



HACKATON FIAP - IA para Devs

Este projeto realiza **detecção de objetos cortantes em vídeos** utilizando **YOLOv5**, com possibilidade de treinar um modelo personalizado e testar em arquivos **.mp4** e emitir alertas automáticos quando objetos suspeitos forem confirmados por múltiplos frames.

Grupo 2

- Julio Cesario de Paiva Leão (julio0023@live.com)
- Luis Gustavo Bueno Colombo (luisgustavobuenocolombo@gmail.com)



Proposta

- ✓ Detectar objetos cortantes (como facas e tesouras) em vídeos utilizando visão computacional.
 - ✓ Implementar um sistema inteligente de confirmação baseado em múltiplos frames.
 - ✓ Enviar alertas automáticos quando um objeto for confirmado em cena.
 - ✓ Permitir treino de um modelo YOLOv5 com dataset personalizado.
 - ✓ Executar testes com vídeos locais, exibindo ou salvando os resultados.
-



Requisitos

- Python 3.8+
- CUDA (opcional, para uso com GPU)
- Dependências listadas em **requirements.txt**



Instalação

```
# Clone o repositório
git clone https://github.com/julioleao/hackaton.git
cd hackaton

# Crie e ative um ambiente virtual (opcional)
python -m venv venv
source venv/bin/activate # ou venv\Scripts\activate no Windows

# Instale as dependências
pip install -r requirements.txt
```



Configuração

As configurações do projeto estão centralizadas no arquivo **configs.py**, incluindo:

- Caminhos dos vídeos e datasets
- Nome do modelo
- Uso de GPU ou CPU
- Tamanho da imagem
- Se o vídeo será exibido em tempo real (**STREAMING**)

Você pode ajustar os parâmetros para testar com diferentes datasets ou vídeos.

Treinamento

O script **trainer.py** realiza o treinamento usando YOLOv5 e cria automaticamente o **data.yaml** com as classes e caminhos para o dataset.

Rodar o treinamento

```
python trainer.py
```

- O modelo será treinado por 4 épocas (modificável).
- O checkpoint será salvo em **runs/train/exp/weights/best.pt**.

Teste

O script **tester.py** roda o vídeo definido em **configs.py** com o modelo treinado e exibe (ou salva) a detecção.

Rodar o teste

```
python tester.py
```

- O vídeo de saída será salvo como **output.mp4** por padrão.
- Para visualização em tempo real, **STREAMING = True** em **configs.py**.

Funcionalidades

- A cada frame, a detecção é realizada usando YOLOv5.
- Se um objeto é identificado por pelo menos 4 frames consecutivos na mesma região, é confirmado.
- Ao confirmar um novo objeto cortante, uma notificação automática é enviada:
 - Windows: via PowerShell (MessageBox)
 - Linux: via notify-send

Parâmetros importantes

- **DIST_THRESHOLD**: distância máxima entre detecções para considerá-las do mesmo objeto.
- **CONFIRM_FRAMES**: número mínimo de frames consecutivos para confirmar o objeto.

Explicação dos Arquivos

Arquivo	Função
<code>configs.py</code>	Define os caminhos, parâmetros de imagem, uso de GPU/CPU, e flags de execução.
<code>trainer.py</code>	Gera o <code>data.yaml</code> e executa o treinamento com o modelo YOLOv5.
<code>tester.py</code>	Usa o modelo treinado para detectar facas em um vídeo de entrada.
<code>requirements.txt</code>	Lista as bibliotecas necessárias para rodar o projeto.
<code>videos/</code>	Pasta onde os vídeos de entrada devem ser colocados.
<code>dataset/</code>	Contém o dataset usado para o treinamento (ex: imagens, anotações).

Observações

- A estrutura do dataset deve seguir o padrão YOLOv5 (pastas `train/images`, `valid/images`, etc.).
- O código utiliza diretamente os módulos do repositório oficial do YOLOv5.
- Alertas são enviados apenas quando um objeto é **confirmado** (não basta aparecer em apenas 1 frame).
- A detecção usa limiares de confiança e IoU configuráveis no `tester.py`.