

Sistema de Alarme Pet Friendly com YOLO e Python



Introdução

A segurança doméstica evoluiu com o avanço da inteligência artificial. Hoje, é possível construir sistemas de monitoramento altamente eficientes utilizando visão computacional e aprendizado de máquina. Neste e-book, vamos desenvolver um **sistema de alarme inteligente** com Python que **detecta a presença de seres humanos em tempo real utilizando a câmera**, disparando um som de alerta automaticamente — mas com um diferencial importante: **ele é pet friendly**.

Isso significa que o sistema é capaz de **ignorar cães e outros animais de estimação**, evitando falsos alarmes comuns em soluções comerciais mais simples. Utilizaremos o algoritmo **YOLO (You Only Look Once)**, um dos modelos de detecção de objetos mais rápidos e precisos disponíveis, ideal para aplicações em tempo real.

O que você vai aprender:

- Como configurar seu ambiente Python para visão computacional;
- Como utilizar a câmera do computador para capturar vídeo em tempo real;
- Como usar o modelo YOLO para identificar humanos e animais em uma imagem;
- Como disparar um som ao detectar uma pessoa;
- Como garantir que o sistema **não reaja a cães** soltos pela casa.

Aplicações práticas:

- Sistemas de segurança caseira;
- Monitoramento de ambientes internos ou externos;
- Alarmes automáticos para residências, escritórios ou lojas;
- Projetos DIY de automação residencial com IA.

Este e-book é ideal para desenvolvedores, estudantes e entusiastas de tecnologia que desejam criar soluções práticas de IA para o dia a dia — de forma acessível, eficiente e ética, respeitando a presença dos nossos companheiros de quatro patas .

Vamos começar?

Capítulo 1: Preparando o Ambiente

Antes de começarmos a programar nosso sistema de alarme pet friendly com YOLO e Python, precisamos garantir que o ambiente esteja pronto para lidar com visão computacional, detecção de objetos e reprodução de som.

Requisitos do Sistema

Para este projeto, você vai precisar de:

- Um computador com **Python 3.8+**
 - Uma **webcam** (integrada ou USB)
 - Acesso à internet para baixar os modelos pré-treinados
 - Sistema operacional: Windows, macOS ou Linux
-

1.1 Instalando o Python

Se ainda não tiver o Python instalado:

- [Baixe o Python aqui](#)
- Certifique-se de **marcar a opção “Add Python to PATH”** durante a instalação
- Verifique se está tudo certo com o comando:

```
python --version
```

1.2 Criando o Ambiente Virtual (opcional, mas recomendado)

```
python -m venv yolovenv  
source yolovenv/bin/activate # Linux/macOS  
yolovenv\Scripts\activate # Windows
```

1.3 Instalando as Bibliotecas Necessárias

Vamos instalar as bibliotecas principais:

```
pip install ultralytics opencv-python playsound
```

Se estiver usando Linux/macOS e o playsound não funcionar bem, considere usar o

```
pygame:
```

```
pip install pygame
```

Explicação rápida:

- ultralytics: para carregar e usar o YOLOv5/v8 com facilidade
 - opencv-python: para capturar vídeo da câmera
 - playsound ou pygame: para emitir um som quando uma pessoa for detectada
-

💡 1.4 Testando a Webcam com OpenCV

Vamos garantir que a câmera esteja funcionando. Crie um arquivo test_camera.py:

```
import cv2

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    cv2.imshow("Camera", frame)
    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Execute com:

```
python test_camera.py
```

✓ 1.5 Testando a Reprodução de Som

Crie um arquivo .mp3 ou .wav chamado alarme.mp3 e salve na mesma pasta do projeto.

Teste com:

```
from playsound import playsound  
playsound('alarme.mp3')
```

Se estiver usando pygame, substitua por:

```
import pygame  
pygame.mixer.init()  
pygame.mixer.music.load("alarme.mp3")  
pygame.mixer.music.play()
```

Estrutura Inicial do Projeto

projeto-alarme-petfriendly/

```
|—— alarme.mp3  
|—— detector.py  
|—— test_camera.py  
|—— README.md  
└—— requirements.txt
```

Com o ambiente pronto, câmera testada e bibliotecas instaladas, estamos prontos para mergulhar no coração do projeto: **o YOLO!** No próximo capítulo, vamos entender como o YOLO funciona e como usar um modelo pré-treinado para detectar pessoas e pets em tempo real.



Capítulo 2: Introdução ao YOLO

O que é YOLO?

YOLO (You Only Look Once) é uma técnica de detecção de objetos em imagens e vídeos que revolucionou a visão computacional pela sua velocidade e eficiência. Ao contrário de outros algoritmos que fazem múltiplas passagens sobre uma imagem, o YOLO realiza **toda a detecção em uma única execução (forward pass)** pela rede neural — por isso o nome “Você Só Olha Uma Vez”.

Com ele, conseguimos detectar em tempo real objetos como:

- Pessoas (person)
- Cães (dog)
- Gatos (cat)
- Veículos (car, truck, bicycle, etc.)
- Muitos outros — dependendo do modelo e do dataset usado para treinamento

Isso torna o YOLO perfeito para aplicações como:

- Sistemas de segurança (como o deste e-book)
- Monitoramento de tráfego
- Robótica
- Contagem de pessoas

Como o YOLO funciona?

O YOLO divide a imagem de entrada em uma **grade (grid)** e, para cada célula dessa grade, ele prevê:

- A presença de um objeto
- A classe do objeto (ex: pessoa, cachorro, etc.)
- As coordenadas da caixa delimitadora (bounding box)

Ao final, o modelo retorna uma lista de objetos detectados, com:

- Nome da classe (ex: person)
- Confiança (probabilidade de acerto, entre 0 e 1)

- Coordenadas da caixa (x, y, largura, altura)
-

Qual versão do YOLO usar?

Atualmente, existem várias versões do YOLO, desde a **v1** até a **v12**, incluindo forks como **YOLOv5** (da Ultralytics)

YOLOv5 ou YOLOv8 da Ultralytics

Ambas são fáceis de usar, oferecem modelos pré-treinados e têm uma API direta para detecção com poucos comandos. Além disso, trabalham com o famoso dataset **COCO**, que já identifica 80 classes, incluindo person, dog e cat — exatamente o que precisamos para nosso sistema pet friendly.

Modelos Pré-treinados

Você pode usar modelos já prontos, como:

- `yolov5s.pt` (small) — mais leve e rápido
- `yolov5m.pt`, `yolov5l.pt` e `yolov5x.pt` — maiores, mais precisos, mas mais lentos

Para a maioria dos casos (inclusive este projeto), o `yolov5s` é suficiente.

Por que YOLO para este projeto?

- **Velocidade**: funciona em tempo real, mesmo em computadores modestos
- **Precisão suficiente** para distinguir humanos de pets
- **Facilidade de uso com Python**
- **Suporte a múltiplas classes ao mesmo tempo**

Com o YOLO pronto para uso, estamos prontos para configurar o ambiente e começar a codificar. Vamos colocar tudo para funcionar no próximo capítulo!

Capítulo 3: Capturando Vídeo e Detectando com YOLO

Agora que temos o ambiente configurado e o YOLO instalado, é hora de integrar tudo para capturar vídeo da câmera e detectar objetos em tempo real.

Neste capítulo, vamos:

- Capturar o vídeo da webcam com OpenCV
 - Carregar um modelo pré-treinado do YOLO
 - Detectar humanos, cães e outros objetos
 - Exibir o resultado na tela com caixas delimitadoras
-

3.1 Carregando o Modelo YOLO

Vamos usar o modelo leve yolov8n.pt, ideal para testes em tempo real:

```
from ultralytics import YOLO

# Carregando o modelo pré-treinado
model = YOLO("yolov8n.pt") # ou "yolov5s.pt" se preferir
```

Observação: na primeira execução, o modelo será baixado automaticamente.

3.2 Capturando Vídeo da Câmera

A seguir, abrimos a webcam e aplicamos o YOLO em tempo real:

```
import cv2
from ultralytics import YOLO

# Carrega o modelo
model = YOLO("yolov8n.pt")

# Abre a câmera
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Detecção com YOLO
    results = model(frame)

    # Obtém as caixas, classes e confidências
    annotated_frame = results[0].plot() # desenha as caixas automaticamente

    # Exibe o frame com as detecções
    cv2.imshow("Detecção YOLO", annotated_frame)

    if cv2.waitKey(1) == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
```

🧠 3.3 Como Interpretar os Resultados

O YOLO detecta objetos e retorna:

- **Classe** (ex: person, dog, cat)
- **Confiabilidade** da detecção (ex: 0.85)
- **Coordenadas** da bounding box

Você pode acessar manualmente os objetos detectados assim:

```
for result in results:  
    for box in result.boxes:  
        cls = int(box.cls[0])  
        conf = float(box.conf[0])  
        class_name = model.names[cls]  
        print(f"Classe: {class_name}, Confiança: {conf:.2f}")
```

Resultado Esperado

Ao executar esse código, você verá a câmera aberta com **caixas e nomes sobrepostos nos objetos detectados**. Pessoas serão marcadas como person, cães como dog e assim por diante. Tudo isso em tempo real!

Próximo Passo

No próximo capítulo, vamos **filtrar apenas pessoas** e configurar o sistema para **ignorar cachorros** — ou seja, torná-lo verdadeiramente **pet friendly**. Também adicionaremos a **função de alarme** com som sempre que uma pessoa for detectada.



Capítulo 4: Sistema de Alarme Pet Friendly

Chegamos à parte mais importante do projeto: **fazer o sistema disparar um alarme sonoro ao detectar um ser humano, mas ignorar cães e gatos**. Isso garante que o sistema seja confiável e **não reaja aos pets da casa** 🐶🐱.

4.1 Lógica do Alarme

A lógica é simples:

1. Capturar o vídeo da câmera
 2. Detectar objetos com YOLO
 3. Verificar se há a classe person
 4. Se houver, **tocar o alarme** (som)
 5. Se detectar apenas dog, não fazer nada. Por lógica não fará nada se detectar outro pet 🐱🐶
-

4.2 Código Completo do Sistema

```
import cv2
from ultralytics import YOLO
from playsound import playsound # ou use pygame se preferir
import time

# Carrega o modelo YOLO
model = YOLO("yolov8n.pt") # ou yolov5s.pt

# Abre a câmera
cap = cv2.VideoCapture(0)
alarm_interval = 10 # segundos

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Faz a detecção
    results = model(frame)
    boxes = results[0].boxes
    classes_detectadas = [model.names[int(box.cls[0])]] for box in boxes]

    # Exibe o vídeo com anotações
    annotated_frame = results[0].plot()
    cv2.imshow("Pet Friendly Alarm", annotated_frame)

    # Se houver uma pessoa e nenhum cachorro, dispare o alarme
    if "person" in classes_detectadas:
        # Ignorar se só tiver cachorro
        if "dog" in classes_detectadas and len(classes_detectadas) == 1:
            pass
        else:
            current_time = time.time()
            if current_time - last_alarm_time > alarm_interval:
                print("⚠ Pessoa detectada! Disparando alarme...")
                playsound("alarme.mp3")
                last_alarm_time = current_time

    if cv2.waitKey(1) == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
```

4.3 Considerações Pet Friendly

O sistema está preparado para ignorar:

- Cães e gatos que passam em frente à câmera
- Ambientes com pets soltos dentro de casa
- Detecções múltiplas em que não há humanos

Você pode também adaptar para ignorar outras classes (chair, etc.) ou detectar apenas humanos com roupas específicas, caso deseje.

Conclusão

Neste e-book, construímos um sistema completo de **alarme inteligente com Python e YOLO**, capaz de detectar a presença de seres humanos em tempo real e **ignorar a presença de animais de estimação**.

Vimos:

- Como configurar o ambiente
- Como utilizar YOLO para detecção de objetos
- Como capturar vídeo da câmera e interpretar os resultados
- Como adicionar um alarme sonoro automático
- Como tornar o sistema **pet friendly**

Esse projeto pode ser expandido com:

- Notificações por e-mail ou WhatsApp
 - Integração com automação residencial (ex: acionar luzes ou trancar portas)
 - Detecção por zonas (áreas específicas da câmera)
-

Anexos

A.1 – Requisitos do requirements.txt

```
ultralytics==8.0.20
```

```
opencv-python
```

```
playsound
```

```
pygame # opcional, se playsound não funcionar no seu sistema
```

📁 A.2 – Estrutura Final do Projeto

projeto-alarme-petfriendly/

```
|—— detector.py  
|—— alarme.mp3  
|—— test_camera.py  
|—— requirements.txt  
└—— README.md
```

🔗 A.3 – Links Úteis

- [Ultralytics YOLO](#)
- [Dataset COCO classes](#)
- [OpenCV Documentation](#)
- [YOLOv5 Colab \(demo online\)](#)

