Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Estudo de caso para filtragem de falsos positivos na detecção de armas de fogo com autoencoders

Lineker Aguiar Alcântara

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Eduardo da Silva Ribeiro

Fevereiro, 2024
João Monlevade–MG

Lineker Aguiar Alcântara

Estudo de caso para filtragem de falsos positivos na detecção de armas de fogo com autoencoders

Orientador: Eduardo da Silva Ribeiro

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Fevereiro de 2024

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

A347e Alcântara, Lineker Aguiar.
Estudo de caso para filtragem de falsos positivos na detecção de armas de fogo com autoencoders. [manuscrito] / Lineker Aguiar Alcântara. - 2024.
58 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Eduardo Ribeiro.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Engenharia de Computação .

1. Aprendizado do computador. 2. Redes neurais (Computação). 3. Sistemas de segurança eletrônico. 4. Videovigilância. 5. Visão por computador. I. Ribeiro, Eduardo. II. Universidade Federal de Ouro Preto. III. Título.

CDU 004.8

Bibliotecário(a) Responsável: Flavia Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Lineker Aguiar Alcântara

Estudo de caso para filtragem de falsos positivos na detecção de armas de fogo com autoencoders.

Monografia apresentada ao Curso de Sistemas de Informação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada em 22 de Fevereiro de 2024.

Membros da banca:

Doutor - Eduardo da Silva Ribeiro - Orientador - Universidade Federal de Ouro Preto
Doutor - Luiz Carlos Bambirra Torres - Universidade Federal de Ouro Preto
Doutor - Talles Henrique de Medeiros - Universidade Federal de Ouro Preto
Bacharel - Bruno César Cota Conceição - Universidade Federal de Ouro Preto

Eduardo da Silva Ribeiro, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 04/03/2024

Documento assinado eletronicamente por **Eduardo da Silva Ribeiro**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 19/03/2024, às 09:28, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0686418** e o código CRC **FD6248C7**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0819 - www.ufop.br

Este trabalho é dedicado a minha família devido ao seu apoio incondicional durante todo o período de curso

Agradecimentos

Agradeço a minha família pelo apoio e força durante todo o curso e não deixaram faltar nada mesmo a distância separando-nos fisicamente. Aos meus familiares e amigos que me sempre depositaram confiança e responsabilidade devida com as minhas atitudes.

Agradeço também a todos os amigos formados durante o curso e em especial os colegas de república com os quais convivi por mais tempo do que minha família nos últimos 5 anos e tiveram papel especial deste período por tornar o curso e distância dos meus pais mais leves.

Agradeço também ao meu professor orientador Eduardo Ribeiro e todos os demais professores pelos ensinamentos e inspirações passadas durante a faculdade, pela formação não só como um profissional, mas também uma pessoa melhor.

"Artificial Intelligence will be humanity's greatest achievement. But it could also be the last"

— Stephen Hawking (1942 – 2018)

Resumo

Sistemas baseados em vídeos de segurança são atualmente essenciais para manutenção da segurança em locais públicos e privados como museus, estações de metrô, bancos ou lojas. No entanto, de acordo a quantidade de locais para monitoramento, bem como a quantidade de pessoas no local simultaneamente, maiores são as chances de fatores humanos como fadiga e cansaço atingir os operadores dos sistemas, tornando-os ineficientes com o passar do tempo. Devido os avanços da área da aprendizagem profunda e no campo de visão computacional este trabalho propõe um estudo sobre o impacto de objetos similares aos objetos alvo em modelos de detecção de objeto. Como um abordagem para lidar com a emissão de falsos positivos o uso de autoencoders como um filtro de falsos positivos que são comumente gerados pelo detector primário. O uso de imagens com objetos similares aos objetos alvos de detecção se mostrou prejudicial para os modelos construídos, com aumentos consideráveis na emissão de falsos positivos e falsos negativos. A abordagem utilizada do uso de autoencoders para tratativa da inserção de objetos similares, no entanto, não se mostrou eficaz para o problema, obtendo variações pouco significativas e em alguns casos em perda de precisão do modelo

Palavras-chaves: Aprendizagem profunda, visão computacional, autoencoders, falso positivo.

Abstract

Security video-based systems are currently essential for maintaining security in public and private locations such as museums, subway stations, banks, or stores. However, depending on the number of locations to monitor and the quantity of people present simultaneously, the chances of human factors such as fatigue and exhaustion affecting the system operators increase, making them inefficient over time. Due to advancements in deep learning and computer vision, this work proposes a study on the impact of objects similar to the target objects in object detection models. As an approach to address the issue of false positives, the use of autoencoders is suggested as a filter for false positives commonly generated by the primary detector. The inclusion of images with objects similar to the target detection objects proved to be detrimental to the constructed models, resulting in significant increases in both false positives and false negatives. However, the approach of using autoencoders to handle the insertion of similar objects did not prove effective for the problem, showing little significant improvement and, in some cases, a loss of model precision.

Key-words: Deep learning, computer vision, autoencoders, false positive

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

FP Falso Positivo

AE *Autoencoder*

AEE *Adversarial Autoencoders*

VP Verdadeiro Positivo

FN Falso Negativo

KNN *K-Nearest Neighbors*

SVM *Suport Vector Machines*

RNA Redes Neurais Artificiais

RNC Redes Neurais Convolucionais

R-CNN *Region-based Convolutional Neural Networks*

SPPNet *Spatial Pyramid Pooling Networks*

FPN *Feature Pyramid Networks*

SSD *Single Shot MultiBox Detector*

YOLO *You Only Look Once*

IOU *Intersection over Union*

mAP *Mean Average Precision*

GPU *Graphics Processing Unit*

E-ELAN *Extended Efficient Layer Aggregation Netowork*

KDE *Kernel Density Estimation*

Lista de símbolos

α	Letra grega Alfa
γ	Letra grega Gama
σ	Letra grega Sigma
$\mathbb{R}$	Conjunto dos números reais
$\sum$	Somatório
$\in$	Pertence

Sumário

1 Introdução

O conhecimento prévio de perigo em potencial pode ser crucial para resposta rápida e efetiva a fim de eliminar ou mitigar os possíveis danos (ENRÍQUEZ et al., 2019). Tradicionalmente, a vigilância de ambientes públicos tem sido realizada por agentes humanos através as imagens capturadas pelo sistema (VALLEZ; VELASCO-MATA; DENIZ, 2021). No entanto, mesmo agentes com vasta experiência neste tipo de trabalho podem deixar passar situações de perigo devido a causas humanas como fadiga, cansaço ou falta de atenção (ASHBY, 2017).

Em grande parte de localidades públicas como bancos, estações de trens ou joalherias, a simples presença de uma arma pode gerar uma situação de ameaça e perigo, assim uma detecção automática nestes tipos de problemas com câmeras de segurança podem certamente reduzir a quantidade dados causados (OLMOS et al., 2021). Como ainda destacado pelo autor o problema da detecção automática de armas é ainda uma questão em aberta devido a diversos fatores como o pequeno tamanho dos objetos na cena ou possibilidade de oclusão pelas mãos

Dessa maneira, o desenvolvimento de sistemas inteligentes capazes de automaticamente realizar a detecção de ameaças ou riscos envolvendo armas de fogo de maneira rápida pode promover vantagens significativas em termos de segurança e tomada de ação (RUIZ-SANTAQUITERIA et al., 2021). Recentemente, devido aos grandes avanços em técnicas de aprendizagem profunda, diversas melhorias têm sido alcançadas em tarefas ligadas ao campo de visão computacional, como classificação, detecção ou segmentação de imagens (OLMOS; TABIK; HERRERA, 2018). No contexto particular para detecção de armas de fogo, apesar dos resultados promissores obtidos nesta tarefa nos últimos anos.

Isto posto, este trabalho visa a modelagem de um sistema de detecção automática de armas através da aplicação de Redes Neurais Convolucionais (RNC) e, utilizando como modelo a *You Only Look Once* (YOLO)V7 (WANG; BOCHKOVSKIY; LIAO, 2023) , além disso, visando mitigar a taxa de falsos positivos para uma maior acurácia do problema, será utilizado um *Autoencoder* (AE) como um filtro para evitar a emissão de alarmes falsos através de técnicas de detecção de anomalias, baseando-se em métricas como o erro de reconstrução do modelo, e a densidade do seu vetor central de compressão.

A escolha do foco na redução da taxa de Falso Positivo (FP), se pela ampla abordagem observada na literatura devido aos impactos causados e possível mobilização de forças ao realizar uma detecção de arma de fogo em um ambiente crítico, além disso, o não foco no *recall* é atribuído ao fato de quem, em curto espaço de tempo, segundos ou milissegundos, o modelo terá diferentes imagens para realizar a detecção, tornando

mais provável a segura de que, ainda com um atraso insignificante, o objeto venha a ser detectado.

1.1 O problema de pesquisa

A violência armada, especificamente com armas de fogo, tem sido um problema recorrente em todo mundo, especialmente em países onde o porte de armas é legalizado. Com o passar dos anos, o número de pessoas inocentes mortas ou feridas por violência armada em locais públicos ainda é alto (CUKIER; EAGEN; DECAT, 2016). Desta maneira a detecção precoce de ameaças a mão armada é crucial para redução de danos, destacado por (OLMOS et al., 2019), sistemas inteligentes para prevenção destes problemas deve produzir alarmes aproximadamente em tempo real e somente quando houver uma grande certeza da presença de armas na cena.

O alcance de altas performances de detecção, em termos de acurácia, mesmo com modelos de estado da arte de detecção de objetos em vídeos de segurança, ainda é um problema em aberto (OLMOS et al., 2019). Para os autores, grande parte dos falsos positivos gerados por tais detectores são objetos de *background* produzidos devido aos algoritmos de busca seletiva dos modelos de detecção que assumem que todas as áreas das imagens de entradas são regiões candidatas.

Para Olmos et al. (2021), a complexidade desta tarefa aumenta significativamente em situações de perigo devido a movimentos rápidos e repentinos, que podem ter uma alta variabilidade de condições de luminosidade e perda de qualidade das imagens, sendo esta um dos principais fatores de aumento da taxa de Falso Positivo (FP), e tornando o maior desafio destes cenários. González et al. (2020) ainda pontua que, a baixa precisão decorre da dificuldade de detecção de objetos muito pequenos, no caso de armas de fogo, sua representatividade pode chegar a cerca de 3% do total da cena.

1.2 Objetivos

A presente pesquisa tem por objetivo primário a avaliação do impacto da adição de um AE na saída da RNC utilizada como base, a YOLO V7, utilizando diferentes metodologias e métricas para filtragem de FP gerados pelo modelo base. Além disso, tem-se como objetivo a construção de conjunto de dados para treinamentos de detecção de armas de fogo, contendo objetos que podem ser detectados como FP e realizar medidas sobre a inserção destes objetos resultado final da rede.

- Treinamentos de modelos de um estágio para detecção de armas de fogo.
- Construção e treinamento de AE para filtragem de falsos positivos

- Elaboração de um conjunto de dados com objetos com potencial de detecção de FP
- Avaliação do modelo base YOLO V7 sob os dados com ou sem objetos similares.

1.3 Metodologia

Como posto, o trabalho aqui desenvolvido tem por objetivo a avaliação de técnicas e estratégias para detecção de armas de fogo. Além das estratégias abordadas, será avaliado o impacto de treinamentos de modelos de aprendizagem profunda contendo imagens com objetos que tenham potencial de ser detectadas como um alarme falso. Assim, os passos seguidos para seu desenvolvimento foram:

- Revisão da literatura a respeito da evolução de técnicas de aprendizagem profunda aplicadas à detecção objetos em âmbitos gerais.
- Pesquisa sobre as abordagens utilizadas na literatura para o problema em específico e seus principais desafios enfrentados.
- Escolha e avaliação das arquiteturas que se destacam dado o contexto da pesquisa.
- Revisão sobre mecanismo utilizado (AE), funcionamento, abordagens e como emprega-lo como um detector de anomalias,
- Pesquisa sobre os conjuntos de dados disponíveis publicamente com respeito a detecção de armas de fogo.
- Escolha de um conjunto de dados base contendo somente exemplos de armas de fogo.
- Elaboração de um conjunto de dados distinto contendo imagens que possam ser detectadas como FP.
- Estudo do impacto causado pelo treinamento de modelos com a inserção das imagens similares ao objeto no conjunto de dados de treinamento.

1.4 Organização do Trabalho

A partir deste ponto, a continuação do trabalho é organizada da seguinte maneira; Capítulo 2 apresenta uma revisão da literatura no que se refere sobre os principais conceitos de aprendizagem profunda e o funcionamento de Redes Neurais Artificiais (RNA) e Redes Neurais Convolucionais (RNC), suas aplicações para a tarefa de detecção de objetos, passando pelo funcionamento de um *autoencoder* convolucional e por fim, discorrendo sobre os trabalhos relacionados. No Capítulo 3 será discutidos os métodos abordados abordados e configurações utilizadas para treinamentos dos modelos e o Capítulo 4 será explanado os resultados obtidos dos experimentos executados e suas possíveis interpretações

2 Revisão bibliográfica

Este capítulo apresenta uma revisão da literatura sob uma ótica geral a respeito das tecnologias e fundamentos das redes neurais, seus aspectos de desenvolvimento,passando pelas redes densamente conectadas, convolucionais e os AE. Será dissertado também a literatura corrente sobre o problema em questão e como este vem sendo abordado pela comunidade de pesquisa.

2.1 Redes Neurais Artificiais

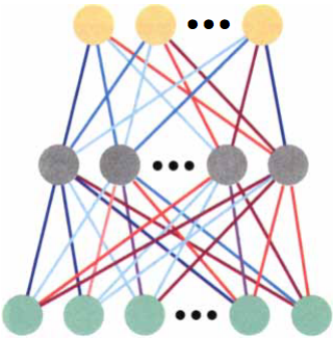
O primeiro modelo computacional de um neurônio foi realizado por McCulloch e Pitts (1943), com objetivo de abstrair o funcionamento do cérebro humano em um modelo matemático. Destacado em Goodfellow, Bengio e Courville (2016), como os primeiros algoritmos de aprendizagem desenvolvidos propunham uma modelagem computacional do sistema de aprendizagem biológico, tendo como resultado o cunho do termo Redes Neurais Artificiais.

Contudo, tais algoritmos de *deep learning* não são desenhados para serem uma representação realística de funções biológicas. O termo moderno "*deep learning*" representa não só a perspectiva neurocientífica sobre os atuais modelos de aprendizagem de máquina, mas também possui uma visão mais generalista do princípio de aprendizagem de múltiplos níveis de composição que podem ser aplicados em estruturas de aprendizagem de máquina que não são necessariamente inspirados no cérebro biológico.

Uma rede neural artificial é tipicamente formada por três camadas de neurônios totalmente conectados, cada neurônio realiza operações matemáticas sobre os dados e transmitem as informações processadas para as camadas subsequentes (HINTON, 1992).

- Camada de entrada: a atividade da camada de entrada se da pela recepção dos dados que serão utilizados como fonte de informação para alimentar a rede. Representada pela camada em verde na Figura 3.
- Camada escondida: sua atividade é definida pelas atividades da camada de entrada e os pesos entre suas conexões. Tais pesos são definidos durante o processo de aprendizagem, e são responsáveis por definir os níveis de ativação dos neurônios. A camada escondida também é onde tem-se a identificação dos padrões dos dados. Representada pela camada em cinza na Figura 3.
- Camada de saída: de maneira similar, a camada de saída tem seu comportamento dependente das unidades da camada escondida e de seus pesos. Produz a resposta

Figura 1 – Rede neural artificial de três camadas totalmente conectadas.



Fonte: Hinton (1992)

final da rede. Representada pela camada em amarelo na Figura 3.

Como destacado em Goodfellow, Bengio e Courville (2016), uma rede neural tem por objetivo a aproximação de uma função f^*. Como para um classificador, deseja-se mapear as entradas x para uma saída y, obtendo $y = f^*(x)$, para tal tarefa o modelo define uma função de mapeamento $y = f(x;\theta)$ e aprende os valores dos parâmetros θ para obter a melhor função aproximadora. Em uma rede de múltiplas camadas $f^1, f^2, ...f^n$ temos uma cadeia de funções para formar $f(x) = f(n) \circ f(n-1) \circ \ldots \circ f(2) \circ f(1)(x)$. Cada neurônio é responsável pela aplicação de duas operações matemáticas principais, a operação linear, descrita pela Equação 2.1 e a operação de não linearidade, descrita na Equação 2.2.

$$f(x; w, b) = w \cdot x + b \tag{2.1}$$

$$Z = \sigma(f(x; w, b)) = \sigma(w \cdot x + b) \tag{2.2}$$

Onde x representa o vetor de dados de entrada, w, os pesos da camada corrente que serão ajustados durante o treinamento para aprender a relação entre as entradas e saída e b, o termo de viés, usado para ajustar a saída da multiplicação entre os pesos e os dados de entada. Importante ressaltar que, para uma rede de multicamadas, o termo x representa sempre a saída da camada anterior, generalizando para uma camada qualquer L, temos a Equação 2.3.

$$Z^L = \sigma(w^L \cdot x^{L-1} + b^L) \tag{2.3}$$

Sem o uso das funções de ativação na saída de cada neurônio, uma rede neural terá sua função de aproximação como uma simples função polinomial de grau 1, ou seja,

uma função linear (SHARMA; SHARMA; ATHAIYA, 2017). A aplicação das funções de ativação permitem a criação de modelos mais complexos, capazes de executarem tarefas de caráter não linear, como encontrar padrões em dados de alta dimensionalidade, imagens, áudios e vídeos.

2.2 Redes Neurais Convolucionais

Redes Neurais Convolucionais (RNC) são modelos especializados no processamento de dados com topologia de grades, como séries temporais, que podem ser tratadas como grades unidimencionais e dados de imagens, com grides bidimencionais (LECUN et al., 1989). As RNC têm se mostrado de grande sucesso e de grandes impactos em aplicações práticas. O termo "convolução" é usado devido as operações matemáticas de convolução aplicadas em pelo menos uma das camadas de sua arquitetura.

2.2.1 Base Neurocientífica

As RNC podem ser a maior história de sucesso da inteligência artificial de inspiração biológica (GOODFELLOW; BENGIO; COURVILLE, 2016). Embora tais modelos sejam guiados por diversos outros campos de estudo, algumas de suas principais características são derivadas da neurociência. Os estudos dirigidos por Hubel e Wiesel (1962) através da observação de como neurônios de gatos respondem a exposição de imagens são de grande influência no desenvolvimento dos modelos criados no decorrer das últimas décadas.

As RNC são construídas com base em três características principais do córtex visual primário, o qual é a primeira área do cérebro que inicia um processamento mais avançado de entradas visuais.

1. O córtex visual primário é organizado em um mapa espacial, possuindo uma estrutura bidimensional que espelha a estrutura da retina. As redes convolucionais utilizam desta propriedade através de suas *features* que são definidas também em um mapa bidimensional.

2. As unidades detectoras de uma rede convolucional são projetadas para emular as células da córtex visual primário, que em grande número, podem realizam operações que podem ser caracterizadas como uma função linear da imagem em um pequeno campo receptivo.

3. As células mais complexas do córtex visual primário são invariantes a pequenas mudanças da posição de imagens, servindo de inspiração para as camadas de *pooling* presente em grande parte das RNC. Ademais as células complexas também possuem invariância para algumas mudanças de luminosidade, que são inspiração

para estratégias da construção de *pooling* entre diferentes canais (GOODFELLOW et al., 2013).

2.2.2 Convolução

A operação de convolução, presente nas RNC são responsáveis por algumas características desse modelo, como compartilhamento de parâmetros, representações equivariantes e interações esparsas (LECUN; BENGIO et al., 1995). De maneira matemática, uma operação de convolução pode ser representada por:

$$s(t) = (x * w)(t) \tag{2.4}$$

Aqui, tem-se $x(t)$ como um sinal de entrada representando uma função contínua no tempo ou uma sequência de amostras, $w(t)$ representa o *kernel*, ou janela que será deslocada no tempo para realizar a convolução com $x(t)$. O sinal de saída $s(t)$ é então obtido através dessa multiplicação ponto a ponto entre os sinais de entrada e somando os produtos resultantes.

Em termos das RNC, o primeiro argumento x da operação de convolução é a nossa entrada de dados, uma imagem ou a saída processada de uma camada anterior, já o segundo argumento w trata-se do *kernel*, conjunto de filtros que serão usados na operação (GOODFELLOW; BENGIO; COURVILLE, 2016). Para a saída, tem-se o que é chamado de mapa de atributos, espaços bidimensionais de diversas camadas com diferentes tipos de representações das características da imagem de entrada (LECUN; BENGIO et al., 1995).

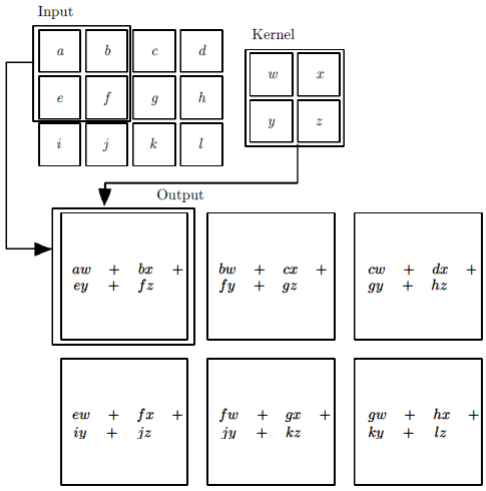
Tratando-se de uma operação em uma imagem I 2D utilizando um *Kernel* K 2D, pode-se representar a Equação 2.4 como:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.5}$$

Como ilustrado na Figura 2, a saída da operação de convolução será determinada pelas dimensões do *kernel* utilizado, com uma relação inversamente proporcional, além de parâmetros como o *stride*, que define o passo de deslize do kernel e o *padding*, que realiza um preenchimento das bordas da imagem com valores nulos com objetivo de realizar uma maior captura das bordas (RASCHKA, 2015).

Trabalhando com dados 2D como imagens, ou dados 1D como sinais de fala, entre outros, tais tipos de dados podem ter diversas variações, como velocidade de fala, tons e entonação para problemas ligados à conversação ou variação de posição, tamanho ou inclinação de objetos em imagens. Estes comportamentos pode causar posições distintas dos atributos dos dados de entrada que podem ser capturados por uma rede densamente conectada suficientemente grande, porém obtendo múltiplos pesos de mesmo valor, além

Figura 2 – Ilustração de uma convolução com kernel 2x2.

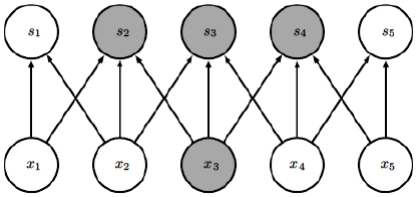


Fonte: Goodfellow, Bengio e Courville (2016)

de um alto custo computacional e possível *overfitting* (LECUN; BENGIO et al., 1995). Assim, as RNC fazem uso das conexões esparsas, compartilhamento de parâmetros e representações equivariantes.

A conexão esparsa é realizada através do uso do *kernel*, em grande parte das aplicações menor do que a imagem de entrada, dessa forma, um único elemento do mapa de recursos está conectado somente com um pedaço da imagem (RASCHKA, 2015). Ademais, tal característica implica em menores quantidades de parâmetros, dessa maneira, reduzindo o uso de memória.

Figura 3 – Ilustração de uma conexão esparsa.



Fonte: Goodfellow, Bengio e Courville (2016)

O compartilhamento de parâmetros, (RASCHKA, 2015), se da pelo fato dos mesmos pesos (*kernel* de convolução), serem utilizados por diferentes pedaços da imagem de entrada. Assim, esta técnica também implica na redução do que é chamado de parâmetros livres, (LECUN; BENGIO et al., 1995), reduzindo a capacidade do modelo e melhorando sua habilidade de generalização.

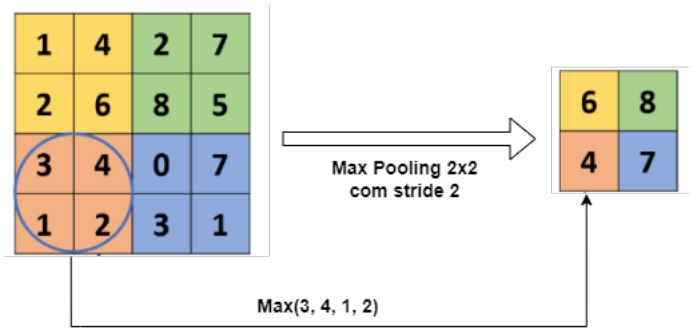
Como efeito do compartilhamento de parâmetros, temos as equivariações à translação, ou, representações equivariantes. Como destacado por Goodfellow, Bengio e Courville (2016), uma função $f(x)$ é equivariante a uma função $g(x)$ se $f(g(x)) = g(f(x))$. Ou seja, se é aplicado uma modificação na imagem de entrada e, posteriormente a operação de convolução, o resultado será o mesmo se aplicada primeiramente a operação de convolução e posteriormente a mesma modificação na saída da convolução.

2.2.3 Pooling

Os modelos de RNC são usualmente formados por diversas camadas de convolução, seguidas por uma função de ativação e posteriormente uma camada de *pooling*, (RASCHKA, 2015), sendo esta servindo de dois propósitos principais. Primeiro, realiza a redução da quantidade de parâmetros e, consequentemente a redução de custo computacional. Segundo, a camada pode controlar o *overfitting* do modelo, extraindo somente informações importante para a rede e descartando detalhes irrelevantes, (GHOLAMALINEZHAD; KHOSRAVI, 2020).

Existem na literatura, diversos tipos de *pooling*, sendo os mais usuais o *Max Pooling*, que retorna o valor máximo dentro de uma janela retangular da saída da camada de convolução, como ilustrado na Figura 4. Outro método comum da operação é o *Average Pooling*, que retorna o valor médio dentro de uma janela retangular da mesma saída da camada de convolução (GHOLAMALINEZHAD; KHOSRAVI, 2020), ilustrado na Figura 5.

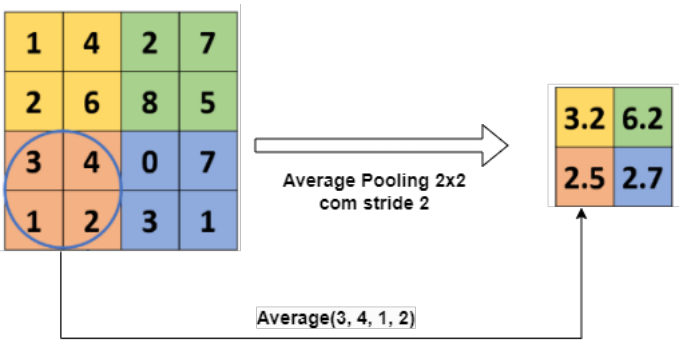
Figura 4 – Max Pooling.



Fonte: Adaptado de Gholamalinezhad e Khosravi (2020)

Modelos tradicionais de RNC são usualmente construídos sobre esses três pilares, camadas convolucionais, camadas de *pooling*, e por fim, uma rede densamente conectada para achatar os operações sobre os dados de bidimensionais para unidimensionais (RASCHKA, 2015)

Figura 5 – Average Pooling.



Fonte: Adaptado de Gholamalinezhad e Khosravi (2020)

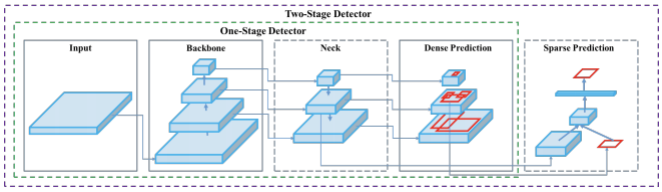
2.3 Redes Neurais Para Detecção de Objetos

Relatado em Zou et al. (2023) a detecção de objetos é uma importante tarefa relacionada ao campo de visão computacional. O objetivo desta tarefa se da pelo desenvolvimento de modelos e técnicas computacionais que permitam responder um básico e essencial conhecimento de aplicações de visão computacional "quais objetos estão onde". Para isso, existem duas importantes métricas, que caracterizam um *trade off*, a precisão da detecção, e sua velocidade de processamento.

Com o rápido desenvolvimento de técnicas de aprendizagem profunda, redes neurais convolucionais profundas têm se tornado cruciais para as tarefas de detecção de objeto (XIAO et al., 2020). Os autores ainda destacam que, comparados com métodos baseados em atributos manualmente criados, as técnicas de aprendizagem profunda pode aprender tanto atributos de baixo nível quanto de alto nível das imagens de entrada, sendo esses, atributos mais representativos do que os manualmente criados. Atualmente, técnicas de detecção de objetos com base no aprendizado profundo tem sido vastamente utilizadas como em veículos autônomos, visão robótica e em vídeos de segurança (ZOU et al., 2023).

Tarefas ligadas ao campo de visão computacional e detecção de objetos, têm, muitas vezes uma grande variedade de tipos de objetos e limitações. Alguns desafios comuns deste campo são objetos sob diferentes pontos de vista, iluminação, qualidade da imagem, diferentes escalas (objetos pequenos e grandes), além de oclusão total ou parcial do que deseja-se detectar (ZOU et al., 2023). Para lidar com tais desafios existem atualmente dois tipos principais de detectores baseados em aprendizagem profunda: os detectores em dois estágios os detectores em um estágio Figura 6.

Figura 6 – Arquitetura geral de um detector de objeto.



Fonte: Bochkovskiy, Wang e Liao (2020)

2.3.1 Detectores de dois estágios

Os detectores de dois estágios, segundo Zou et al. (2023), dividem o processo em duas etapas, primeiro a localização das regiões de interesse e posteriormente suas classificações. Na primeira etapa, estes modelos geram propostas de regiões que podem conter objetos através das *Region-based Convolutional Neural Networks* (R-CNN) (UIJLINGS et al., 2013). Na segunda etapa essas regiões são classificadas como objetos ou não.

Modelos em dois estágios são evoluções graduais desde as R-CNN, *Spatial Pyramid Pooling Networks* (SPPNet) (HE et al., 2015), *Fast RCNN* (GIRSHICK, 2015), *Faster RCNN* (REN et al., 2015) e *Feature Pyramid Networks* (FPN) (LIN et al., 2017a). Este tipo de arquitetura alcança com facilidade uma alta acurácia nas tarefas empregadas, no entanto, devido ao seu nível de complexidade e grande número de computações são raramente aplicados em tarefas que exigem uma alta velocidade de detecção ainda com *hardwares* limitados (ZOU et al., 2023)

2.3.2 Detectores de um estágio

Para os detectores de um estágio, sua arquitetura é projetada para prever diretamente a classe e a localização dos objetos em uma única etapa. Em grande parte, são compostos por RNC para extrair as características das imagens e, em seguida é aplicada uma camada de classificação e regressão para obter as previsões (NELSON; SOLAWETZ, 2020).

As arquiteturas de um estágio possuem diversos modelos de referência como *Single Shot MultiBox Detector* (SSD) (LIU et al., 2016), tendo como maior contribuição as técnicas de detecção com multi-referências e multi-resoluções. Lin et al. (2017b) descobriam que tais tipos de modelos sofriam com o grande desbalanceamento entre as *bounding boxes* dos objetos que deseja-se detectar as *bounding boxes* com objetos de fundo, introduzindo assim a *RetinaNet* com sua nova função de custo *Focal Loss*, colocando um maior foco nos objetos de maior dificuldade de detecção, permitindo que tais arquiteturas pudessem se comparar aos detectores de dois estágios em termos de precisão (ZOU et al., 2023).

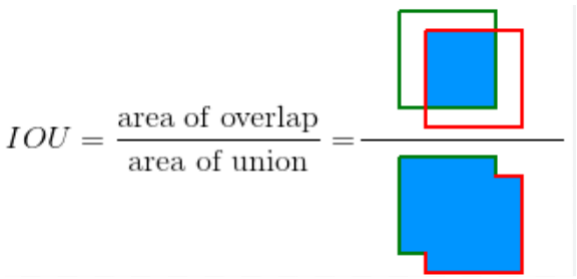
Sobre as arquiteturas de um estágio existe a *You Only Look Once* (YOLO) (REDMON et al., 2016), sendo a primeira arquitetura deste tipo na era do aprendizado profundo (ZOU et al., 2023). O modelo é caracterizado por dividir a imagem de entrada em diversas regiões e predizer para cada região *bounding boxes* e suas probabilidades de conter um objeto, simultaneamente. Embora tenha uma alta performance em termos de velocidade de processamento, sua primeira versão não se aplica com grande eficiência para objetos de pequeno porte. Evoluções recentes fizeram o modelo superar tais limitações, como suas versões subsequentes YOLOV3 (REDMON; FARHADI, 2018), YOLOV4 (BOCHKOVSKIY; WANG; LIAO, 2020) e (WANG; BOCHKOVSKIY; LIAO, 2023), sendo a última a YOLOV7, introduzindo estruturas otimizadas de reparametrização e atribuição dinâmica de rótulos.

2.3.3 Métricas de avaliação

A tarefa de detecção de objetos, possui diversas variações e possibilidades de diferentes abordagens para diferentes problemas. Dessa forma, algumas métricas e conceitos são estabelecidos a fim de realizar comparações para as soluções criadas (PADILLA; NETTO; SILVA, 2020).

A primeira métrica vastamente utilizada pelos modelos de detecção é a *Intersection over Union* (IOU), medida baseada no índice Jaccard (JACCARD, 1901) que mede um coeficiente de similaridade entre dois conjuntos de dados, Figura 7. Como demonstrado por Padilla, Netto e Silva (2020) a medida de IOU é utilizada para determinar se uma detecção é ou não correta, medindo a área de sobreposição entre a *bounding box* predita (B_p) e a *bounding box* verdadeira, denotada por B_{gt}, sendo calculada como a Equação 2.6. Assim, a partir de um limite estabelecido do valor retornado é determinado se uma detecção é ou não válida.

Figura 7 – *Intersection over Union* (IOU).



Fonte: Padilla, Netto e Silva (2020)

$$J(B_p, B_{gt}) = IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{2.6}$$

Com a validação se uma detecção é verdadeira, outras métricas são retiradas a partir do resultado da IOU. A precisão, descrita pela Equação 2.7 mostra a habilidade do modelo em realizar comente detecções relevantes, ou seja, somente detecções verdadeiras (PADILLA; NETTO; SILVA, 2020). Além da precisão, é tomado o*recall*, que mede a habilidade do modelo de detectar todos os casos positivos presentes no conjunto de dados avaliados, descrito pela Equação 2.8.

$$P = \frac{VP}{VP + FP} \tag{2.7}$$

$$R = \frac{VP}{VP + FN} \tag{2.8}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i, \tag{2.9}$$

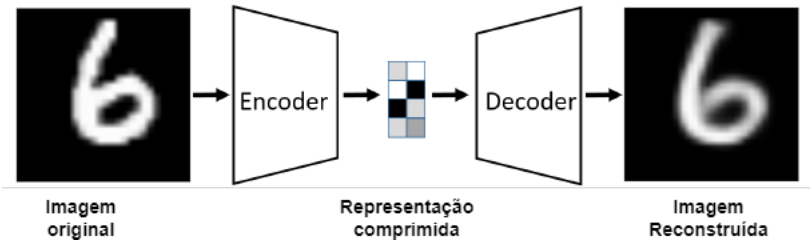
Outra métrica referencial para comparação entre modelos é a *Mean Average Precision* (mAP), Equação 2.9, a qual retira a média da precisão média dentre todas as N classes avaliadas. Usualmente, é utilizada duas variações, a primeira com IOU fixado em 0.5 e a segunda uma média de sua variação entre um IOU de 0.5 até 0.95, variando em passos de 0.05, permitindo uma avaliação mais acurada do modelo, (ZOU et al., 2023).

2.4 Autoencoders

Inicialmente, os AE eram em grande parte utilizados como um redutor de dimensionalidade de dados (BOURLARD; KAMP, 1988), (HINTON; ZEMEL, 1993). No entanto, recentes evoluções no campo da inteligência artificial trouxeram os AE para a vanguarda nas inteligência artificiais generativas e demais aplicações (GOODFELLOW; BENGIO; COURVILLE, 2016).

Como detalhado em Beggel, Pfeiffer e Bischl (2019), um AE é uma rede neural que mapeia uma imagem de entrada $x \in X = \mathbb{R}^n$ para uma imagem de saída $x^{'} \in X$. O modelo como um todo pode ser divido em dois blocos, ou duas funções, uma função *encoder* $f : X \to Z$ e uma função *decoder* $g : Z \to X$. Ambos os modelos em conjunto operam $x^{'} = g(f(x))$, obtém-se então a saída do *encoder* $z = f(x) \in \mathbb{R}^m (m << n)$ que é uma representação de baixa dimensionalidade de x, prevenindo que o modelo copie a imagem de entrada diretamente para saída e forçando-o com que aprenda uma uma representação com altos níveis de informação (*encoder*) e posteriormente retornar desta representação latente para a imagem original (*decoder*) (BANK; KOENIGSTEIN; GIRYES, 2023), como demonstrado na Figura 8.

Figura 8 – Represenação de um Autoencoder convolucional.



Fonte: Adaptado de Bank, Koenigstein e Giryes (2023)

Para o treinamento, Beggel, Pfeiffer e Bischl (2019) o define como a minimização do erro de reconstrução $L(x, x^{'})$ que em grande parte é utilizado o erro médio quadrático entre os pixeis ou a distância euclidiana. Assim, o uso de AE para a tarefa de detecção de anomalias tem como base que, após o treinamento, o modelo aprenderá o espaço latente para o exemplos classificados como normais, resultando em um baixo erro de reconstrução para tais exemplos e um alto erro de reconstrução para as anomalias.

2.5 Trabalhos Relacionados

Com objetivo de lidar com problemas de falsos positivos em sistemas detectores de armas de fogo, (VALLEZ; VELASCO-MATA; DENIZ, 2021) propõem a exposição do detector armas no seu ambiente de uso por um determinado período de tempo e, com os objetos comumente detectados como Falso Positivo (FP) realizar o treinamento de um filtro para ser adicionado após o detector. A adição do filtro se da pelo treinamento de um *Autoencoder* (AE) que fará o descarte dos objetos detectados como falsos alarmes.

Visando tomar a decisão se uma detecção deve ou não ser filtrada, os autores utilizaram três abordagens distintas. Primeiramente, foi utilizado o erro de reconstrução do AE, onde tem-se erros maiores para imagens Verdadeiro Positivo (VP), e erros menores para imagens FP. As duas demais técnicas se baseiam nos dados do vetor de representação central do AE que são utilizados para o treinamento de modelos de classificação como o *K-Nearest Neighbors* (KNN) e *Suport Vector Machines* (SVM) que possibilitam a diferenciação entre as duas classes de imagens.

Já para Beggel, Pfeiffer e Bischl (2019), o treino contínuo de AE pode reduzir o erro de reconstrução dos *outliers*, podendo dessa forma degradar sua performance como um detector de anomalias. A fim de minimizar tal problema, os autores propõem o uso dos *Adversarial Autoencoders* (AEE), os quais permitem impor a distribuição prévia sobre o vetor de representação central de tal maneira que as anomalias são colocadas em uma baixa região de probabilidade.

A partir do uso do critério de treino adversário (GOODFELLOW et al., 2020), os AEE permitem impor uma distribuição no seu espaço latente a partir da qual as amostras podem ser extraídas e ter suas saídas que variam suavemente de acordo as alterações neste mesmo espaço. Como a distribuição é imposta, é esperado uma separação no vetor central entre os dados anômalos e os dados usuais.

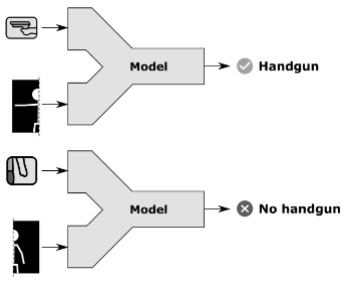
Em uma forma diferente de abordagem, (RUIZ-SANTAQUITERIA et al., 2021) além de servir-se de informações sobre a arma, os autores também visam utilizar informações sobre o corpo humano, como posições de articulações e membros como dados complementares para melhorar a performance do modelo. A hipótese levantada é que imagens obtidas por câmeras de vigilância muitas vezes possuem uma baixa qualidade, como baixa resolução, presença de ruídos e baixa luminosidade, além do objeto alvo estar muitas vezes longe das câmeras ou parcialmente coberto. Neste contexto, as informações oferecidas pela disposição do corpo portador da arma pode melhorar significativamente a detecção.

Inicialmente, é realizada a coleta de informações da pose do corpo humano através do *framework OpenPose*, gerando um conjunto de 25 *key-points* cada um com sua respectiva confiança. Após a coleta dos *key-points*, as regiões das mãos são então extraídas e alimentadas em um classificador binário para determinar se existe uma arma ou não na determinada região. Os autores utilizaram dois métodos distintos para realizar a detecção final, o primeiro, Figura 9, onde somente a região das mãos são utilizadas para o classificador, e o segundo, Figura 10, com a combinação da região extraída, bem como a pose do corpo em uma imagem mais completa.

Figura 9 – Classificador da região das mãos.

Fonte: Ruiz-Santaquiteria et al. (2021)

Figura 10 – Combinação da pose do corpo com a região das mãos.



Fonte: Ruiz-Santaquiteria et al. (2021)

3 Desenvolvimento

Neste capítulo será apresentado a elaboração geral do trabalho, a seção 3.1 apresentará a construção das bases de dados utilizadas. Na seção 3.2 apresentará todas as principais tecnologias utilizadas para desenvolvimento do trabalho, desde o treinamento até a avaliação do sistema como um todo. Os modelos de detecção utilizados serão descritos na seção 3.3 e por fim na seção 3.4 a construção e raciocínio de treinamento do AE.

3.1 Base da dados

Para realização do treinamento dos modelos e execução de experimentos comparativos sob diferentes tipos de imagens, duas fontes principais foram utilizadas para construção de dois conjuntos distintos.

A primeira base de dados utilizada é retirada do trabalho de Pérez-Hernández et al. (2020), sendo uma base de domínio público, publicada pela Universidade de Granada, contendo cerca de 3000 imagens. Relatado pelo autores, as imagens selecionadas foram extraídas internet com uma ou mais aparições de arma de fogo em diversas situações, incluindo vídeos de segurança, oferecendo um rico contexto de tipos de *background*. O objetivo principal ao utilizar esta fonte se da pela observação da performance dos modelos treinados com somente exemplos positivos do objeto que se deseja detectar.

Tabela 1 – Distribuição da base de dados da universidade de granada

Métrica	Contagem
Número de imagens	2971
Imagens com anotações	2971
Quantidade de anotações	3432

Fonte: Produzido pelos autores.

Como observado na Tabela 1, todas as imagens possuem pelo menos uma anotação, contendo somente uma classe de arma. Além da checagem da distribuição dos dados, outras métricas foram retiradas a fim de realizar um estudo sobre os dados utilizados. Observando as medidas da Tabela 2, nota-se que a a área média das *bouding boxes* (a área média obtida representa uma medida direta dos dados, e não uma multiplicação da largura e altura média) representa cerca de 38% da área total da imagem, dado as imagens utilizadas com dimensões de 640 x 640 pixels.

Tabela 2 – Estatísticas (em pixels) sobre as *bouding boxes*

Métrica	Largura	Altura	Área
Média	345	349	156864
Desvio padrão	215	198	141129

Fonte: Produzido pelos autores.

Para a segunda base de dados, tem-se como o objetivo a avaliação dos modelos utilizados sob dados com objetos similares as armas de fogo, oferecendo um contexto parecido para o modelo. Assim, foram coletados os dados da base pulicada por Pérez-Hernández et al. (2020). Nomeada por *Sohas detection*, o conjunto é formado por armas e pequenos objetos que são manuseados de maneira similar às armas, incluindo seis diferentes categorias, como armas de fogo, facas, cédulas de dinheiro, carteira/bolsa, cartões e celulares, seguindo a distribuição da Tabela 3.

Tabela 3 – Distribuição da base de dados *Sohas detection*

Arma	Faca	Celular	Cédulas de dinheiro	Carteira	Cartões	Número de imagens
1510	2277	715	477	600	279	5838

Fonte: Produzido pelos autores.

Tabela 4 – Estatísticas (em pixels) sobre as *bouding boxes* da base *Sohas detection* para classe de armas

Métrica	Largura	Altura	Área
Média	375	382	178658
Desvio padrão	209	187	141013

Fonte: Produzido pelos autores.

Observando a Tabela 4, nota-se que os objetos de armas de fogo são suavemente maiores quando comparados a primeira base estudada, tendo suas imagens também em dimensões 640 x 640, a área ocupada pelas armas é cerca 43%. Dessa forma, para concretizar a estruturação do segundo conjunto de dados, as imagens coletadas da universidade de Granada foram inseridas no conjunto de *Sohas detection* a fim de aumentar a amostragem do objeto que se deseja detectar, resultando na base detalhado por Tabela 5.

Como resultado final, obteve-se os dois conjuntos de dados, o primeiro, sumarizado na Tabela 1 contendo somente exemplos positivos de armas de fogo, e o segundo, sumarizado na Tabela 5, contendo amostras de objetos similares e sendo utilizados em contextos

Tabela 5 – Distribuição da base de dados com formada pelo conjunto de Granada e *Sohas detection.*

Arma	Faca	Celular	Cédulas de dinheiro	Carteira	Cartões
4942	1879	755	545	571	340

Fonte: Produzido pelos autores.

análogos. Ambas as bases foram divididas em percentuais de 70% para treinamento, 20% para validação e 10% para etapa de teste, sendo que para a segunda base de dados, a divisão foi realizada de forma a preservar a distribuição das classes.

3.2 Tecnologias utilizadas

O desenvolvimento do trabalho foi realizado através da linguagem de programação *Python*[1], que fornece legibilidade de código, diversas estruturas de alto nível, uma vasta coleção de módulos internos e *frameworks* externos para lidar com diferentes tipos de problemas (BORGES, 2014).

Como ambiente de desenvolvimento, utilizou-se o *Google Colaboratory*[2], que se caracteriza como um serviço hospedado do *Jupyter Notebook* que requer pouca ou nenhuma configuração para uso, além de acesso grátis à recursos como *Graphics Processing Unit* (GPU), especialmente construído para a prototipação e desenvolvimento de ciência de dados e aprendizagem de máquina. Com os serviços oferecidos pela plataforma, os modelos foram treinados em uma GPU NVIDIA Tesla T4 com 16GB de memória VRAM e o drive CUDA pré-instalado na versão 12.2.

Para o treinamento de modelos como SSD (LIU et al., 2016) e *RetinaNet* (LIN et al., 2017b), foram realizados a partir do *framework MMDetection*[3] (CHEN et al., 2019), ferramenta de código aberto com características como arquitetura modular, suporte para múltiplas tarefas de visão computacional, alta eficiência e disponibilidade de dezenas de diferentes modelos já treinados sob diversos conjuntos de dados de referência para a área. Para o treinamento da YOLOV7, foi utilizado seu código fonte diretamente do seu repositório [4].

Em conjunto com os principais *frameworks* utilizados, sendo o *PyTorch*[5] para trabalhar com os modelos do *MMDetection* e o *TensorFlow*[6] para implementação do AE, foram utilizados o *OpenCV* para o processamento das imagens para entrada do AE,

[1] <https://www.python.org/>
[2] <https://colab.google/>
[3] <https://github.com/open-mmlab/mmdetection>
[4] <https://github.com/WongKinYiu/yolov7>
[5] <https://pytorch.org/>
[6] <https://www.tensorflow.org/?hl=pt-br>

o *Scikit-learn* para estimação da densidade do kernel do AE, e a plataforma *Weight & Biases*[7] para o rastreio dos modelos treinados e suas métricas associadas. Como ferramenta auxiliar para computar as métricas a partir do uso do filtro de falsos positivos, fez-se uso do *software* construído por Padilla et al. (2021), ferramenta com interface visual para cálculo de métricas ligadas a detecção de objetos.

3.3 Modelos de detecção

3.3.1 YOLO V7

3.3.1.1 Estrutura do modelo

O modelo YOLO V7 foi utilizado como a arquitetura principal do trabalho desenvolvido, sendo o detector base para posteriormente ter suas saídas filtradas pelo AE. Como descrito por Wang, Bochkovskiy e Liao (2023), o modelo é desenhado para um processamento em tempo real, se destacando pela sua velocidade de trinamento, inferência, acurácia e adaptabilidade em uma ampla gama de aplicações, tais métricas são oriundas da redução de cerca de 40% da quantidade de parâmetros de treinamento e 50% de custo computacional, superando os modelos estado da arte para detecção em tempo real. Algumas de suas principais características serão descritas a seguir.

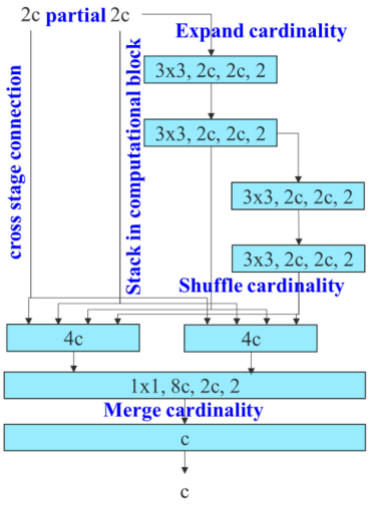
Como *backbone*, a YOLO V7 faz utilização da *Extended Efficient Layer Aggregation Netowork* (E-ELAN) Figura 11, que utiliza a expansão, embaralhamento e mesclagem da cardinalidade dos blocos computacionais, permitindo uma habilidade de aprendizado contínua sem a degradação do fluxo do gradiente.

Diferentes abordagens para o dimensionamento do modelo, como diferentes aplicações requerem diferentes modelos, priorizando acurácias em alguns casos e velocidade em outros, o dimensionamento de modelo tem por objetivo prover o encaixe dessas características em determinados tipos de dispositivos computacionais. Em grande parte dos casos, o dimensionamento considera seus parâmetros de maneira isolada, como resolução da imagem de entrada, número de canais, número de camadas ou quantidade de pirâmides de atributos. Para o modelo, os autores mostram como esse dimensionamento pode ser realizado levando em consideração todos esses parâmetros juntos. Mantendo as propriedades originais do modelo e sua estrutura ótima.

Os autores fazem uso também do denominado *Bag of Freebies*, introduzidos na YOLO V4 (BOCHKOVSKIY; WANG; LIAO, 2020), que são métodos que permitem aumentar a performance do modelo sem o aumento do custo de treinamento. Entre os métodos, estão a reparametrização a nível de modelo, onde o treino é divido em dois módulos e suas saídas são agrupadas para obter a saída final, é aplicado também métodos

[7] <https://wandb.ai/site>

Figura 11 – Arquitetura geral da E-ELAN.



Fonte: Wang, Bochkovskiy e Liao (2023)

para auxiliar a detecção nas camadas intermediárias pela *auxiliary head* e das camadas finais pela *Lead Head*, o mecanismos é denominado pelos autores como *coarse for auxiliary and fine for lead loss*, na qual a perda é calculada para ambas as camadas de detecção baseada na saída da *Lead Head*, que possuem uma capacidade de aprendizagem maior.

3.3.1.2 Treinamentos

O primeiro experimento do modelo foi realizado sob o base de dados publicada pela Universidade de Granada, Tabela 1, com objetivo de avaliação de sua performance através de imagens com somente exemplos positivos de armas de fogo (cada imagem possuem pelo menos um objeto demarcado). Com a divisão da base de dados em 70% para treinamento, 20% para validação e 10% para etapa de teste.

Para primeiro experimento realizado, sofreram alterações apenas o número de épocas de treinamento e a quantidade de épocas para o *checkpoint*. Os principais hiperparâmetros da rede, junto com as modificações realizadas são detalhados na Tabela 6.

Com objetivo de avaliar a performance com a modificação de seus hiperparâmetros, um segundo treinamento foi realizado. Para este, o hiperparâmetro, *focal loss*, inicialmente desativado, foi setado para o valor de 1.5, permitindo com o que o modelo aplique tal função de perda durante a fase de treino.

A *focal loss*, (LIN et al., 2017b), é desenhada para lidar com as dificuldades enfrentadas pelos detectores de um estágio que são extremamente desbalanceados durante o treinamento para classificação das classes de *foreground* e *background*. Para endereçar tal problema, os autores realizaram uma modificação na função de custo da entropia cruzada

Tabela 6 – Hiperparâmetros para o experimento 1

Hiperparâmetros	Valor padrão	Alteração
Épocas	300	200
Taxa de aprendizagem	0.01	-
Otimizador	SGD	-
Momentum	0.937	-
Batch size	16	-
Focal loss gamma	0	-
checkpoint	-1	50

Fonte: Produzido pelos autores.

Equação 3.1.

$$\mathrm{EC}(p, y) = \begin{cases} -\log(p) & \text{se } y = 1 \\ -\log(1-p) & \text{caso contrário} \end{cases} \tag{3.1}$$

A modificação se da pela adição do termo $\alpha_{p_t}(1-p_t)^{\gamma}$, resultando na Equação 3.2. O parâmetro principal de controle sendo $\gamma > 0$ reduz a perda para os exemplos bem classificados, colocando um maior foco nos exemplos mais difíceis e onde há um maior número de erros.

$$FL(p_t) = -\alpha_{p_t}(1-p_t)^{\gamma} log(p_t) \tag{3.2}$$

Definindo p_t como:

$$\mathrm{p}_t = \begin{cases} p & \text{se } y = 1 \\ 1-p & \text{caso contrário} \end{cases} \tag{3.3}$$

Dando o foco do trabalho na avaliação de modelos sob perspectivas de detecção de falsos positivos, o terceiro treinamento seu deu através da união das bases de dados da Universidade de Granada e a base nomeada de *sohas detection* Tabela 3. A união dos dois conjuntos tem como resultados a Tabela 5.

Para a realização do treinamento e avaliação dos objetos similares ao que se deseja detectar, todas as anotações relacionadas as classes adjacentes como faca, celular, cédulas de dinheiro, carteira e cartões foram removidas, permanecendo apenas as anotações relativas as armas de fogo. Anteriormente a remoção das anotações, fez-se a divisão das imagens entre treino, validação e teste de forma a preservar a distribuição de todas as classes. Devido aos resultados que serão apresentados no Capítulo 4, os hiperparâmetros utilizados

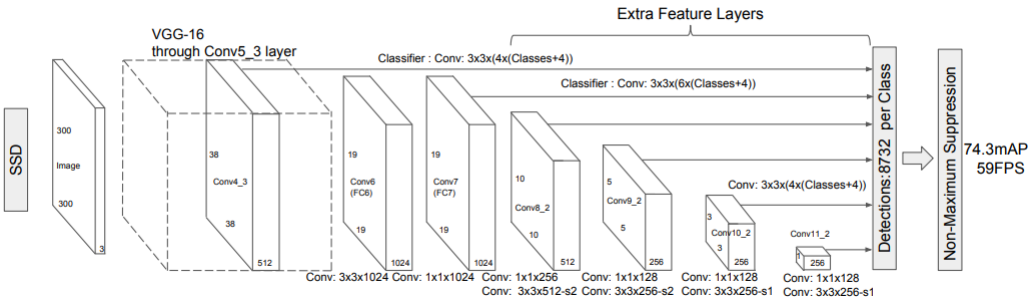
para este experimento são referentes a Tabela 6, ou seja, sem a utilização da *Focal Loss* como função de perda.

3.3.2 Detectores para comparação

Objetivando estabelecer métricas de comparação para o modelo principal utilizado (YOLO V7), alguns do detectores de um estágio presentes na literatura foram escolhidos. Para todos os modelos de comparação, o conjunto de dados utilizados para o treinamento foi o coletado da Universidade de Granada Tabela 1, assim, pode-se verificar posteriormente suas performances sob dados com somente imagens de armas de fogo e testados com a inserção de imagens com potenciais falsos positivos.

- **SSD**: O primeiro modelo utilizado foi o *Single Shot MultiBox Detector* (SSD), como descrito sobre os modelos de um estágio, essa arquitetura realiza a predição das *bounding boxes* e de suas respectivas classes de forma simultânea. Detalhado na Figura 12, o modelo consiste na produção de uma coleção de *bounding boxes* de tamanho fixo através de uma estrutura base, sendo utilizada pelos autores a VGG-16 (SIMONYAN; ZISSERMAN, 2014) como extratora de características, seguida por uma série de camadas convolucionais especializadas para detecção de objetos e por fim aplicação do *non-maximum supression* para seleção das detecções.

Figura 12 – Arquitetura geral SSD.



Fonte: Liu et al. (2016)

Uma de suas principais características é a realização da detecção a partir de mapas de características de diversos tamanhos, formados em diferentes etapas da rede. Como saída, são obtidos 8732 predições para cada classe. Visando lidar com o efeito do desbalanceamento entre as classes positivas e negativas, os autores realizam um filtro para selecionar as predições de maior confiança, obtendo uma proporção de três exemplos negativos para um positivo ao fim do modelo.

Suas variações de arquitetura consistem especialmente no tamanho da imagem de entrada, passando pelas dimensões de 300 x 300 e 512 x 512. Como relatado pelos

Tabela 7 – Hiperparâmetros utilizados para a SSD

Hiperparâmetros	Valor padrão	Alteração
Épocas	-	20
Taxa de aprendizagem	000.2	-
Otimizador	SGD	-
Momentum	0.9	-
Batch size	8	4

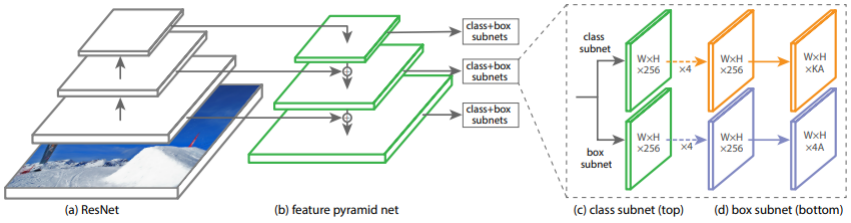
Fonte: Produzido pelos autores.

autores a última variação possui uma melhor performance para detecção de objetos pequenos e de diferentes escalas, sendo essa assim, escolhida para o presente trabalho. Ademais os hiperparâmetros utilizados são especificados na Tabela 7.

- **RetinaNet**

  A RetinaNet tem como uma de suas principais funcionalidades a função de perda *focal loss*, descrita pela Equação 3.2 com o objetivo de aumentar a assertividade de detectores de um estágio devido ao desbalanceamento entre os exemplos positivos e negativos ao fim do processamento da rede.

Figura 13 – Arquitetura geral RetinaNet.



Fonte: Lin et al. (2017b)

Detalhado na Figura 13, sua arquitetura é composta por uma rede residual profunda (HE et al., 2016) como extrator de características, seguida por uma FPN e dois tipos de sub-redes: a sub-rede de classificação para predição da probabilidade de um objeto para cada âncora, composta com por 4 camadas de convolução 3 x 3; e a sub-rede de regressão das *bounding boxes*, construída em paralelo e de forma idêntica a sub-rede de regressão.

Assim como os demais modelos treinados para fins comparativos, as alterações dos hiperparâmetros da RetinaNet Tabela 8 foram realizados para permitir seu treinamento em uma memória reduzida e sob o conjunto de dados específico. Demais variações

Tabela 8 – Hiperparâmetros utilizados para RetinaNet

Hiperparâmetros	Valor padrão	Alteração
Épocas	-	20
Taxa de aprendizagem	000.2	-
Otimizador	SGD	-
Momentum	0.9	-
Batch size	8	4
Gamma	2	-
Alfa	0.25	

Fonte: Produzido pelos autores.

de sua arquitetura são relativas a profundidade de seu extrator de características, variando entre 18, 50 e 101 camadas de profundidade, sendo utilizada aqui a variação de 50 camadas para ganho de assertividade sem perda de performance do modelo quanto a complexidade computacional.

3.4 Autoencoder

Visando a aplicação e avaliação geral do funcionamento do AE como um possível filtro de falsas detecções após a saída da rede utilizada como detector base tem-se a seguinte abordagem. A detecção de falsos positivos se trata, primordialmente na coleta das imagens em que a rede base detecta um objeto de maneira errônea e realizar o treinamento do AE com estes exemplos a fim de que o mesmo aprenda seus padrões de distribuição de pixels.

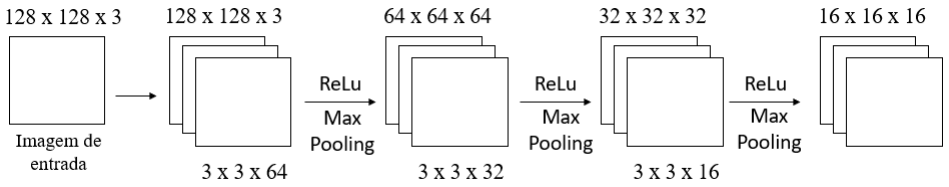
Dessa maneira, a estruturação do treinamento do modelo se da partir dos dados de potenciais falsos positivos descritos na Tabela 3 com a exclusão dos exemplos de armas de fogo. Devido a baixa complexidade do modelo, apenas os conjuntos de validação e teste foram usados, como treino e validação respectivamente. A estrutura utilizada é composta de dois módulos, um *encoder*, responsável pela compressão dos dados em uma dimensionalidade menor em relação à entrada e o *decoder*, com papel de a partir da dimensão reduzida redimensionar os dados comprimidos para a representação real, com objetivo de se aproximar da imagem de entrada.

Para o primeiro módulo, sua estrutura é composta por três camadas convolucionais com filtros de dimensões 3 x 3, seguidos por uma função de ativação ReLu Equação 3.4 e uma camada *max pooling* (reduzindo as dimensões de saída pela metade), que a partir de uma imagem, de entrada de 128 x 128 x 3 produz uma saída de 16 x 16 x 16. A profundidade do mapa de características, inicialmente de 64, também é reduzida juntamente com as

dimensões da imagem, como ilustrado na Figura 14.

$$f(x) = \max(0, x) \tag{3.4}$$

Figura 14 – Arquitetura do encoder.

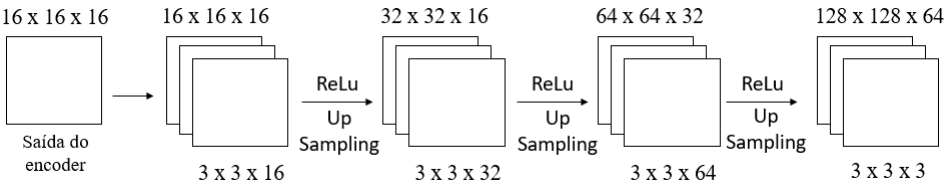


Fonte: Produzido pelos autores

O segundo módulo (*decoder*), possui uma estrutura semelhante ao *encoder*. Agora, cada camada convolucional é seguida é por uma camada de *upsampling*, a qual tem objetivo de aumentar as dimensões de entrada. Por fim, é adicionada uma camada convolucional a mais, trazendo a saída do modelo final para as dimensões da imagem de entrada de 128 x 128 x 3, ilustrado pela Figura 15 dessa vez seguido pela função de ativação *sigmoid* Equação 3.5.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

Figura 15 – Arquitetura do decoder.



Fonte: Produzido pelos autores

A estrutura completa é composta por 52.067 parâmetros treináveis, sendo robusta e ao mesmo tempo eficiente para treinamento e inferência, sem a necessidade de um grande número de exemplos para treinamento e sem acrescer grande complexidade computacional ao sistema como um todo. Para o treinamento, foram utilizados o otimizador Adam (KINGMA; BA, 2014), a função de perda de erro quadrático médio Equação 3.6, taxa de aprendizagem de 0.001, *batch size* de 8 durante 50 épocas.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.6}$$

Para aplicação do filtro de falsos positivos com uma maior robustez, como relatado por Beggel, Pfeiffer e Bischl (2019), além do uso do erro de reconstrução, também será usado a distribuição do vetor de baixa dimensionalidade, que, segundo os autores, pode constituir uma separação clara na probabilidade dos exemplos positivos e negativos.

4 Resultados

Neste capítulo, serão apresentados os resultados obtidos a partir dos treinamentos dos modelos e a aplicação do filtro de falsos positivos. Na seção 4.1 serão descritos os resultados de cada treinamento dos modelos de detecção, e os resultados obtidos com o treinamento e aplicação AE na seção 4.2.

4.1 Performance dos modelos de detecção

4.1.1 Modelos de comparação

Para os modelos que serão usados com objetivos de comparação de performance, sendo eles o SSD e a *RetinaNet*, ambas arquiteturas utilizadas também nos trabalhos de referência, a base de dados utilizada é a base da Universidade de Granada, composta somente por armas de fogo.

4.1.1.1 SSD

Para a SSD, o modelo foi treinado utilizando os hiperparâmetros descritos na Tabela 7, ainda que utilizando sua configuração padrão, o modelo apresenta uma performance satisfatória com uma precisão de 0.877, levando a uma taxa de falsos positivos próxima a 10% de suas detecções e cerca de 30% de falsos negativos. Ou seja, a cada 100 detecções do modelo, potencialmente 13 serão falsas, para *recall*, a cada 100 detecções que deveriam ser detectadas, cerca de 70 seriam de fato realizadas.

Tabela 9 – Performance SSD

Conjunto	Precisão	Recall	mAP@0.5	mAP@0.5:0.95
Granada	0.877	0.707	0.734	0.658

Fonte: Produzido pelos autores.

4.1.1.2 RetinaNet

O modelo da *RetinaNet* foi treinado sobre o o conjunto de hiperparâmetros da Tabela 8. Os resultados do modelo, descritos na Tabela 10 apresentam resultados bastante similares aos resultados referentes da SSD, divergindo em poucas casa decimais para as métricas selecionadas e possuindo uma performance robusta para um objetivo de comparação.

Tabela 10 – Performance RetinaNet

Conjunto	Precisão	Recall	mAP@0.5	mAP@0.5:0.95
Granada	0.865	0.715	0.754	0.657

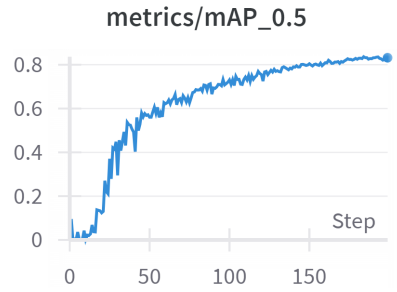
Fonte: Produzido pelos autores.

4.1.2 Yolo V7

Como relatado no Capítulo 3, as variações principais dos modelos treinados são referentes a arquitetura YOLO V7, sendo realizado em primeiro momento um treinamento com seus hiperparâmetros padrões, em segundo momento a utilização de função de perda *focal loss*, ambos sobre a base de dados descrita na Tabela 1. Por último, a partir da melhor configuração do modelo, realizou-se um terceiro treinamento sobre os dados descritos na Tabela 5.

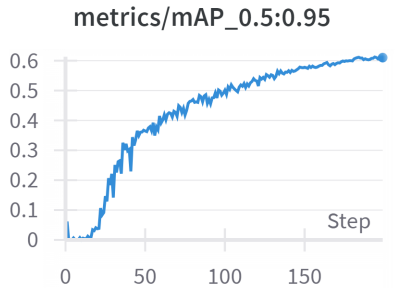
Os resultados do primeiro experimento conduzido sob a base dados da Tabela 1 e com os hiperparâmetros descritos pela Tabela 6 apresentaram uma performance satisfatória. Analisando a Figura 16, pode-se observar como o mAP, para um limite de IOU de 0.5 já se estabiliza a partir de uma média de 0.8 ao fim do treinamento. Em uma medida mais robusta, variando o limite de IOU de 0.5 até 0.95, Figura 17, o mAP atinge um valor máximo por volta de 0.6.

Figura 16 – mAP com IoU de 0.5, exp. 1



Fonte: Produzido pelos autores

Figura 17 – mAP com IoU variando de 0.5 à 0.95, exp. 1

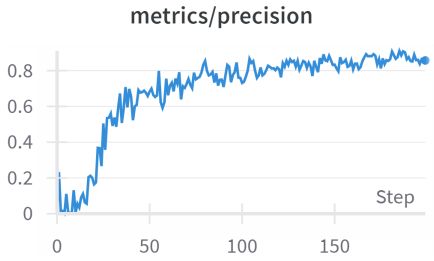


Fonte: Produzido pelos autores

Observando as métricas de precisão e *recall*, vemos que o modelo possui uma maior deficiência em detectar as classes positivas presente no conjunto de dados. O gráfico da Figura 18 nos mostra uma precisão próxima de 0.8, ou seja, a cada 10 detecções do modelo, cerca de 8 estarão corretas. Já a Figura 19 nos mostra um *recall* próximo próximo de 0.7, elucidando a presença de Falso Negativo (FN), classes positivas na base de dados que o modelo deixa de detectar.

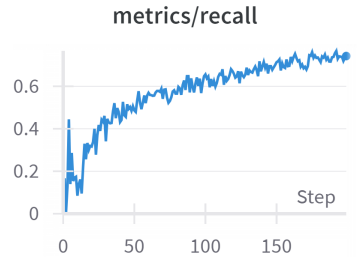
O teste do modelo foi realizado em duas etapas, o primeiro sob o conjunto de teste oriundo da base de dados da Universidade de Granada. Posteriormente, o mesmo

Figura 18 – Precisão, exp. 1



Fonte: Produzido pelos autores

Figura 19 – Recall, exp. 1



Fonte: Produzido pelos autores

modelo foi executado com a base de teste proveniente da união do primeiro conjunto com o *sohas detection* Tabela 5, esta, contendo exemplos de objetos similares as armas de fogo. A Tabela 11 sumariza os resultados obtidos. Nota-se como o modelo diminui sua precisão (aumento de detecção de FP) a partir da presença de imagens com objetos semelhantes ao objeto alvo, é importante ressaltar que, para esse modelo, tais imagens não foram previamente vistas na etapa de treino, sendo inseridas somente na etapa de teste. As imagens ilustradas pelas Figura 20 e Figura 21 contém exemplos de um subconjunto da base de teste e a predição realizada pelo modelo. Já as Figura 22 e Figura 23 ilustram a performance do modelo sob o conjunto com potenciais falsos positivos.

Tabela 11 – Resultado dos testes do primeiro modelo treinado sob conjuntos de dados com e sem potenciais falsos positivos

Conjunto	Precisão	Recall	mAP@0.5	mAP@0.5:0.95
Granada	0.932	0.745	0.728	0.598
Granada e Sohas	0.831	0.761	0.73	0.602

Fonte: Produzido pelos autores.

O segundo experimento baseou-se na utilização da *focal loss* como função de perda do modelo, colocando o hiperparâmetro $\gamma = 1.5$. Os gráficos de precisão Figura 26, *recall* Figura 27 e mAP Figura 24 Figura 25, nos mostra como a performance do modelo se estabiliza em métricas inferiores as do modelo com hiperparâmetros padrões.

Com a Tabela 12 sumariza seus resultados de teste somente na base sem falsos positivos em potencial. Apesar da manutenção na boa performance no que diz respeito a precisão, as demais métricas como *recall* e mAP obtiveram uma alta degradação com a utilização da *focal loss* com $\gamma = 1.5$. É importante ressaltar que, a performance observada não implica em uma generalização de resultados para a função de perda em si utilizada, sendo essa flexível para uma ampla combinação de seus hiperparâmetros que, se exauridos, podem resultar em resultados mais satisfatórios. Devido às métricas inferiores, e limitações

Figura 20 – Subconjunto da base de teste - labels

Fonte: Produzido pelos autores

Figura 21 – Subconjunto da base de teste - predição

Fonte: Produzido pelos autores

Figura 22 – Subconjunto de teste com potenciais FP - labels

Fonte: Produzido pelos autores

Figura 23 – Subconjunto de teste com potenciais FP - predição

Fonte: Produzido pelos autores

computacionais para otimização dos hiperparêmtros, testes relativos aos falsos positivos com essa configuração não foram executados.

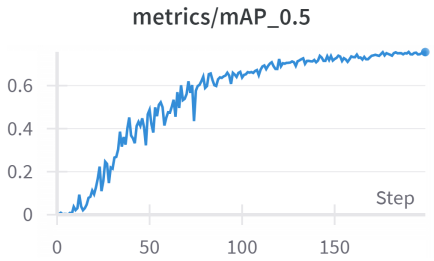
Tabela 12 – Resultado dos testes do modelo com utilização da *focal loss*

Conjunto	Precisão	Recall	mAP@0.5	mAP@0.5:0.95
Granada	0.924	0.49	0.473	0.393

Fonte: Produzido pelos autores.

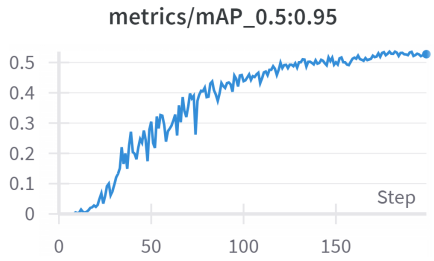
Como último passo, foi realizado o experimento sob o conjunto de dados resultante da união entre as duas bases utilizadas, Tabela 3 e Tabela 1, resultando nos dados detalhados em Tabela 5. O objetivo principal desta etapa está em tornar visível os objetos tratados com potenciais falsos positivos para o modelo durante a fase de treinamento e

Figura 24 – mAP: IoU = 0.5, exp. 2



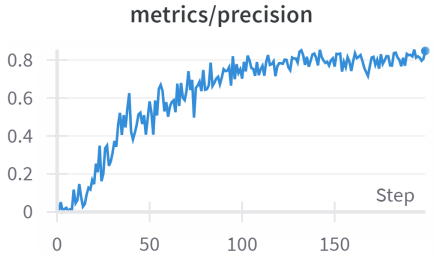
Fonte: Produzido pelos autores

Figura 25 – mAP: IoU variando de 0.5 à 0.95, exp. 2



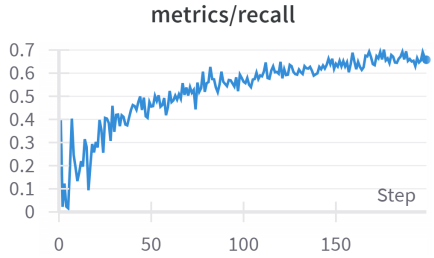
Fonte: Produzido pelos autores

Figura 26 – Precisão, exp. 2



Fonte: Produzido pelos autores

Figura 27 – Recall, exp. 2



Fonte: Produzido pelos autores

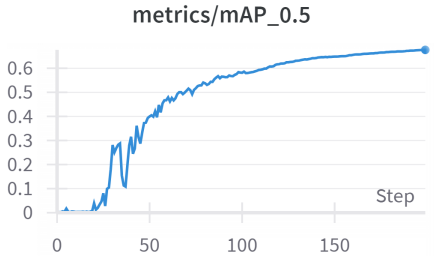
observar o impacto em sua performance. Para este treinamento, devido aos resultados anteriores foram utilizados os hiperparâmetros da Tabela 6.

Analisando as evoluções da precisão, *recall* e mAP, destaca-se como as curvas de mAP, Figura 28 e Figura 29, possuem uma rápida convergência, atingindo um platô em épocas distantes do fim do treinamento. A curvas de precisão Figura 30 e *recall* Figura 31 apresentam ainda um comportamento instável no início do treinamento, podendo ter como causa o tamanho usado de *batch* de imagens devido aos limites computacionais enfrentados. É notável que com a aplicação do novo conjunto de dados torna-se necessário a otimização dos hiperparâmetros deste novo modelo, no entanto, para o presente trabalho é necessário a comparação dos modelos utilizando as mesmas configurações, variando somente a base de dados. Uma abordagem que pode ser utilizada posteriormente é a otimização de ambos (primeiro e último modelo) em sua melhor configuração de hiperparâmetros possível, que faz-se necessário de uma alta capacidade computacional para atingir tais objetivos.

Os resultados executados sobre a base de teste, que já é populada com potenciais falsos positivos são detalhados na Tabela 13.

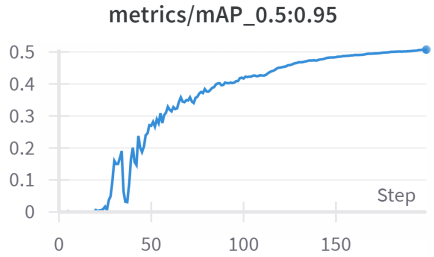
Com os dados do treinamento e teste aqui apresentados, destaca-se dois pontos principais: para o primeiro modelo submetido a imagens similares sem antes terem sido

Figura 28 – mAP: IoU = 0.5, exp. 3



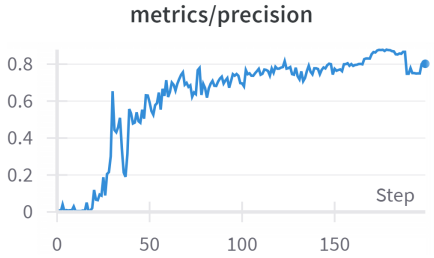
Fonte: Produzido pelos autores

Figura 29 – mAP: IoU variando de 0.5 à 0.95, exp. 3



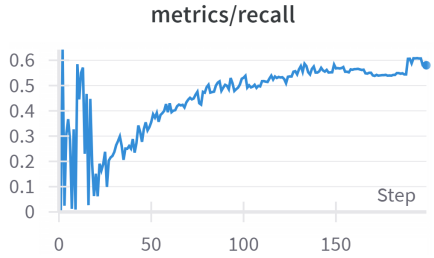
Fonte: Produzido pelos autores

Figura 30 – Precisão, exp. 3



Fonte: Produzido pelos autores

Figura 31 – Recall, exp. 3



Fonte: Produzido pelos autores

Tabela 13 – Resultado dos testes do modelo treinado com potenciais falsos positivos

Conjunto	Precisão	Recall	mAP@0.5	mAP@0.5:0.95
Granada e Sohas	0.789	0.618	0.702	0.51

Fonte: Produzido pelos autores.

processadas na fase de treinamento, o impacto é direto em sua assertividade, especialmente na queda da métrica de precisão que ressalta o aumento na detecção de FP; para o segundo caso analisado, onde objetos similares estão presente nos conjuntos de treino, validação e teste, utilizando a mesma configuração de hiperparâmetros para uma comparação justa, nota-se uma queda ainda maior nas métricas de assertividade, como exposto na Tabela 13.

4.2 Performance do autoecoder

A estratégia de treinamento e aplicação do AE é embasada no treinamento sobre os dados de potenciais falsos positivos. Devido a pouca quantidade de parâmetros do modelo, foram separados os conjuntos de validação e teste da base de dados de *sohas*, para serem utilizados como treino e validação, respectivamente, retirando-se de ambos os conjuntos os

exemplos de arma de fogo e permanecendo apenas com os potenciais falsos positivos. O treinamento do modelo criado detalhado nas Figura 14 e Figura 15 foi efetuado a partir do dos hiperparâmetros detalhados na Tabela 14 e sob os dados descritos na Tabela 15.

Tabela 14 – Hiperparâmetros do autoencoder

Hiperparâmetro	Valor
Épocas	80
Learning rate	0.001
Otimizador	Adam
Batch size	32
Função de perda	MSE

Fonte: Produzido pelos autores.

Tabela 15 – Dados usados para o treinamento do autoencoder

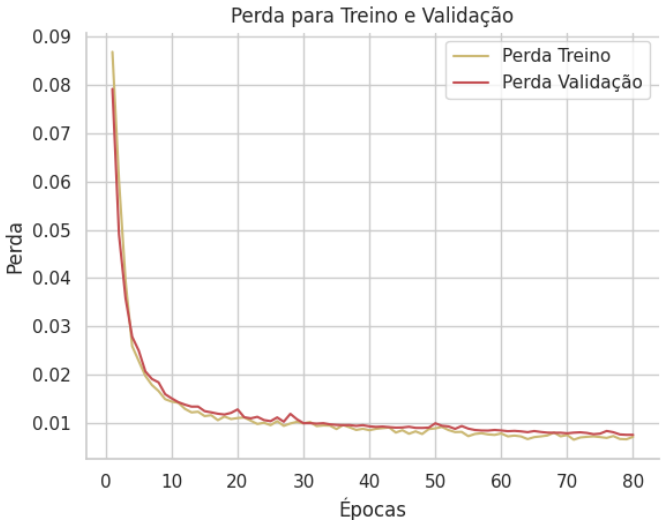
Conjunto	Treino	Validação
Contagem	859	428

Fonte: Produzido pelos autores.

Como resultado do treinamento, pode-se observar pela Figura 32 como ambas as perdas de treino e validação convergem de forma rápida até o fim do treinamento. Para avaliar a qualidade da reconstrução do AE, podemos analisar a Figura 33 que mostra um exemplo de imagem entrada tendo uma faca como objeto central e a reconstrução realizada pelo modelo. As imagens com exemplos de armas de fogo foram extraídas do conjunto com falsos positivos para avaliação da reconstrução de objetos ditos anômalos. A Figura 34 ilustra a reconstrução do modelo para um exemplo que deve ser tratado como um objeto anômalo.

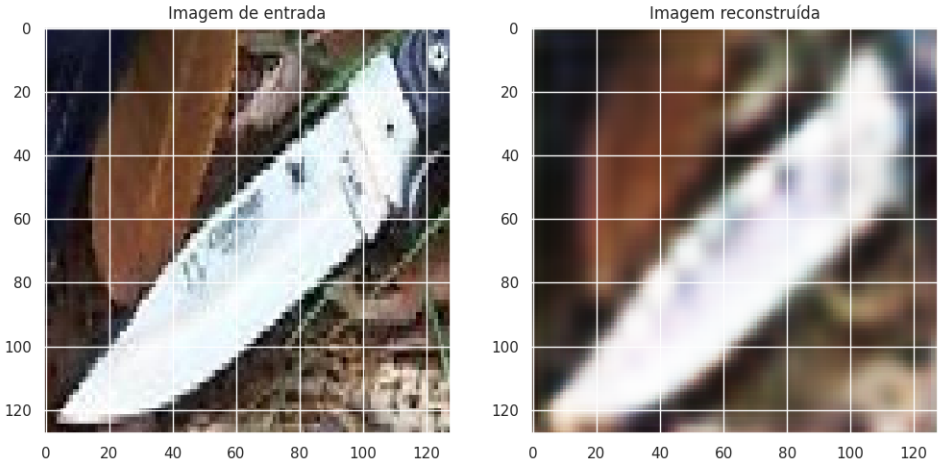
Através do conjunto de treino e do conjunto de teste, o qual é composto por imagens anômalas, é possível avaliar o erro de reconstrução médio para ambos os tipos de imagens, descritos pela Tabela 16, nota-se que o erro de reconstrução para imagens que serão tratadas como anomalias é cerca de 1.34x maior do que as imagens que o AE foi treinado. É possível observar também que, devido ao desvio padrão de ambos as categorias de imagens, ainda existe um faixa de sobreposição de erros de reconstrução, podendo se

Figura 32 – Reconstrução do autoencoder para imagens de treino.



Fonte: Produzido pelos autores.

Figura 33 – Reconstrução do autoencoder para imagens de treino.



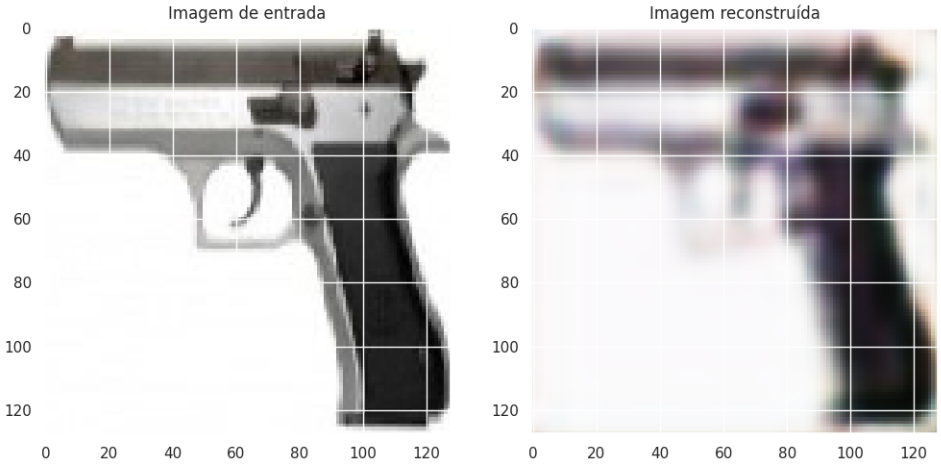
Fonte: Produzido pelos autores.

tornar sensível a escolha de um *threshold* para descriminar o que são ou não imagens anômalas.

O passo seguinte envolve a aplicação do *Kernel Density Estimation* (KDE)[1] sobre o vetor de distribuição comprimida usando uma *kernel* gaussiano. A técnica é aplicada

[1] <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KernelDensity.html>

Figura 34 – Reconstrução do autoencoder para imagens anômalas.



Fonte: Produzido pelos autores.

Tabela 16 – Erro médio de reconstrução

	Imagens de treino	Imagens anômalas
Média	0.0067	0.009
Desvio padrão	0.0043	0.005

Fonte: Produzido pelos autores.

sobre a saída do *encoder* após o encaixe no conjunto de dados de treinamento, estimando dessa forma sua distribuição de probabilidade. Com objetivo de determinar limites entre imagens caracterizadas como anomalias ou não, foi selecionado um subconjunto de 32 imagens de treino e teste, sendo o conjunto de teste composto por imagens anômalas, para cada imagem dos conjuntos seu *score* é calculado mediante o KDE e é retirado então a média e desvio padrão sobre o conjunto de 32 *scores* das imagens de treino e teste.

Tabela 17 – Medidas de densidade do kernel

	Imagens de treino	Imagens anômalas
Densidade média	2821.53	2428.38
Desvio padrão	$1.14 \cdot 10^{-11}$	406.97

Fonte: Produzido pelos autores.

Através da Tabela 17, é notável a diferença de medias entre as imagens anômalas ou não. Para a imagens sem presença de armas temos um desvio padrão dos *scores* atribuídos pelo KDE bastante próximo de zero. Utilizando o vetor latente de compressão nota-se uma

maior margem entre a densidade média de imagens anômalas e imagens não anômalas. Contudo a variação do *score* das imagens anômalas ainda é alta, fazendo com que haja uma faixa de sobreposição de valores e dificultando a definição de um *threshold*.

Durante a aplicação do AE como filtro, notou-se uma grande variabilidade dos erros de reconstrução e densidade das detecções advindas das do modelo. Com os dados oriundos das Tabela 17 e Tabela 16 foram definidos como imagens à serem filtradas aquelas com uma densidade abaixo de 2700. Os resultados são exibidos na Tabela 18.

Tabela 18 – Resultado da aplicação do Autoencoder

Conjunto	Precisão	Recall	mAP@0.5	mAP@0.5:0.95
Granada	0.705	0.650	0.702	0.533

Fonte: Produzido pelos autores.

Como é possível notar, comparando os resultados da Tabela 18 e Tabela 13 a utilização do AE não se mostrou efetiva o suficiente para trazer melhorias significativas na performance do sistema como um todo. Para o índice de precisão, houve uma queda de 0.789 para 0.705, o que pode ser interpretado como que o AE esteja barrando algumas detecções verdadeiras da rede base. As demais métricas, como o *recall*, observa-se baixas flutuações em seus valores, um relativo aumento após o filtro, onde este é advindo da passagem de detecções que a rede tenha considerado falsas.

A não melhora do modelo após a aplicação do AE pode ser explicada devido as margens de segurança bastante apertadas entre as medidas de erro de reconstrução e também da densidade do *kernel* do vetor latente. As detecções oriundas a rede podem ter origem de diversos trechos da imagem e não só dos objetos similares as armas de fogo, levando a uma grande variação das métricas utilizadas como *threshold*. Dessa maneira, apesar da efetiva análise do impacto negativo da inserção de falsos positivos nos conjuntos de treino e teste em modelos de detecção de objeto, a abordagem da utilização do AE para melhorar tais performances não se fez aplicável para o problema estudado.

5 Conclusão

A busca de aplicações de modelos de inteligência artificial em sistemas de segurança para auxílio na tomada de decisão vem ganhando bastante força com as recentes evoluções de algoritmos, coleta de dados e poder computacional. Empregar tais sistemas de forma eficiente, como destacado por Enríquez et al. (2019) pode reduzir os danos potenciais drasticamente. Como destacado no Capítulo 2, grande parte dos esforços vem sendo empregados para a redução de emissões de alarmes falsos desses sistemas e aumentando sua precisão de detecção.

Colocado em pauta a construção e uma maior eficiência de sistemas automáticos de detecção de armas de fogo, este trabalho apresenta um estudo sobre o impacto de objetos similares ao objeto que se deseja detectar em modelos de detecção de objetos. Ademais, é proposto um sistema que se utiliza de um AE para atuar como um filtro de emissão de alarmes falsos através do seu reconhecimento dos padrões desses objetos característicos.

Através da elaboração de dois conjuntos de dados, um contendo somente armas de fogo, e outro construído com cinco demais classes com potencial de serem tratadas pelo modelo como armas, foi realizado o estudo sobre o impacto causado desses objetos. Como observado no Capítulo 4, o modelo treinado somente sobre a visão de armas de fogo apresenta resultados bastante satisfatórios, com uma precisão de 0.932, no entanto, quando esse mesmo modelo é exposto a um conjunto de teste com objetos similares, sua precisão decai para 0.831, podendo emitir agora cerca de 17 falsos positivos a cada 100 detecções.

Para o segundo modelo treinado com introdução de objetos similares em seu conjunto de treino, é interessante a observação de como esses dados impactaram negativamente sua performance, saindo de uma precisão de 0.932 para 0.789, levando a uma taxa de falsos positivos de cerca de 21%. Além disso, a introdução desses, fez com que a rede se tornasse menos eficaz na detecção de armas de fogo, aumentando sua taxa de falsos negativos, antes com um *recall* de 0.745 para 0.618. Ainda com a variação do IOU para a medição do mAP, sua performance também apresenta uma queda considerável.

Por fim, a tratativa aqui estudada da utilização de um AE para a melhora de sistemas vulneráveis onde os índices de FP são cruciais para um melhor trabalho não apresentou melhorias satisfatórias. Obtendo poucas mudanças efetivas nas métricas observadas e em alguns casos em uma queda na precisão do modelo, não superando o modelo base YOLO V7 e nem os detectores utilizados como referência, a SSD Tabela 9 e a *RetinaNet* Tabela 10. As causas da baixa performance podem ser indícios dos *threshols* utilizadas com poucas margem de separação entre si, indicando a pouca diferenciação do AE entre

armas de fogo e objetos falsos positivos.

Referências

ASHBY, M. P. The value of cctv surveillance cameras as an investigative tool: An empirical analysis. *European Journal on Criminal Policy and Research*, Springer, v. 23, n. 3, p. 441–459, 2017. Citado na página 16.

BANK, D.; KOENIGSTEIN, N.; GIRYES, R. Autoencoders. *Machine learning for data science handbook: data mining and knowledge discovery handbook*, Springer, p. 353–374, 2023. Citado 2 vezes nas páginas 28 e 29.

BEGGEL, L.; PFEIFFER, M.; BISCHL, B. Robust anomaly detection in images using adversarial autoencoders. In: SPRINGER. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. [S.l.], 2019. p. 206–222. Citado 3 vezes nas páginas 28, 29 e 42.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. Citado 3 vezes nas páginas 26, 27 e 35.

BORGES, L. E. *Python para desenvolvedores*. 1. ed. [S.l.]: Novatech, 2014. Citado na página 34.

BOURLARD, H.; KAMP, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, Springer, v. 59, n. 4-5, p. 291–294, 1988. Citado na página 28.

CHEN, K. et al. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. Citado na página 34.

CUKIER, W.; EAGEN, S. A.; DECAT, G. Gun violence. *Aggression and violence*, Routledge, p. 179–193, 2016. Citado na página 17.

ENRÍQUEZ, F. et al. Vision and crowdsensing technology for an optimal response in physical-security. In: SPRINGER. *Computational Science–ICCS 2019: 19th International Conference, Faro, Portugal, June 12–14, 2019, Proceedings, Part V 19*. [S.l.], 2019. p. 15–26. Citado 2 vezes nas páginas 16 e 53.

GHOLAMALINEZHAD, H.; KHOSRAVI, H. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020. Citado 2 vezes nas páginas 24 e 25.

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 26.

GONZÁLEZ, J. L. S. et al. Real-time gun detection in cctv: An open problem. *Neural networks*, Elsevier, v. 132, p. 297–308, 2020. Citado na página 17.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016. Citado 7 vezes nas páginas 19, 20, 21, 22, 23, 24 e 28.

GOODFELLOW, I. et al. Generative adversarial networks. *Communications of the ACM*, ACM New York, NY, USA, v. 63, n. 11, p. 139–144, 2020. Citado na página 30.

GOODFELLOW, I. et al. Maxout networks. In: PMLR. *International conference on machine learning*. [S.l.], 2013. p. 1319–1327. Citado na página 22.

HE, K. et al. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 37, n. 9, p. 1904–1916, 2015. Citado na página 26.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. Citado na página 39.

HINTON, G. E. How neural networks learn from experience. *Scientific American*, JSTOR, v. 267, n. 3, p. 144–151, 1992. Citado 2 vezes nas páginas 19 e 20.

HINTON, G. E.; ZEMEL, R. Autoencoders, minimum description length and helmholtz free energy. *Advances in neural information processing systems*, v. 6, 1993. Citado na página 28.

HUBEL, D. H.; WIESEL, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, Wiley-Blackwell, v. 160, n. 1, p. 106, 1962. Citado na página 21.

JACCARD, P. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, v. 37, p. 547–579, 1901. Citado na página 27.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na página 41.

LECUN, Y.; BENGIO, Y. et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, Cambridge, MA USA, v. 3361, n. 10, p. 1995, 1995. Citado 2 vezes nas páginas 22 e 23.

LECUN, Y. et al. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, v. 2, 1989. Citado na página 21.

LIN, T.-Y. et al. Feature pyramid networks for object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 2117–2125. Citado na página 26.

LIN, T.-Y. et al. Focal loss for dense object detection. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2980–2988. Citado 4 vezes nas páginas 26, 34, 36 e 39.

LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. [S.l.], 2016. p. 21–37. Citado 3 vezes nas páginas 26, 34 e 38.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, p. 115–133, 1943. Citado na página 19.

NELSON, J.; SOLAWETZ, J. Yolov5 is here: State-of-the-art object detection at 140 fps. *Roboflow com https://blog roboflow com/yolov5-is-here Accessed*, v. 17, 2020. Citado na página 26.

OLMOS, R.; TABIK, S.; HERRERA, F. Automatic handgun detection alarm in videos using deep learning. *Neurocomputing*, Elsevier, v. 275, p. 66–72, 2018. Citado na página 16.

OLMOS, R. et al. A binocular image fusion approach for minimizing false positives in handgun detection with deep learning. *Information Fusion*, Elsevier, v. 49, p. 271–280, 2019. Citado na página 17.

OLMOS, R. et al. Multicast: Multi confirmation-level alarm system based on cnn and lstm to mitigate false alarms for handgun detection in video-surveillance. *arXiv preprint arXiv:2104.11653*, 2021. Citado 2 vezes nas páginas 16 e 17.

PADILLA, R.; NETTO, S. L.; SILVA, E. A. D. A survey on performance metrics for object-detection algorithms. In: IEEE. *2020 international conference on systems, signals and image processing (IWSSIP)*. [S.l.], 2020. p. 237–242. Citado 2 vezes nas páginas 27 e 28.

PADILLA, R. et al. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, v. 10, n. 3, 2021. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/10/3/279>. Citado na página 35.

PÉREZ-HERNÁNDEZ, F. et al. Object detection binary classifiers methodology based on deep learning to identify small objects handled similarly: Application in video surveillance. *Knowledge-Based Systems*, Elsevier, v. 194, p. 105590, 2020. Citado 2 vezes nas páginas 32 e 33.

RASCHKA, S. *Python machine learning*. [S.l.]: Packt publishing ltd, 2015. Citado 3 vezes nas páginas 22, 23 e 24.

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 na página 27.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. Citado na página 27.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, v. 28, 2015. Citado na página 26.

RUIZ-SANTAQUITERIA, J. et al. Handgun detection using combined human pose and weapon appearance. *IEEE Access*, IEEE, v. 9, p. 123815–123826, 2021. Citado 3 vezes nas páginas 16, 30 e 31.

SHARMA, S.; SHARMA, S.; ATHAIYA, A. Activation functions in neural networks. *Towards Data Sci*, v. 6, n. 12, p. 310–316, 2017. Citado na página 21.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado na página 38.

UIJLINGS, J. R. et al. Selective search for object recognition. *International journal of computer vision*, Springer, v. 104, p. 154–171, 2013. Citado na página 26.

VALLEZ, N.; VELASCO-MATA, A.; DENIZ, O. Deep autoencoder for false positive reduction in handgun detection. *Neural Computing and Applications*, Springer, v. 33, n. 11, p. 5885–5895, 2021. Citado 2 vezes nas páginas 16 e 29.

WANG, C.-Y.; BOCHKOVSKIY, A.; LIAO, H.-Y. M. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2023. p. 7464–7475. Citado 4 vezes nas páginas 16, 27, 35 e 36.

XIAO, Y. et al. A review of object detection based on deep learning. *Multimedia Tools and Applications*, Springer, v. 79, p. 23729–23791, 2020. Citado na página 25.

ZOU, Z. et al. Object detection in 20 years: A survey. *Proceedings of the IEEE*, IEEE, 2023. Citado 4 vezes nas páginas 25, 26, 27 e 28.