Pós Graduação em IA para desenvolvedores

FIAP

Leandro Cordeiro David - RM356103

Documentação da Solução de Detecção de Armas Brancas e de Fogo

Repositório github: https://github.com/leandrodvd/fiap_hackathon_ia4devs

Vídeo de demonstração no youtube: https://youtu.be/ygtwh-6BPSQ?si=tC1yxIWcGa5b5Vt3

1. Introdução

Esta documentação descreve o desenvolvimento de uma solução para análise de vídeo com detecção de objetos cortantes, como facas e tesouras, além de armas de fogo. O objetivo é identificar automaticamente esses objetos em vídeos e enviar alertas por e-mail quando uma detecção ocorrer.

2. Dataset Utilizado

Para o treinamento do modelo, foi utilizado o dataset "Weapon Detection Dataset for YOLO", disponível no Kaggle em
https://www.kaggle.com/datasets/raghavnanjappan/weapon-dataset-for-yolov5/

Este conjunto de dados contém 4.000 imagens, sendo 2.000 de armas de fogo e 2.000 de facas e objetos similares. Cada imagem possui anotações no formato YOLO, indicando a posição e a classe dos objetos presentes.

3. Ferramentas e Bibliotecas

- **YOLOv8**: Biblioteca utilizada para detecção de objetos, escolhida por sua eficiência e modelos pré-treinados que podem ser customizados com novos datasets.
- **OpenCV**: Utilizada para manipulação e processamento de vídeos.
- **Mailgun**: Serviço de e-mail utilizado para envio de alertas quando um objeto é detectado.

4. Estrutura do Projeto

A estrutura do projeto está organizada da seguinte forma:

bash
CopiarEditar

```
project_root/
├── datasets/
│   ├── images/
```

```
|   |   ├── train/       # Imagens de treinamento
|   |   └── test/        # Imagens de teste
|   ├── labels/
|   |   ├── train/       # Anotações das imagens de treinamento
|   |   └── test/        # Anotações das imagens de teste
|   └── dataset.yaml     # Arquivo de configuração do dataset
├── runs/
|   └── detect/
|       └── train/       # Resultados do treinamento
├── train.py             # Script para treinamento do modelo
└── analyze.py           # Script para análise de vídeos e envio de
alertas
```

5. Treinamento do Modelo

O treinamento foi realizado utilizando o script `train.py`. Neste script, o modelo base YOLOv8n (nano) foi carregado e treinado com o dataset mencionado. Os parâmetros principais utilizados foram:

- **Épocas**: 50
- **Tamanho da imagem (imgsz)**: 640
- **Tamanho do batch (batch)**: 16
- **Dispositivo (device)**: CPU

Devido à ausência de uma GPU, o treinamento foi realizado em uma CPU Intel Core i7 em um MacBook, com duração aproximada de 5 horas. Após o treinamento, o modelo gerado foi salvo no arquivo `best.pt`.

6. Análise de Vídeos e Envio de Alertas

Para a análise de vídeos, foi desenvolvido o script `analyze.py`, que recebe como parâmetros o caminho do vídeo a ser analisado, as credenciais do Mailgun e o endereço de e-mail para envio de alertas. O fluxo de execução é o seguinte:

1. **Carregamento do Modelo**: O modelo treinado (`best.pt`) é carregado.
2. **Processamento do Vídeo**: O vídeo é lido frame a frame utilizando o OpenCV.
3. **Detecção de Objetos**: Para cada frame, o modelo realiza a detecção de objetos.
4. **Desenho de Caixas Delimitadoras**: Para cada objeto detectado com confiança acima de 0.2, é desenhada uma caixa delimitadora no frame.
5. **Verificação de Detecção**: Se algum objeto for detectado, uma flag é ativada.
6. **Envio de Alerta**: Caso a flag esteja ativada, é enviado um e-mail de alerta através do Mailgun.

7. **Geração de Vídeo de Saída**: Os frames processados são salvos em um novo vídeo (`output.mp4`) para visualização das detecções.

7. Resultados

O modelo apresentou desempenho satisfatório, detectando a maioria das armas de fogo e facas nos vídeos de teste. No entanto, algumas limitações foram observadas:

- **Ângulos e Posições**: Detecções falharam em certos ângulos ou posições dos objetos.
- **Objetos Similares**: Alguns objetos que se assemelham a armas ou facas foram detectados incorretamente.

Para melhorar a precisão, seria necessário ampliar e diversificar o dataset de treinamento, incluindo mais variações de ângulos, iluminações e tipos de objetos.

8. Considerações Finais

Esta solução demonstra a viabilidade de utilizar modelos YOLO para detecção de objetos específicos em vídeos e a integração com serviços de notificação para alertas em tempo real. Embora o modelo atual funcione como um MVP (Produto Mínimo Viável), melhorias podem ser implementadas para aumentar sua precisão e robustez.

9. Repositório do Código

Todo o código desenvolvido está disponível no GitHub:
https://github.com/leandrodvd/fiap_hackathon_ja4devs

Este repositório contém instruções detalhadas para reprodução dos resultados e testes adicionais.