Deteção de armas em imagens e vídeo

Jorge Teixeira

António Cunha Alexandre Neto

Departamento Engenharia Informática

Universidade de Trás-os-Montes e Alto

Douro

***Abstract*—** Este artigo vem a explorar a construção de um modelo de deteção de armas de fogo em imagem e vídeo.

***Keywords—deteção de objetos, deep learning, deteçao de arma, segurança***

1. Diagram

   Description automatically generated**INTRODUÇÃO**

Hoje em dia é possível construir e modificar modelos de identificação, tanto em tempo real como por imagem, com grande eficácia e precisão. Então a sua implementação em sistemas de monotorização ou de segurança era obvia. A capacidade de utilizarmos *deep learnin*g para treinar um modelo que consegue identificar o objeto ou objetos pretendidos e colocar uma *bounding box* no mesmo e corretamente identificá-lo. Com este estudo pretendemos aplicar estas técnicas na identificação de armas.

1. **OBJETIVO**

Graphical user interface

Description automatically generatedNeste trabalho é pretendido e exploração e a aprendizagem dos métodos e ferramentas utilizadas a construção e treino de um modelo que seja capas de detetar armas. Com isto foi priorizado o uso de técnicas de treino mais simples ou com uma maior afinidade por um equipamento com pouco poder computacional.

1. **ESTADO DE ARTE**

A Classificação de Imagens é uma tarefa fundamental que tenta compreender uma imagem. O objetivo é classificar a imagem atribuindo-lhe um rótulo específico. Normalmente, refere-se a imagens nas quais apenas um objeto aparece e é analisado.

Em contraste, deteção de objetos envolve tarefas de classificação e localização onde podem existir vários objetos dentro da imagem. Os métodos de última geração podem ser categorizados em dois tipos principais: métodos de uma etapa e métodos de dois etapas. Os métodos de uma etapa, dão maior importância a velocidade, como por exemplo os modelos de YOLO, SSD e RetinaNet. Os métodos de duas etapas priorizam a precisão da deteção e temos como exemplo os modelos de Faster R-CNN, Mask R-CNN e Cascade R-CNN.

Neste trabalho, foquei me em utilizar um método de uma etapa por causa da sua rapidez e para podermos ter uma detenção em tempo real. YOLO sendo um dos mais rápidos a detetar objetos. De seguida estudei as várias versões do método YOLO sendo estas YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5. Sendo YOLOv4 e YOLOv5, as mais recentes e com otimizações e alterações no *Backbone*, *Neck* e *Head* relativamente ao YOLOv3.

Mas por haver um maior conteúdo educacional foi utilizado foi decidido optar-se pela YOLOv3 e a partir dela fazer todos os testes necessários e modificações. Utilizando um dos modelos mais pequenos YOLOv3-Tiny.

Olhando para os resultados de pjreddie.com (imagem acima), a arquitetura YOLOv3-Tiny é aproximadamente seis vezes mais rápida do que os seus irmãos mais velhos, alcançando mais de 220 FPS numa única GPU.

1. **METODOLOGIA**

Neste trabalho foi usado uma *pipeline*,para demonstrar os passos necessários a realização do mesmo.

A picture containing text, athletic game, sport, racquetball

Description automatically generatedO primeiro passo é criar um *Dataset* compostos com imagens suficientes ao treino do modelo. De seguida é o tratamento desse *Dataset* para que esteja estruturado para ser lido pelo modelo. Terceiro passo é a criação do modelo. Apos de procedermos ao treino temos de o testar passando uma imagem e um vídeo.

# Criação Dataset

O *Dataset* é parte mais importante na criação de um bom modelo. Mas como o tema é deste trabalho toca no ponto de segurança nacional, fui incapaz de obter *Datasetes* de organizações privadas. Com isto foi recolhido imagens publicas de armas. Para facilitar o treino em vez de usar as 80 classes do COCO *Dataset* e depois adicionar o meu *Dataset* de armas. Procedi a só usar o meu *Dataset* com uma única class “*gun*” com 3000 imagens de pistolas, foi encontrado imagens de outras armas, mas não com tantas como pistolas e com boa qualidade (sem marcas de água que possam interferir com o treino do modelo). Para a anotação de algumas imagens e também para conhecer a ferramenta foi utilizado a ferramenta ImageLabel para manualmente criar as *bounding boxes* à volta dos objetos e retirar o documento de texto necessário para ser usado no treino do modelo*.*

# Pré-processamento

Diagram

Description automatically generatedAs imagens foram redimensionadas para 416x416 porque é o parâmetro de entrada para YOLO. Algem disso foram usadas Darknet e PyTorch *frameworks* que ajudam a otimizar os dados para serem lidos de uma forma mais rápida pela GPU.

Também foi feita uma organização das imagens e a criação dos ficheiros .data e .cfg para serem usados no treino do modelo.

1. **Modelo**

Primeiro, o algoritmo divide a imagem em um *grid*. Cada uma dessas células é responsável por fazer a previsão de 5 caixas delimitadoras, para caso haja mais de um objeto naquela célula. Também é retornado à pontuação de confiança que diz quanta certeza tem na existência de um objeto naquela caixa delimitadora. Para cada caixa, a célula também faz a previsão de uma *class*. Isso funciona como se fosse um classificador. Fornecendo um valor de probabilidade para cada uma das classes possíveis. O valor de confiança para a caixa delimitadora e da previsão da classe são combinados em uma pontuação final, diz a probabilidade dessa caixa conter um objeto específico. Acontece que a maioria dessas caixas terá um valor de confiança extremamente baixo, então por isso geralmente se considera apenas as caixas cuja pontuação final seja 30% ou mais. Esse valor de 30% é o limiar, chamado de *threshold*, e ele pode ser alterado dependendo do quão preciso queremos que ele seja.

# V. RESULTADOS

Os resultados do modelo foram 0.675438 ac, 0.742858 avg loss.

Os resultados de video tambem foram bastante favoraveis, nos videos com muitos objetos aglomerados o modelo não conseguia detectalos a todos, detetando aqueles mais proximos mais facimente.

# VI. CONCLUSÃO

Este trabalho demonstrou-me as qualidades e as dificuldades na área de deep leraning. Com a realização, deste desafio compreendi que o tratamento de dados é 70% do trabalho na construção de um bom modelo e os 30% restantes é a força bruta que temos de colocar no código para podermos ter bons resultados e maus resultados para podemos traçar um “porque” de o modelo aprender melhor com umas determinadas configurações. Este trabalho deu a oportunidade de explorar e aprender deteção em foto, vídeo e *webcam* e muitas outras ferramentas para manipular os dados e podê-los processar.

**REFERÊNCIAS**

1. Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767* (2018).
2. You Only Look Once: Unified, Real-Time Object Detection Joseph Redmon∗ , Santosh Divvala∗†, Ross Girshick¶ , Ali Farhadi∗† University of Washington∗ , Allen Institute for AI† , Facebook AI Research¶
3. Performance Benchmarking of YOLO Architectures for Vehicle License Plate Detection from Real-time Videos Captured by a Mobile Robot Amir Ismail1,2 , Maroua Mehri1 , Anis Sahbani2 and Najoua Essoukri Ben Amara1 1Universite de Sousse, Ecole Nationale d’Ing ´ enieurs de Sousse, ´ LATIS-Laboratory of Advanced Technology and Intelligent Systems, 4023, Sousse, Tunisia 2Enova Robotics, Novation City, Technopole de Sousse, 4000, Sousse, Tunisia
4. Developing a Real-Time Gun Detection Classifier Justin Lai Stanford University jzlai@stanford.edu Sydney Maples Stanford University [smaples@stanford.edu](mailto:smaples@stanford.edu)
5. Código: https://github.com/JorgeTeixeiraPT/Deeplearning.git