fixo de 160×120 (conforme descrito na Seção 4.1).

5.2 Análise dos Resultados

Na Seção 5.1, dois treinamentos foram realizados, um conjunto de dados de armas (DB-2) e outro conjunto de dados de armas e não armas (DB-3), ambos focados em um principal objetivo: desenvolver um método computacional que envolva redes neurais convolucionais profundas para detecção de armas de fogo em imagens, levando em consideração a confiabilidade e a velocidade de detecção. Para tal, uma metodologia foi desenvolvida (4) e avaliada em duas abordagens. A primeira avalia a metodologia deste trabalho, juntamente com o detector de objetos proposto (4.2), onde todo o processo de treinamento, validação e testes foi realizado com os conjuntos de dados apresentados na Seção 4.1. A segunda propõe uma etapa inicial à metodologia proposta neste trabalho, focada em modificar o código fonte e utilizar o detector de objetos proposto para marcar e gerar anotações (*bouding boxes*) em um conjunto grande de imagens, para aprimorar a metodologia proposta. Os testes realizados em ambas as abordagens, utilizou-se o mesmo conjunto de dados (DB-T), assim como feito em Olmos et al. (2018) e Elmir et al. (2019).

Olmos et al. (2018), os criadores do conjunto de dados utilizado ao longo deste trabalho, obtiveram uma precisão de $84,21\%$ e uma sensibilidade de $100,00\%$, utilizando o algoritmo *Faster R-CNN*. Tal implementação foi capaz de detectar todas as 304 armas do banco de teste, em contrapartida, o algoritmo apresentou um elevado número de falsos positivos ($FP = 57$), o que impactou diretamente na precisão do modelo, assim como na métrica F_1 ($F_1 = 91,43\%$).

Elmir et al. (2019) utilizou o antecessor do algoritmo utilizado por Olmos et al. (2018), o *Fast R-CNN*. O modelo proposto por Elmir et al. (2019) não apresentou resultados expressivos, mas alcançou uma precisão semelhante ($Pr = 80,76$) ao trabalho de Olmos et al. (2018) com um número ligeiramente menor de falsos positivos ($FP = 56$). O *Fast R-CNN* não apresentou uma sensibilidade significativa, pois não detectou 72 armas (FP), resultando em uma sensibilidade de $76,31\%$, resultando em um $F_1 = 78,37\%$.

A primeira abordagem deste trabalho buscou avaliar a metodologia proposta e a eficácia do detector de objetos *YOLOv2* em detectar armas, quando comparada com os resultados obtidos na literatura e utilizando os mesmos conjuntos de dados. Tal abordagem utilizou o conjunto de dados DB-2 para treinamento e validação e obteve resultados parcialmente superiores aos encontrados na literatura, como mostra a Tabela 4 da Seção 5.1.1. Quando comparado a Olmos et al. (2018), os resultados apresentados sobre o banco de testes DB-T apontam, em um dos casos, um aumento na precisão e na métrica F_1, e

uma redução na sensibilidade.

Na etapa de validação, uma única época foi selecionada para efetuar a etapa de teste e avaliação dos resultados. Toda a etapa de teste realizada na Subseção 5.1.1 foi feita com a base de teste DB-T, onde foram avaliados 3 valores de confiança. Os três valores definidos foram usados para representar cenários onde o modelo seja capaz de detectar com um alto, médio e baixo valor de confiança. Neste caso foram usados os valores de 25%, 50% e 75%, respectivamente.

O modelo proposto na primeira abordagem apresenta resultados satisfatórios. Com um valor de confiança igual 75%, o detector de objetos apresenta uma precisão de $93,67\%$ e uma sensibilidade de $92,43\%$, resultando em um F_1 de $93,05\%$, superior ao apresentado por Olmos et al. (2018). Vale lembrar que a métrica F_1 nada mais é do que uma média harmônica entre a precisão e a sensibilidade, resumindo em apenas um número a qualidade geral do modelo. Os outros valores de confiança apresentam valores de precisão superiores aos encontrados na literatura, mas não apresentam uma qualidade em suas detecções, sendo observado pelo aumento de falsos negativos (armas não detectadas).

A primeira abordagem, utilizando o conjunto de dados DB-2, na qual apresenta apenas uma classe (*"pistol"*), apresentou resultados satisfatórios em apenas um dos três valores de confiança definidos. Foi observado que a presença de apenas uma única classe contribuiu de forma negativa, impedindo uma generalização mais efetiva e na qualidade das detecções, sendo confirmado pela quantidade de falsos negativos e falsos positivos.

A segunda abordagem utilizou a metodologia proposta no Capítulo 4, mas com uma etapa inicial diferente. Utilizando o próprio detector de objetos *YOLOv2*, juntamente com pesos pré-treinados em um conjunto de dados conhecido no mundo acadêmico (LIN et al., 2014), a etapa inicial modificou o código fonte e marcou as imagens contidas no conjunto de dados DB-1 (com exceção das 795 imagens de armas, conforme descrito na Subseção 5.1.2), gerando as informações dos objetos contidos em cada uma das imagens. Todas essas anotações (*bounding boxes*) foram renomeadas para uma mesma classe, chamada de *"others"*, atuando como amostras negativas para o treinamento. Vale lembrar que o conjunto de dados DB-2 representa um terço do conjunto de dados DB-1, sendo submetido a dois processos de *data augmentation*. Logo, esses dois conjuntos de dados, o conjunto DB-1 marcado e o conjunto DB-2 "aumentada", foram unificados, o que deu origem ao conjunto de dados intitulado de DB-3, apresentando um equilíbrio entre amostras positivas (armas) e amostras negativas (não armas).

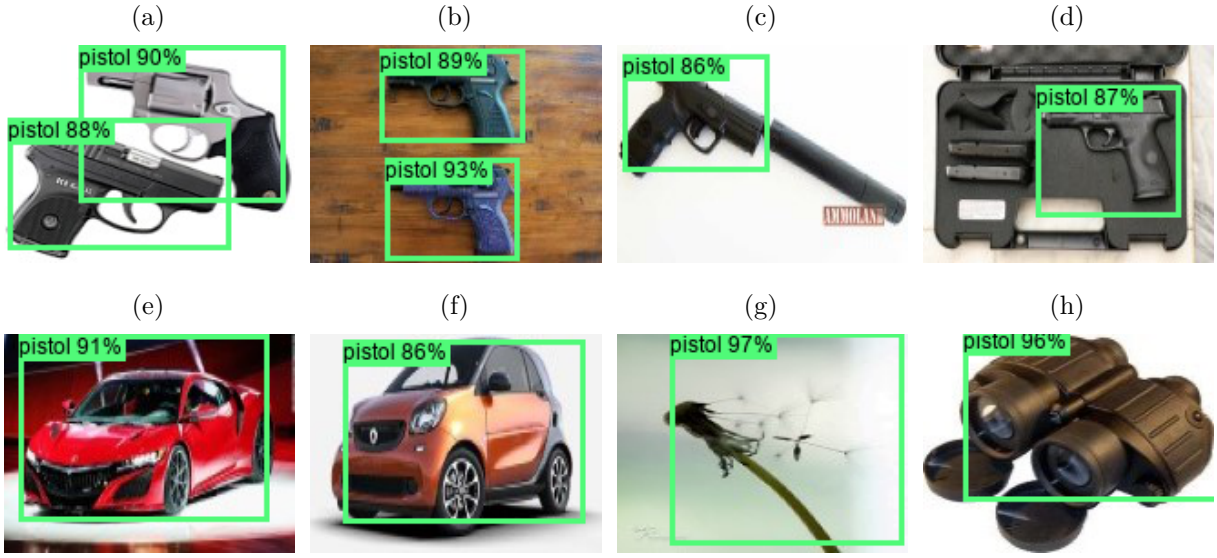
Depois dessa etapa inicial, a segunda abordagem foi iniciada, utilizando o conjunto de dados DB-3 para treinamento e validação. A etapa de validação é expressa pela Tabela 5, onde foram selecionadas 4 épocas baseadas em sua médias harmônica ($\overline{h}$). Em seguida, a curva ROC das 4 épocas foram calculadas com a adição de 3 valores de limiar para IoU, dessa forma, cada época gerou 3 curvas (0%, 50% e 75%), conforme mostra a Figura 28. A

área sobre a curva (AUC) foi utilizada para definir a melhor época dentre as 4 selecionadas e, assim, calcular o melhor valor de confiança para cada uma das 3 curvas.

O modelo proposto não foi capaz de detectar todas as armas em todos 3 valores de limiar para IoU, mas o maior número de falsos negativos em uma das 3 configurações não representa nem 6% da quantidade de armas presentes no conjunto de dados DB-T (304), podendo ser observado pelo melhor e o pior resultado obtido. O melhor resultado apresenta uma confiança de 58% e nenhum valor para IoU, onde obteve uma precisão de $89,91\%$ e uma sensibilidade de $99,67\%$, resultando em um F_1 de $94,54\%$. O pior resultado apresenta uma confiança de 52% e um valor de 75% para IoU, onde obteve uma precisão de $88,96\%$ e uma sensibilidade de $95,39\%$, resultando em um F_1 de $92,06\%$. Importante destacar que os valores de confiança representam o limiar configurado para a detecção do *YOLOv2*, ou seja, não necessariamente todas as detecções apresentam um valor igual à 58%, estes valores podem variar, como exemplo, o melhor e o pior valor de confiança da segunda abordagem podem variar a confiança entre $52 - 100\%$ e $58 - 100\%$, respectivamente.

O detector de objetos *YOLOv2* apresentou um aumento na qualidade da detecção, assim como um aumento na precisão e na sensibilidade em relação à primeira abordagem. A etapa inicial da segunda abordagem contribuiu de maneira positiva para isso, podendo ser observado pela métrica F_1, onde todos os 3 valores de limiar para IoU apresentaram resultados superiores aos encontrados na literatura. Toda a etapa inicial auxiliou e proporcionou um aprimoramento na proposta deste trabalho e, a junção de um conjunto de dados contendo apenas armas e outro banco contendo imagens variadas (amostras negativas), forneceu ao detector de objetos dados suficientes para melhorar seu desempenho em todas as métricas, na qual foi possível reduzir a quantidade de falsos negativos e a quantidade de falsos positivos. A Figura 29 ilustram alguns exemplos de imagens do banco de teste DB-T rotuladas corretamente (VP) e outras rotuladas erroneamente (FP) como armas, resultantes da segunda abordagem da segunda abordagem.

Figura 29 – Exemplos de imagens detectadas no banco de teste DB-T, resultantes da segunda abordagem; (a - d) - Exemplos de verdadeiros positivos; e (e - h) - Exemplos de falsos positivos.



Fonte: o Autor.

6 Conclusão e Trabalhos Futuros

O objetivo deste trabalho foi desenvolver um detector de armas de fogo em imagens utilizando CNN. Para tal, uma metodologia foi elaborada utilizando o detector de objetos *YOLOv2* de Redmon e Farhadi (2017), que foi avaliada em duas abordagens realizadas com os conjuntos de dados criado por Olmos et al. (2018).

Após a análise dos resultados obtidos para as duas abordagens, constatou-se que em ambos houve uma redução na quantidade de falsos positivos em relação aos trabalhos de Olmos et al. (2018) e Elmir et al. (2019), impactando diretamente no aumento da precisão do detector de objetos proposto. Vale lembrar que o presente trabalho inseriu a avaliação das detecções realizadas (IoU) e apresentou o valor de confiança das detecções, informações não fornecidas pelos trabalhos usados como comparação.

Na primeira abordagem o detector de objetos foi treinado com um conjunto de dados contendo apenas armas, e apresentou uma redução superior a 60% na quantidade de falsos positivos, proporcionando um aumento superior a 10% na precisão em relação ao *Faster R-CNN*. Em contrapartida, 23 armas não foram detectadas (falsos negativos), mas essa pequena redução na sensibilidade não impactou significativamente na métrica F_1, responsável por apresentar um único valor que indica a qualidade geral do modelo. Com um aumento de quase 2%, o detector de objetos *YOLOv2* apresentou um F_1 superior ao trabalho de Olmos et al. (2018), sendo alcançado com uma confiança de 75%, uma precisão de $93,67\%$, uma sensibilidade de $92,43\%$, resultando em um F_1 de $93,05\%$. Sendo assim, a primeira abordagem atingiu o objetivo proposto neste trabalho e superou o pioneiro na área (segundo o próprio autor Olmos et al. (2018)) de detecção de armas de fogo em imagens utilizando CNN.

Dessa forma, o detector de objetos *YOLOv2* se mostrou superior aos detectores utilizados neste mesmo conjunto de dados, porém, em uma tentativa de aumentar a qualidade das detecções do modelo de maneira geral, uma segunda abordagem foi realizada, mas com uma etapa inicial diferente. Utilizando um dos conjuntos de dados de Olmos et al. (2018), na qual não havia nenhuma marcação, o detector de objetos *YOLOv2* foi modificado e utilizado para marcar mais de 9000 imagens, compondo um novo conjunto de dados (DB-3), sendo utilizado como amostras negativas. Esse novo conjunto de dados é composto pelo conjunto marcado (DB-1*) e pelo conjunto de dados de treinamento da primeira abordagem com *data augmentation*.

Na segunda abordagem deste trabalho, o detector foi treinado com um novo conjunto de dados criado (DB-3), onde apresentou uma redução de quase 40% na quantidade de falsos positivos em relação ao trabalho de Olmos et al. (2018), impactando diretamente na

precisão, onde em todas as configurações alcançou valores superiores à 88%. Em relação à sensibilidade, onde o maior valor de falso negativo não representa nem 6% da quantidade de armas presente na base de testes, todos os resultados são superiores à 95%. Dessa forma, a métrica F_1 obteve resultados superiores à 92%. Em todas as três configurações, o modelo apresentou resultados com uma confiança superior à 52%, onde a principal diferença entre os resultados se dá pelos valores de IoU. Com um aumento de quase 4%, o detector de objetos *YOLOv2* apresentou um F_1 superior ao trabalho de Olmos et al. (2018), sendo alcançado com uma confiança de 58%, uma precisão de $89,91\%$, uma sensibilidade de $99,67\%$, resultando em um F_1 de $94,54\%$.

Ambas as abordagens avaliaram a metodologia proposta neste trabalho e ainda atingiram o objetivo proposto, além disso, superando os resultados encontrados na literatura. Pode-se destacar duas contribuições importantes deste trabalho: a primeira está relacionada com a vantagem de se trabalhar com *softwares* de código aberto, onde foi possível modificar o código fonte do detector para auxiliar na tarefa de marcação automática de imagens; e a segunda está relacionada com a primeira, onde um conjunto de dados, consideravelmente grande e sem anotações, foi marcado automaticamente e utilizado como amostras negativas, aprimorando os resultados e a qualidade das detecções da segunda abordagem.

6.1 Trabalhos Futuros

Para pesquisas futuras acredita-se que outros detectores de objetos devem ser utilizados, como exemplo, versões atuais do detector de objetos utilizado neste trabalho (BOCHKOVSKIY et al., 2020; JOCHER et al., 2021) e detectores que utilizem o mesmo princípio do *YOLOv2*, como exemplo, o *SSD: Single Shot MultiBox Detector* (LIU et al., 2016). Uma abordagem para visualizar os conjuntos de dados deve ser considerada, onde deve preceder qualquer técnica para detecção, como exemplo o método estatístico t-SNE (MAATEN; HINTON, 2008).

Para a detecção de armas, destaca-se a possibilidade de aumentar ou criar um novo conjunto de dados, reforçando a utilização tanto de imagens como de vídeos. Destaca-se ainda a criação de um banco de teste mais amplo e que contenha suas devidas anotações, sendo composta por imagens e vídeos, levando em consideração o compartilhamento de tal base de maneira pública e gratuita, viabilizando a reprodutibilidade e futuras discussões. Ainda sobre o conjunto de dados, destaca-se uma possível abordagem iniciada ao longo deste trabalho, utilizando-se *software* específico para a modelagem de armas, possibilitando a criação de uma grande quantidade de imagens de armas em distintas posições, ou seja, um conjunto de dados artificial. A Figura 30 apresenta algumas imagens artificiais.

Figura 30 – Exemplo de uma imagem artificial em 4 visões diferentes.

Fonte: o Autor.

Referências

ALAHI, A.; ORTIZ, R.; VANDERGHEYNST, P. Freak: Fast retina keypoint. In: IEEE. 2012 IEEE Conference on Computer Vision and Pattern Recognition. [S.l.], 2012. p. 510–517. Citado na página 20.

BARRETO, S. C. Reconhecimento de placas veiculares utilizando deep learning: análise da influência de dados sintéticos no processo de reconhecimento. 2018. Citado na página 50.

BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. In: SPRINGER. European conference on computer vision. [S.l.], 2006. p. 404–417. Citado 2 vezes nas páginas 20 e 27.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. 2020. Citado 2 vezes nas páginas 30 e 65.

BRASIL. Lei nº 10.826, de 22 de dezembro de 2003: Dispõe sobre registro, posse e comercialização de armas de fogo e munição, sobre o sistema nacional de armas - sinarm, define crimes e dá outras providências. Diário Oficial da União, Brasília, DF, n. 1, p. 1, 2003. Disponível em: <https://www2.camara.leg.br/legin/fed/lei/2003/lei-10826-22-dezembro-2003-490580-norma-pl.html>. Citado na página 15.

BRASIL. Decreto nº 9.847, de 25 de junho de 2019: Regulamenta a lei nº 10.826, de 22 de dezembro de 2003, para dispor sobre a aquisição, o cadastro, o registro, o porte e a comercialização de armas de fogo e de munição e sobre o sistema nacional de armas e o sistema de gerenciamento militar de armas. Diário Oficial da União, Brasília, DF, n. 1, p. 1, 2019. Disponível em: <https://www.in.gov.br/en/web/dou/-/decreto-n-9847-de-25-de-junho-de-2019-172805974>. Citado 2 vezes nas páginas 15 e 16.

BRASIL. Decreto nº 10.627, de 12 de fevereiro de 2021: Altera o anexo i ao decreto nº 10.030, de 30 de setembro de 2019, que aprova o regulamento de produtos controlados. Diário Oficial da União, Brasília, DF, n. 1, p. 1, 2021. Disponível em: <https://www.in.gov.br/en/web/dou/-/decreto-n-10.627-de-12-de-fevereiro-de-2021-303712257>. Citado 2 vezes nas páginas 15 e 16.

BRASIL. Decreto nº 10.628, de 12 de fevereiro de 2021: Altera o decreto nº 9.845, de 25 de junho de 2019, que regulamenta a lei nº 10.826, de 22 de dezembro de 2003, para dispor sobre a aquisição, o cadastro, o registro e a posse de armas de fogo e de munição. Diário Oficial da União, Brasília, DF, n. 4, p. 4, 2021. Disponível em: <https://www.in.gov.br/en/web/dou/-/decreto-n-10.628-de-12-de-fevereiro-de-2021-303712338>. Citado 2 vezes nas páginas 15 e 16.

BRASIL. Decreto nº 10.629, de 12 de fevereiro de 2021: Altera o decreto nº 9.846, de 25 de junho de 2019, que regulamenta a lei nº 10.826, de 22 de dezembro de 2003, para dispor sobre o registro, o cadastro, e a aquisição de armas e de munições por caçadores, colecionadores e atiradores. Diário Oficial da União, Brasília, DF, n. 4, p. 4, 2021. Disponível em: <https://www.in.gov.br/en/web/dou/-/decreto-n-10.629-de-12-de-fevereiro-de-2021-303712419>. Citado 2 vezes nas páginas 15 e 16.

BRASIL. Decreto nº 10.630, de 12 de fevereiro de 2021: Altera o decreto nº 9.847, de 25 de junho de 2019, que regulamenta a lei nº 10.826, de 22 de dezembro de 2003, para dispor sobre a aquisição, o cadastro, o registro, o porte e a comercialização de armas de fogo e de munição e sobre o sistema nacional de armas e o sistema de gerenciamento militar de armas. Diário Oficial da União, Brasília, DF, n. 5, p. 5, 2021. Disponível em: <https://www.in.gov.br/en/web/dou/-/decreto-n-10.630-de-12-de-fevereiro-de-2021-303724469>. Citado 2 vezes nas páginas 15 e 16.

CERQUEIRA, D. et al. Atlas da violência 2020. Relatório Institucional, Instituto de Pesquisa Econômica Aplicada - IPEA, Brasília, DF, p. 96, 2020. Disponível em: <https://doi.org/10.38116%2Friatlasdaviolencia2020>. Citado 2 vezes nas páginas 15 e 16.

CERQUEIRA, D. R. d. C. Causas e consequências do crime no Brasil. [S.l.]: Banco Nacional de Desenvolvimento Econômico e Social, 2014. Citado na página 15.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: IEEE. 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). [S.l.], 2005. v. 1, p. 886–893. Citado 3 vezes nas páginas 9, 27 e 28.

DAUBECHIES, I. Ten lectures on wavelets. [S.l.]: Siam, 1992. v. 61. Citado na página 18.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. In: IEEE. 2009 IEEE conference on computer vision and pattern recognition. [S.l.], 2009. p. 248–255. Citado 2 vezes nas páginas 28 e 42.

ELMIR, Y.; LAOUAR, S. A.; HAMDAOUI, L. Deep learning for automatic detection of handguns in video sequences. In: AMINE, A. et al. (Ed.). 3rd edition of the National Study Day on Research on Computer Sciences (JERI 2019), Saida, Algeria, April 27, 2019. CEUR-WS.org, 2019. (CEUR Workshop Proceedings, v. 2351). Disponível em: <http://ceur-ws.org/Vol-2351/paper_69.pdf>. Citado 10 vezes nas páginas 23, 24, 25, 44, 52, 57, 58, 59, 60 e 64.

EVERINGHAM, M. et al. The pascal visual object classes (voc) challenge. International journal of computer vision, Springer, v. 88, n. 2, p. 303–338, 2010. Citado 3 vezes nas páginas 32, 42 e 47.

FUKUSHIMA, K.; MIYAKE, S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: Competition and cooperation in neural nets. [S.l.]: Springer, 1982. p. 267–285. Citado na página 29.

G1. Cronologia: massacre em Suzano. 2019. <https://g1.globo.com/sp/mogi-das-cruzes-suzano/noticia/2019/03/13/cronologia-massacre-em-suzano.ghtml>. Acesso: 30-05-2019. Citado na página 16.

G1. Sobe para 50 número de mortos em ataques a mesquitas na Nova Zelândia. 2019. <https://g1.globo.com/mundo/noticia/2019/03/16/sobe-para-50-numero-de-mortos-em-ataques-a-mesquitas-na-nova-zelandia.ghtml>. Acesso: 25-05-2019. Citado na página 16.

GANDHI, R. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms - Understanding object detection algorithms. 2018. <https://towardsdatascience.com/

r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>. Acesso: 21-11-2021. Citado na página 34.

GELANA, F.; YADAV, A. Firearm detection from surveillance cameras using image processing and machine learning techniques. In: Smart Innovations in Communication and Computational Sciences. [S.l.]: Springer, 2019. p. 25–34. Citado 2 vezes nas páginas 23 e 24.

GESICK, R.; SARITAC, C.; HUNG, C.-C. Automatic image analysis process for the detection of concealed weapons. In: ACM. Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies. [S.l.], 2009. p. 20. Citado 2 vezes nas páginas 9 e 18.

GIRSHICK, R. Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. [S.l.: s.n.], 2015. p. 1440–1448. Citado na página 29.

GIRSHICK, R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2014. p. 580–587. Citado na página 29.

GONZALEZ, R.; WOODS, R. Processamento Digital De Imagens. ADDISON WESLEY BRA, 2009. ISBN 9788576054016. Disponível em: <https://books.google.com.br/books?id=r5f0RgAACAAJ>. Citado na página 26.

GREGA, M.; ŁACH, S.; SIERADZKI, R. Automated recognition of firearms in surveillance video. In: IEEE. 2013 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA). [S.l.], 2013. p. 45–50. Citado 3 vezes nas páginas 21, 22 e 23.

GREGA, M. et al. Automated detection of firearms and knives in a cctv image. Sensors, Multidisciplinary Digital Publishing Institute, v. 16, n. 1, p. 47, 2016. Citado 2 vezes nas páginas 22 e 23.

HALIMA, N. B.; HOSAM, O. Bag of words based surveillance system using support vector machines. International Journal of Security and Its Applications, v. 10, n. 4, p. 331–346, 2016. Citado na página 20.

HARRIS, C. G.; STEPHENS, M. et al. A combined corner and edge detector. In: CITESEER. Alvey vision conference. [S.l.], 1988. v. 15, n. 50, p. 10–5244. Citado na página 20.

HE, K. et al. Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. [S.l.: s.n.], 2017. p. 2961–2969. Citado 2 vezes nas páginas 16 e 29.

JACCARD, P. Etude de la distribution florale dans une portion des alpes et du jura. Bulletin de la Societe Vaudoise des Sciences Naturelles, v. 37, p. 547–579, 01 1901. Citado na página 47.

JOCHER, G. et al. ultralytics/yolov3: v9.1 - YOLOv5 Forward Compatibility Updates. Zenodo, jan. 2021. Disponível em: <https://doi.org/10.5281/zenodo.4435632>. Citado 2 vezes nas páginas 30 e 65.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). Advances in Neural Information Processing Systems. Curran Associates, Inc., 2012. v. 25, p. 1097–1105. Disponível em: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>. Citado 2 vezes nas páginas 28 e 29.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. Neural computation, MIT Press, v. 1, n. 4, p. 541–551, 1989. Citado na página 29.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. European conference on computer vision. [S.l.], 2014. p. 740–755. Citado 3 vezes nas páginas 42, 54 e 61.

LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. European conference on computer vision. [S.l.], 2016. p. 21–37. Citado na página 65.

LOEY, M. et al. Fighting against covid-19: A novel deep learning model based on yolo-v2 with resnet-50 for medical face mask detection. Sustainable cities and society, Elsevier, v. 65, p. 102600, 2021. Citado na página 50.

LOFF, S. Apollo 11 Mission Overview. 2019. <https://www.nasa.gov/mission_pages/apollo/missions/apollo11.html>. Acesso: 22-09-2020. Citado na página 26.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. International journal of computer vision, Springer, v. 60, n. 2, p. 91–110, 2004. Citado 3 vezes nas páginas 18, 26 e 27.

MAATEN, L. Van der; HINTON, G. Visualizing data using t-sne. Journal of machine learning research, v. 9, n. 11, 2008. Citado na página 65.

MACHEMER, T. Nasa enviou câmeras fotográficas à Lua nos anos 1960 para planejar pouso histórico da Apollo. 2019. <https://bit.ly/NatGeoNasa>. Acesso: 21-09-2020. Citado na página 26.

MAZZA, G. C. Flexibilização de armas: Entenda o que muda com os decretos de Bolsonaro. 2021. <https://jovempan.com.br/noticias/politica/flexibilizacao-de-armas-entenda-o-que-muda-com-os-decretos-de-bolsonaro.html>. Acesso: 19-02-2021. Citado na página 15.

MERKEL, D. Docker: lightweight linux containers for consistent development and deployment. Linux journal, v. 2014, n. 239, p. 2, 2014. Citado na página 49.

OLMOS, R.; TABIK, S.; HERRERA, F. Automatic handgun detection alarm in videos using deep learning. Neurocomputing, Elsevier, v. 275, p. 66–72, 2018. Citado 20 vezes nas páginas 16, 22, 23, 24, 25, 39, 40, 41, 42, 44, 50, 52, 53, 57, 58, 59, 60, 61, 64 e 65.

PRATI, R.; BATISTA, G.; MONARD, M. Curvas roc para avaliação de classificadores. Revista IEEE América Latina, v. 6, n. 2, p. 215–222, 2008. Citado na página 46.

REDMON, J. Darknet: Open Source Neural Networks in C. 2013–2016. <http://pjreddie.com/darknet/>. Citado 5 vezes nas páginas 37, 41, 42, 50 e 51.

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. [S.l.: s.n.], 2016. p. 779–788. Citado 6 vezes nas páginas 29, 30, 31, 32, 34 e 35.

REDMON, J.; FARHADI, A. Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2017. p. 7263–7271. Citado 13 vezes nas páginas 10, 16, 25, 30, 35, 36, 41, 42, 43, 44, 50, 53 e 64.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. Computer Vision and Pattern Recognition, cite as, 2018. Citado na página 30.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. [S.l.: s.n.], 2015. p. 91–99. Citado 2 vezes nas páginas 23 e 29.

ROSEBROCK, A. Intersection over Union (IoU) for object detection. 2016. <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>. Acesso: 22-06-2020. Citado na página 48.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. Citado 2 vezes nas páginas 29 e 42.

SORANO, V.; VELASCO, C.; NERI, F. Decreto de Bolsonaro facilita porte de arma para mais categorias. 2019. <https://g1.globo.com/politica/noticia/2019/05/08/decreto-de-bolsonaro-facilita-porte-de-arma-para-mais-categorias.ghtml>. Acesso: 20-05-2019. Citado na página 15.

SZEGEDY, C. et al. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. [S.l.: s.n.], 2015. p. 1–9. Citado 2 vezes nas páginas 29 e 31.

TIWARI, R. K.; VERMA, G. K. A computer vision based framework for visual gun detection using harris interest point detector. Procedia Computer Science, Elsevier, v. 54, p. 703–712, 2015. Citado 3 vezes nas páginas 19, 20 e 21.

TIWARI, R. K.; VERMA, G. K. A computer vision based framework for visual gun detection using surf. In: 2015 International Conference on Electrical, Electronics, Signals, Communication and Optimization (EESCO). [S.l.: s.n.], 2015. p. 1–5. Citado 2 vezes nas páginas 20 e 21.

UIJLINGS, J. R. et al. Selective search for object recognition. International journal of computer vision, Springer, v. 104, n. 2, p. 154–171, 2013. Citado na página 29.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: IEEE. Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001. [S.l.], 2001. v. 1, p. I–I. Citado na página 26.

WESTLAKE, N.; CAI, H.; HALL, P. Detecting people in artwork with cnns. In: SPRINGER. European Conference on Computer Vision. [S.l.], 2016. p. 825–841. Citado 2 vezes nas páginas 10 e 35.

WU, Z. et al. Multi-scale vehicle detection for foreground-background class imbalance with improved yolov2. Sensors, Multidisciplinary Digital Publishing Institute, v. 19, n. 15, p. 3336, 2019. Citado na página 50.

XIAO, Z. et al. Automatic detection of concealed pistols using passive millimeter wave imaging. In: IEEE. 2015 IEEE International Conference on Imaging Systems and Techniques (IST). [S.l.], 2015. p. 1–4. Citado 4 vezes nas páginas 9, 18, 19 e 20.

YUJIRI, L.; SHOUCRI, M.; MOFFA, P. Passive millimeter wave imaging. IEEE Microwave Magazine, v. 4, n. 3, p. 39–50, 2003. Citado na página 19.