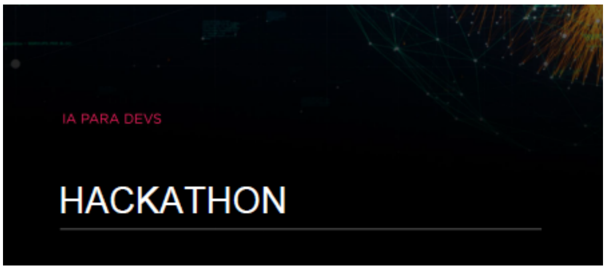


A FIAP VisionGuard, empresa de monitoramento de câmeras de segurança, está analisando a viabilidade de uma nova funcionalidade para otimizar o seu software.

O objetivo da empresa é usar de novas tecnologias para identificar situações atípicas e que possam colocar em risco a segurança de estabelecimentos e comércios que utilizam suas câmeras.

Um dos principais desafios da empresa é utilizar Inteligência Artificial para identificar objetos cortantes (facas, tesouras e similares) e emitir alertas para a central de segurança.

A empresa tem o objetivo de validar a viabilidade dessa feature, e para isso, será necessário fazer um MVP para detecção supervisionada desses objetos.

🔭 VISÃO GERAL DO PROJETO - Detecção de Objetos Cortantes com IA

🎯 Objetivo

- Criar um MVP com detecção supervisionada de objetos cortantes (facas, tesouras etc.) em vídeos de segurança, com:
- Dataset anotado
- Modelo treinado
- Detecção em vídeo
- Sistema de alerta simples (e-mail/log/print)
- Documentação clara e código reprodutível via Google Colab e GitHub

🧭 ROADMAP GERAL DO MVP

Etapa Nome Ferramentas Resultado Esperado

- 1 Setup do ambiente Colab + GitHub Ambiente com libs e conexão com o Drive
- 2 Preparação do Dataset Roboflow + imagens Dataset anotado com classes faca, tesoura, etc.
- 3 Treinamento do modelo YOLO (Ultralytics) Modelo treinado .pt
- 4 Teste em vídeo OpenCV + modelo Bounding boxes nos vídeos com precisão razoável
- 5 Sistema de alerta e-mail/logs Alerta ao detectar objeto cortante
- 6 Deploy de script final GitHub + Colab Código documentado + link funcional
- 7 Documentação + vídeo Markdown + vídeo de 15min Explicação clara do processo

```
#🔧 Etapa 1 - Setup do Ambiente

# 📁 Instalar dependências e conectar com Google Drive
!pip install -q ultralytics opencv-python-headless roboflow

from google.colab import drive
drive.mount('/content/drive')
```

Mostrar saída oculta

🛠️ Guia Visual Passo a Passo – Roboflow para Detecção de Facas e Tesouras

🔗 1. Acesse: https://roboflow.com

Crie uma conta gratuita ou entre com Google/GitHub.

🧪 2. Crie um novo projeto

Clique em “Create New Project”

Preencha os campos:

- Campo Valor sugerido
- Project Name Cortantes
- Project Type Object Detection
- Annotation Group bounding box
- License Public (ou Private se preferir)

🖼️ 3. Adicione as imagens

✅ Para imagens positivas: Download os datasets sugeridos:

- Knives Dataset

- Scissors Detection

Clique em "Upload Images" no seu projeto Roboflow

Selecione os arquivos .jpg/.png e as labels .txt (caso exportadas de outro projeto)

❌ Para imagens negativas:

Faça upload de imagens sem objetos cortantes (ex: cozinha vazia, sala, rua)

Não anote nada nessas imagens (o próprio Roboflow entende como "background")

🏷️ 4. Verifique as anotações

Vá em “Annotations” e revise cada imagem.

Verifique se as classes estão corretas:

knife, scissors, cutter, etc.

Renomeie classes, se necessário, em “Classes” > “Edit”

⚙️ 5. Gerar versão do dataset Clique em “Generate Dataset”

Escolha:

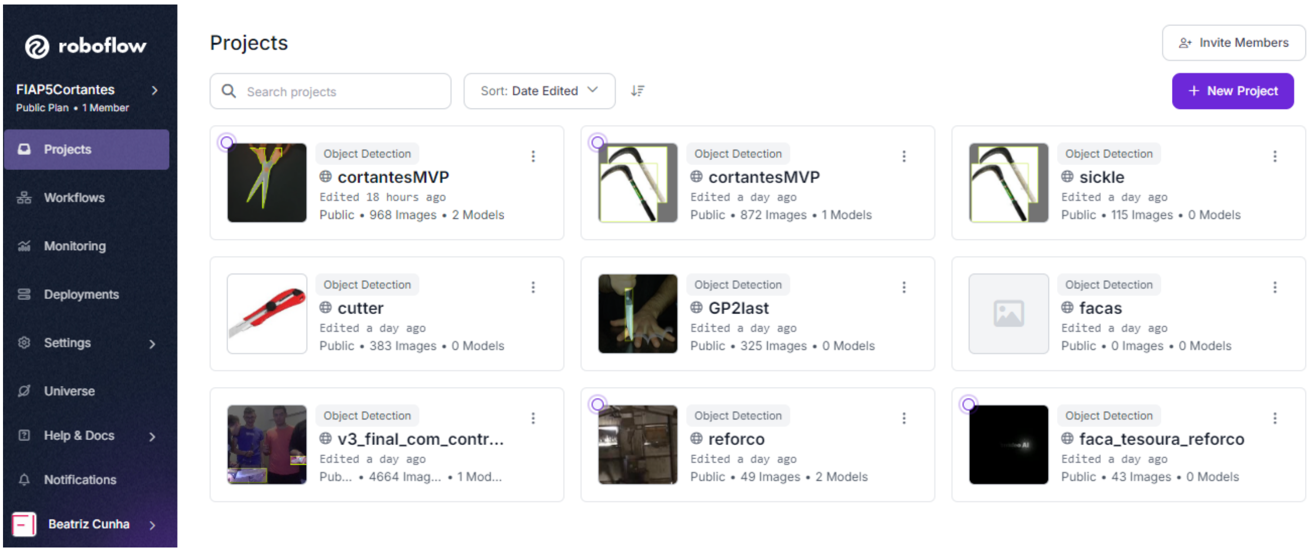
Resize: 640x640 (YOLO padrão)

Augmentations: Horizontal Flip, Blur, Exposure, etc. (opcional)

📦 6. Exportar o dataset (formato YOLO8)

Após gerar, vá em “Download Dataset”

Escolha o formato: YOLOv8



Projects

Object Detection
cortantesMVP
Edited 18 hours ago
Public • 968 Images • 2 Models

Object Detection
cortantesMVP
Edited a day ago
Public • 872 Images • 1 Models

Object Detection
sickle
Edited a day ago
Public • 115 Images • 0 Models

Object Detection
cutter
Edited a day ago
Public • 383 Images • 0 Models

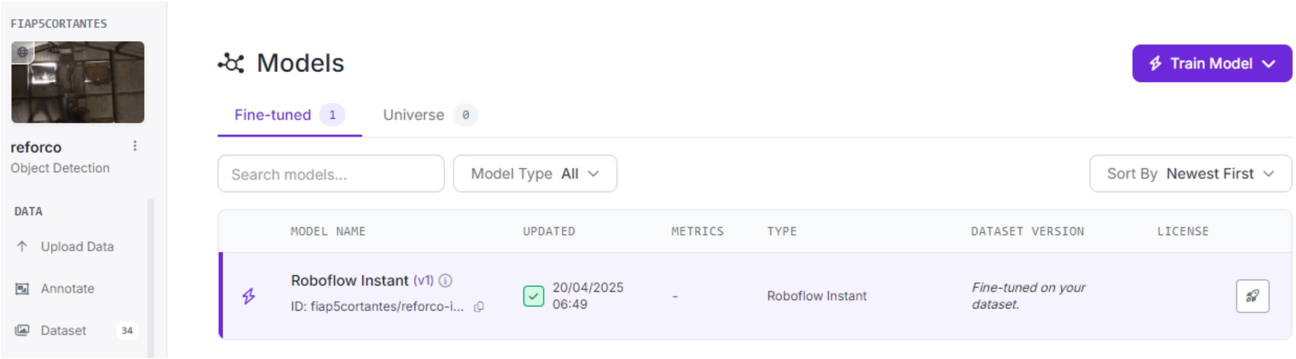
Object Detection
GP2last
Edited a day ago
Public • 325 Images • 0 Models

Object Detection
facas
Edited a day ago
Public • 0 Images • 0 Models

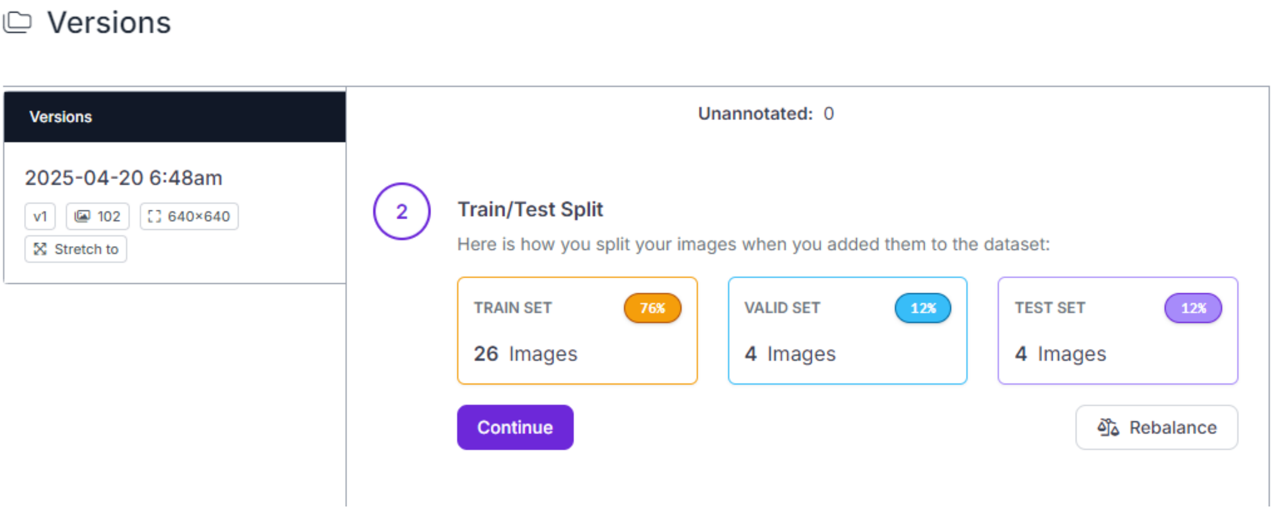
Object Detection
v3_final_com_contr...
Edited a day ago
Pub... • 4664 Imag... • 1 Mod...

Object Detection
reforco
Edited a day ago
Public • 49 Images • 2 Models

Object Detection
faca_tesoura_reforco
Edited a day ago
Public • 43 Images • 0 Models



MODEL NAME	UPDATED	METRICS	TYPE	DATASET VERSION	LICENSE
Roboflow Instant (v1) ID: fiap5cortantes/reforco-i...	20/04/2025 06:49	-	Roboflow Instant	*Fine-tuned on your dataset.*	



Unannotated: 0

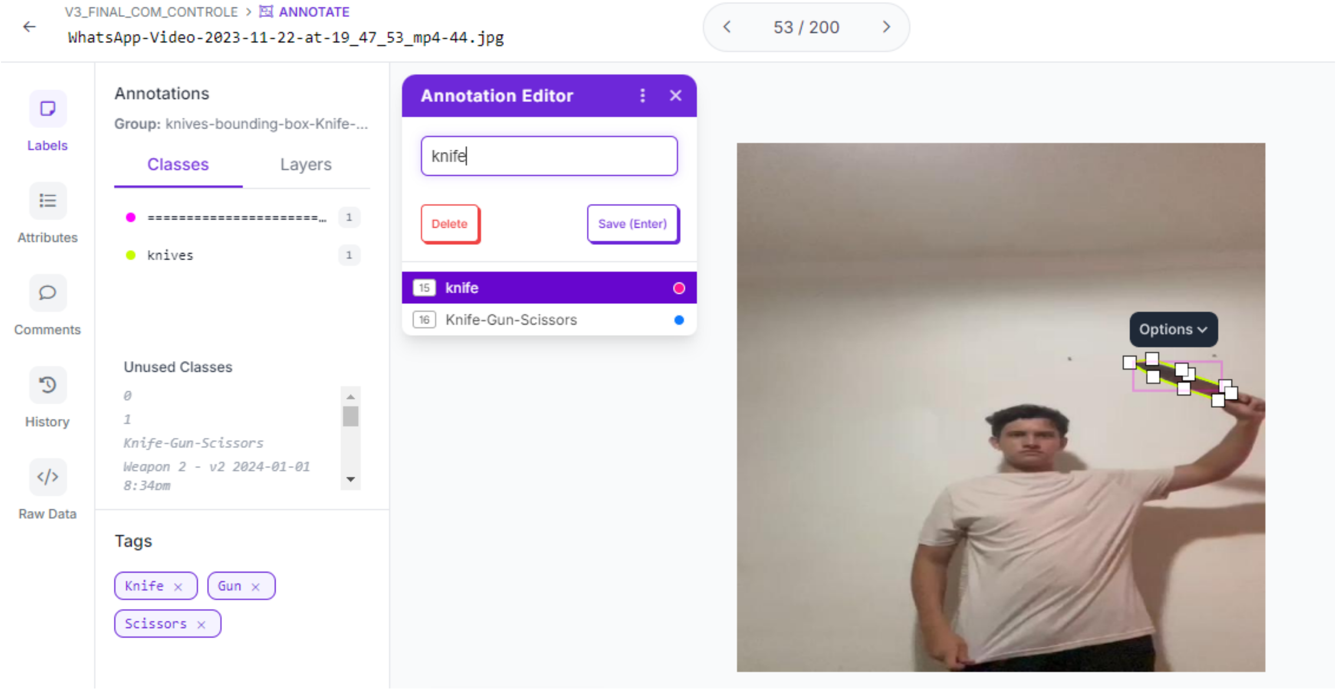
2 **Train/Test Split**

Here is how you split your images when you added them to the dataset:

TRAIN SET 76%
26 Images

VALID SET 12%
4 Images

TEST SET 12%
4 Images



```
#Etapa 3 – Treinamento do modelo YOLO no Colab com o dataset final unificado (v4) 🚀
from getpass import getpass
from roboflow import Roboflow

# 🔒 API segura
api_key = getpass("Digite sua Roboflow API Key: ")
rf = Roboflow(api_key=api_key)
workspace = rf.workspace()

print("Workspace detectado:", workspace.name)
print("\n🔍 Projetos válidos no seu workspace:\n")

# Usa método robusto e seguro
projects = []
try:
```

```
try:
    for project_slug in workspace.list_projects():
        try:
            proj = workspace.project(project_slug)
            print(f"- Nome visível: {proj.name} | Slug: {proj.url}")
            projects.append(proj.url)
        except Exception as e:
            print(f"⚠️ Ignorado: {project_slug} → {str(e)}")
except Exception as e:
    print("Erro ao listar projetos:", str(e))
```

```
Digite sua Roboflow API Key: ··········
loading Roboflow workspace...
Workspace detectado: FIAP5Cortantes

🔍 Projetos válidos no seu workspace:

[{'id': 'fiap5cortantes/controle-scwxf', 'type': 'object-detection', 'name': 'Controle', 'created': 1661384141.107, 'updated': 17456
Erro ao listar projetos: 'NoneType' object is not iterable
```

Configurando YOLOv8

```
!pip install roboflow ultralytics  # Instala Roboflow e YOLOv8

from roboflow import Roboflow
rf = Roboflow(api_key="uK7BzFHdK5qilTZ5iQ1r")  # Substitua pela sua API key
project = rf.workspace("fiap5cortantes").project("cortantesmvp-xme9i")
version = project.version(2)
dataset = version.download("yolov8")  # Formato YOLOv8 padrão
```

```
Collecting roboflow
  Downloading roboflow-1.1.61-py3-none-any.whl.metadata (9.7 kB)
Collecting ultralytics
  Downloading ultralytics-8.3.112-py3-none-any.whl.metadata (37 kB)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from roboflow) (2025.1.31)
Collecting idna==3.7 (from roboflow)
  Downloading idna-3.7-py3-none-any.whl.metadata (9.9 kB)
Requirement already satisfied: cycler in /usr/local/lib/python3.11/dist-packages (from roboflow) (0.12.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from roboflow) (1.4.8)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from roboflow) (3.10.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.11/dist-packages (from roboflow) (2.0.2)
Collecting opencv-python-headless==4.10.0.84 (from roboflow)
  Downloading opencv_python_headless-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (20 kB)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.11/dist-packages (from roboflow) (11.1.0)
Collecting pillow-heif>=0.18.0 (from roboflow)
  Downloading pillow_heif-0.22.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.6 kB)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.11/dist-packages (from roboflow) (2.8.2)
Collecting python-dotenv (from roboflow)
  Downloading python_dotenv-1.1.0-py3-none-any.whl.metadata (24 kB)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from roboflow) (2.32.3)
Requirement already satisfied: six in /usr/local/lib/python3.11/dist-packages (from roboflow) (1.17.0)
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.11/dist-packages (from roboflow) (2.3.0)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.11/dist-packages (from roboflow) (4.67.1)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.11/dist-packages (from roboflow) (6.0.2)
Requirement already satisfied: requests-toolbelt in /usr/local/lib/python3.11/dist-packages (from roboflow) (1.0.0)
Collecting filetype (from roboflow)
  Downloading filetype-1.2.0-py2.py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (4.11.0.86)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (1.14.1)
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (2.6.0+cu124)
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (0.21.0+cu124)
Requirement already satisfied: psutil in /usr/local/lib/python3.11/dist-packages (from ultralytics) (5.9.5)
Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.11/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (2.2.2)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.11/dist-packages (from ultralytics) (0.13.2)
Collecting ultralytics-thop>=2.0.0 (from ultralytics)
  Downloading ultralytics_thop-2.0.14-py3-none-any.whl.metadata (9.4 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->roboflow) (1.3.2)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->roboflow) (4.57.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib->roboflow) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib->roboflow) (3.2.3)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.1.4->ultralytics) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.1.4->ultralytics) (2025.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->roboflow) (3.4
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (3.18.0)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralyti
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch>=1.8.0->ultralytics) (2025.3.2)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=1.8.0->ultralytics)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=1.8.0->ultralytics)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=1.8.0->ultralytics)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=1.8.0->ultralytics)
```

```
    Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
```

```
!cp -r /content/cortantesMVP-2 /content/drive/MyDrive/
```

```
^C
```

```
import os
import yaml
from pathlib import Path

# 1. Carregar os nomes das classes do data.yaml
with open(f"{DATASET_DIR}/data.yaml") as f:
    data = yaml.safe_load(f)
    names = data['names']
print("🔍 Classes no dataset:", names)

# 2. Verificar distribuição de classes em todas as divisões
for split in ['train', 'valid', 'test']:
    print(f"\n📊 Verificando '{split}':")

    images_dir = f"{DATASET_DIR}/{split}/images"
    labels_dir = f"{DATASET_DIR}/{split}/labels"

    # Contador para todas as classes
    class_counts = {name: 0 for name in names}

    # Verificar cada arquivo de anotação
    for label_file in os.listdir(labels_dir):
        with open(f"{labels_dir}/{label_file}") as f:
            for line in f:
                class_id = int(line.split()[0])
                class_name = names[class_id]
                class_counts[class_name] += 1

    # Mostrar resultados
    for class_name, count in class_counts.items():
        print(f"✅ {class_name}: {count} instâncias")

# 3. Verificação adicional para 'scissors'
scissors_in_val = any("scissors" in names for names in data['names'])
print("\n🔎 Tesouras no validação?", "SIM" if scissors_in_val else "NÃO")
```

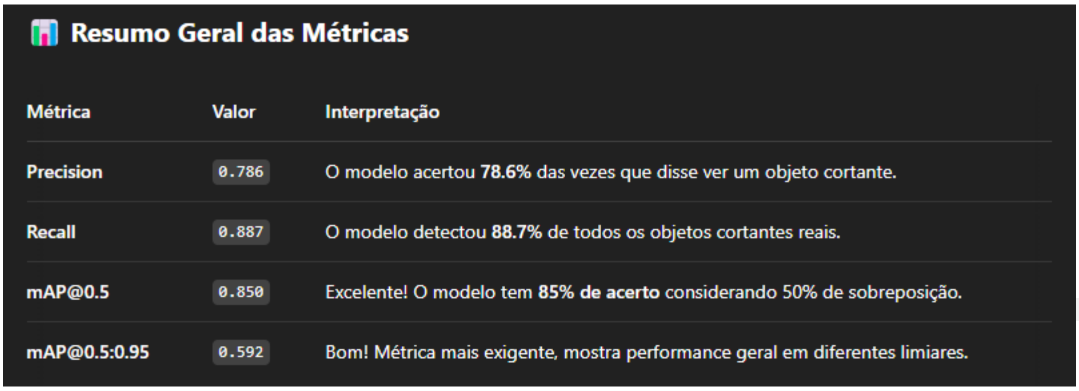
```
🔍 Classes no dataset: ['Knife', 'cutter', 'knife', 'knives', 'scissor', 'scissors', 'sickle', 'weapon']

📊 Verificando 'train':
✅ Knife: 0 instâncias
✅ cutter: 946 instâncias
✅ knife: 652 instâncias
✅ knives: 0 instâncias
✅ scissor: 0 instâncias
✅ scissors: 214 instâncias
✅ sickle: 286 instâncias
✅ weapon: 2 instâncias

📊 Verificando 'valid':
✅ Knife: 0 instâncias
✅ cutter: 128 instâncias
✅ knife: 9 instâncias
✅ knives: 0 instâncias
✅ scissor: 0 instâncias
✅ scissors: 0 instâncias
✅ sickle: 57 instâncias
✅ weapon: 0 instâncias

📊 Verificando 'test':
✅ Knife: 49 instâncias
✅ cutter: 69 instâncias
✅ knife: 3 instâncias
✅ knives: 0 instâncias
✅ scissor: 0 instâncias
✅ scissors: 0 instâncias
✅ sickle: 0 instâncias
✅ weapon: 0 instâncias

🔎 Tesouras no validação? SIM
```

```
# Script para unificar classes (execute antes do treino)
import os
from pathlib import Path

class_mapping = {
    'Knife': 'knife',
    'knives': 'knife',
    'scissor': 'scissors',

}

for split in ['train', 'valid', 'test']:
    label_dir = Path(f"{DATASET_DIR}/{split}/labels")
    for label_file in label_dir.glob('*.txt'):
        with open(label_file, 'r+') as f:
            lines = []
            for line in f:
                class_id, *coords = line.split()
                old_name = names[int(class_id)]
                new_name = class_mapping.get(old_name, old_name)
                new_id = names.index(new_name)
                lines.append(f"{new_id} {' '.join(coords)}\n")
            f.seek(0)
            f.writelines(lines)
```

Treinamento do Modelo YOLO8

```
# 1. Importações essenciais
from google.colab import drive
import torch
from ultralytics import YOLO
import os

# 2. Montar Google Drive
#drive.mount('/content/drive')

# 3. Definir caminhos
DRIVE_BASE = "/content/drive/MyDrive"
DATASET_DIR = f"{DRIVE_BASE}/cortantesMVP-2"
DATA_YAML = f"{DATASET_DIR}/data.yaml"

# 4. Verificar estrutura
print(f"✅ Dataset encontrado em: {DATASET_DIR}")
print("Conteúdo:", os.listdir(DATASET_DIR))
print("\nExemplo de imagens de treino:", os.listdir(f"{DATASET_DIR}/train/images")[:3])

# 5. Configurar dispositivo
device = 'cuda' if torch.cuda.is_available() else 'cpu'
print(f"\n🚀 Dispositivo selecionado: {device.upper()}")

# 6. Treinamento do modelo
model = YOLO('yolov8n.pt')  # Modelo pré-treinado

try:
    results = model.train(
        data=DATA_YAML,
        epochs=30,
        imgsz=640,
        batch=16,
        device=device,
        workers=2  # Reduza se ocorrerem erros de memória
    )
    print("\n✅ Treinamento concluído com sucesso!")

except Exception as e:
    print(f"\n❌ Erro durante o treinamento: {str(e)}")
    print("Dica: Reduza o batch size ou image size se faltar memória")

# 7. Salvar resultados
!mkdir -p "{DRIVE_BASE}/YOLOv8_results"
!cp -r "/content/runs" "{DRIVE_BASE}/YOLOv8_results"
print(f"\n📁 Resultados salvos em: {DRIVE_BASE}/YOLOv8_results")
print(f"✅ Resultados salvos em: {DRIVE_BASE}/YOLOv8_results")
```

```
✅ Dataset encontrado em: /content/drive/MyDrive/cortantesMVP-2
Conteúdo: ['README.dataset.txt', 'README.roboflow.txt', 'data.yaml', 'test', 'train', 'valid']

Exemplo de imagens de treino: ['cutter186_jpg.rf.78d0311377f825bbd036117729951590.jpg', 'cutter186_jpg.rf.93457e5647b7b94117fdda6

🚀 Dispositivo selecionado: CUDA
Ultralytics 8.3.112 🚀 Python-3.11.12 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
engine/trainer: task=detect, mode=train, model=yolov8n.pt, data=/content/drive/MyDrive/cortantesMVP-2/data.yaml, epochs=30, time=
Overriding model.yaml nc=80 with nc=8

                   from  n    params  module                                       arguments
  0                  -1  1       464  ultralytics.nn.modules.conv.Conv             [3, 16, 3, 2]
  1                  -1  1      4672  ultralytics.nn.modules.conv.Conv             [16, 32, 3, 2]
  2                  -1  1      7360  ultralytics.nn.modules.block.C2f             [32, 32, 1, True]
  3                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  4                  -1  2     49664  ultralytics.nn.modules.block.C2f             [64, 64, 2, True]
  5                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  6                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  7                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  8                  -1  1    460288  ultralytics.nn.modules.block.C2f             [256, 256, 1, True]
  9                  -1  1    164608  ultralytics.nn.modules.block.SPPF            [256, 256, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1     37248  ultralytics.nn.modules.block.C2f             [192, 64, 1]
 16                  -1  1     36992  ultralytics.nn.modules.conv.Conv             [64, 64, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    123648  ultralytics.nn.modules.block.C2f             [192, 128, 1]
 19                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 22        [15, 18, 21]  1    752872  ultralytics.nn.modules.head.Detect           [8, [64, 128, 256]]
Model summary: 129 layers, 3,012,408 parameters, 3,012,392 gradients, 8.2 GFLOPs

Transferred 319/355 items from pretrained weights
Freezing layer 'model.22.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks...
AMP: checks passed ✅
train: Fast image access ✅ (ping: 0.4±0.1 ms, read: 4.2±2.5 MB/s, size: 9.9 KB)

train: Scanning /content/drive/MyDrive/cortantesMVP-2/train/labels.cache... 1583 images, 4 backgrounds, 1 corrupt: 100%|█████████
WARNING ⚠️ Box and segment counts should be equal, but got len(segments) = 224, len(boxes) = 2097. To resolve this only boxes wil
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3, met

WARNING ⚠️ val: Slow image access detected (ping: 14.0±28.5 ms, read: 1.2±1.6 MB/s, size: 7.7 KB). Use local storage instead of r

val: Scanning /content/drive/MyDrive/cortantesMVP-2/valid/labels.cache... 127 images, 1 backgrounds, 0 corrupt: 100%|██████████|
Plotting labels to runs/detect/train3/labels.jpg...
optimizer: 'optimizer=auto' found, ignoring 'lr0=0.01' and 'momentum=0.937' and determining best 'optimizer', 'lr0' and 'momentum
optimizer: AdamW(lr=0.000833, momentum=0.9) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias(decay=0.
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/detect/train3
Starting training for 30 epochs...
```



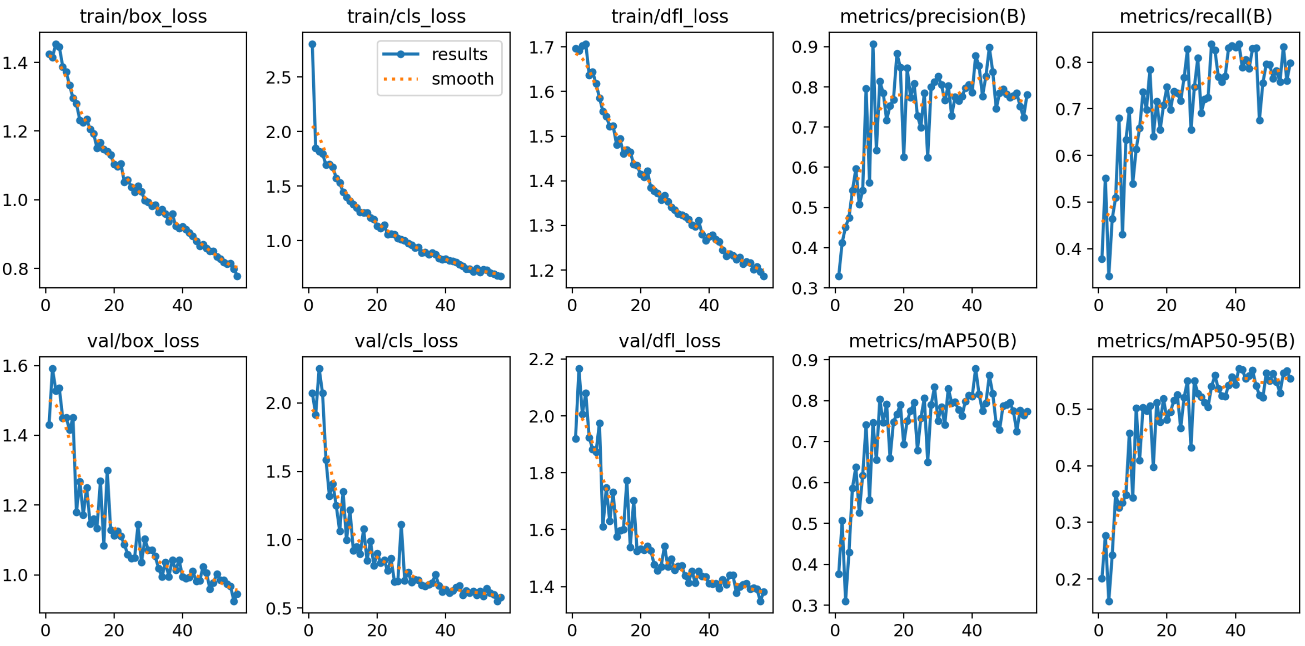
📊 Resumo Geral das Métricas

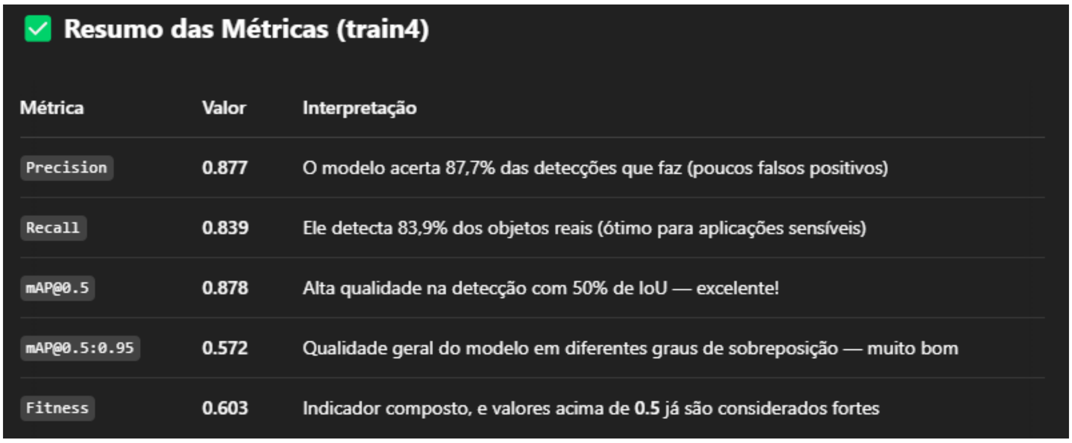
Métrica	Valor	Interpretação
Precision	`0.786`	O modelo acertou **78.6%** das vezes que disse ver um objeto cortante.
Recall	`0.887`	O modelo detectou **88.7%** de todos os objetos cortantes reais.
mAP@0.5	`0.850`	Excelente! O modelo tem **85% de acerto** considerando 50% de sobreposição.
mAP@0.5:0.95	`0.592`	Bom! Métrica mais exigente, mostra performance geral em diferentes limiares.

```
 from ultralytics import YOLO

model = YOLO('yolov8s.pt')  # Modelo mais preciso
model.train(
    data=f"{DATASET_DIR}/data.yaml",
    epochs=80,
    batch=16,
    imgsz=640,
    single_cls=False,  # Mantém multi-classes
    overlap_mask=True,
    optimizer='AdamW',
    lr0=0.001,
    patience=15
)
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8s.pt to 'yolov8s.pt'...

100%|██████████| 21.5M/21.5M [00:00<00:00, 356MB/s]
Ultralytics 8.3.112 🚀 Python-3.11.12 torch-2.6.0+cu124 CUDA:0 (Tesla T4, 15095MiB)
engine/trainer: task=detect, mode=train, model=yolov8s.pt, data=/content/drive/MyDrive/cortantesMVP-2/data.yaml, epochs=80, time=
Overriding model.yaml nc=80 with nc=8

                   from  n    params  module                                       arguments
  0                  -1  1       928  ultralytics.nn.modules.conv.Conv             [3, 32, 3, 2]
  1                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  2                  -1  1     29056  ultralytics.nn.modules.block.C2f             [64, 64, 1, True]
  3                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  4                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  5                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  6                  -1  2    788480  ultralytics.nn.modules.block.C2f             [256, 256, 2, True]
  7                  -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  8                  -1  1   1838080  ultralytics.nn.modules.block.C2f             [512, 512, 1, True]
  9                  -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    591360  ultralytics.nn.modules.block.C2f             [768, 256, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 16                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 19                  -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1   1969152  ultralytics.nn.modules.block.C2f             [768, 512, 1]
 22        [15, 18, 21]  1   2119144  ultralytics.nn.modules.head.Detect           [8, [128, 256, 512]]
Model summary: 129 layers, 11,138,696 parameters, 11,138,680 gradients, 28.7 GFLOPs

Transferred 349/355 items from pretrained weights
Freezing layer 'model.22.dfl.conv.weight'
AMP: running Automatic Mixed Precision (AMP) checks...
AMP: checks passed ✅
train: Fast image access ✅ (ping: 1.2±0.4 ms, read: 5.4±3.4 MB/s, size: 9.9 KB)

train: Scanning /content/drive/MyDrive/cortantesMVP-2/train/labels.cache... 1583 images, 4 backgrounds, 1 corrupt: 100%|█████████
WARNING ⚠️ Box and segment counts should be equal, but got len(segments) = 224, len(boxes) = 2097. To resolve this only boxes wil
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3, met

val: Fast image access ✅ (ping: 7.0±9.1 ms, read: 1.6±3.0 MB/s, size: 7.7 KB)

val: Scanning /content/drive/MyDrive/cortantesMVP-2/valid/labels.cache... 127 images, 1 backgrounds, 0 corrupt: 100%|██████████|
Plotting labels to runs/detect/train4/labels.jpg...
optimizer: AdamW(lr=0.001, momentum=0.937) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias(decay=0.0
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/detect/train4
Starting training for 80 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size

  0%|          | 0/99 [00:00<?, ?it/s]
       1/80      3.7G      1.335       11.3      1.648         42        640:   0%|          | 0/99 [00:00<?, ?it/s]
```

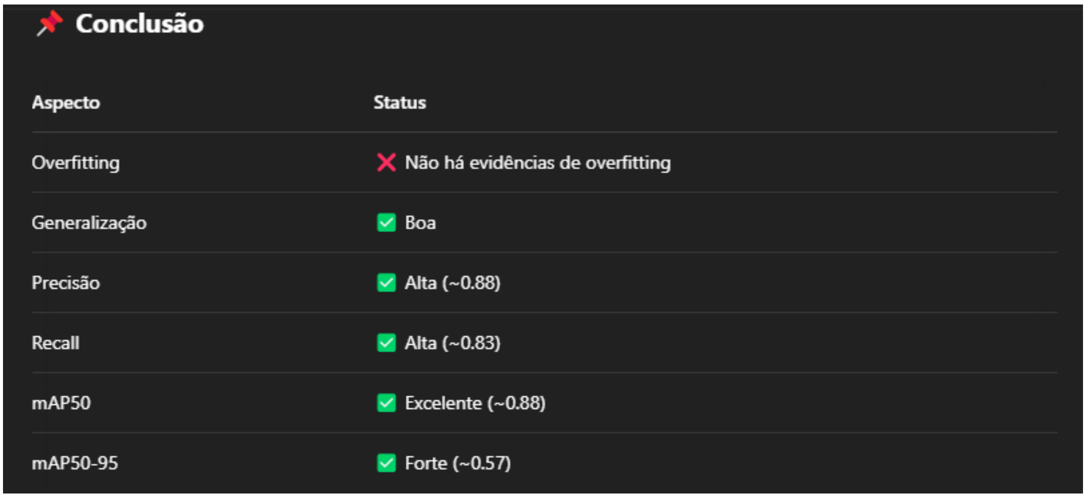
Train4





✅ Resumo das Métricas (train4)

Métrica	Valor	Interpretação
`Precision`	0.877	O modelo acerta 87,7% das detecções que faz (poucos falsos positivos)
`Recall`	0.839	Ele detecta 83,9% dos objetos reais (ótimo para aplicações sensíveis)
`mAP@0.5`	0.878	Alta qualidade na detecção com 50% de IoU — excelente!
`mAP@0.5:0.95`	0.572	Qualidade geral do modelo em diferentes graus de sobreposição — muito bom
`Fitness`	0.603	Indicador composto, e valores acima de **0.5** já são considerados fortes



📌 Conclusão

Aspecto	Status
Overfitting	❌ Não há evidências de overfitting
Generalização	✅ Boa
Precisão	✅ Alta (~0.88)
Recall	✅ Alta (~0.83)
mAP50	✅ Excelente (~0.88)
mAP50-95	✅ Forte (~0.57)

```
#✅ Código de Teste SMTP para Gmail
import smtplib

# Configurações SMTP
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
EMAIL_USER = "visionfiap@gmail.com"
EMAIL_PASSWORD = "ebet xgvh mjje gpzh"  # Senha de App do Gmail

try:
    # Conectar ao servidor
    server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
    server.starttls()
    server.login(EMAIL_USER, EMAIL_PASSWORD)
    print("✅ Conexão e autenticação SMTP bem-sucedida!")
    server.quit()
except smtplib.SMTPAuthenticationError as e:
    print("❌ Erro de autenticação SMTP:", e.smtp_error.decode())
except Exception as e:
    print("❌ Outro erro:", str(e))
```

```
✅ Conexão e autenticação SMTP bem-sucedida!
```

```
# 📧 Sistema de Alertas com Capturas (Versão Final)
import smtplib
import email.utils
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
import cv2
from datetime import datetime
import os
from collections import deque
import time
from ultralytics import YOLO

# ================= CONFIGURAÇÕES =================
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
EMAIL_USER = "visionfiap@gmail.com"
```

```
EMAIL_PASSWORD = "ebet xgvh mjje gpzh"  # Senha de app do Google
ALERT_RECIPIENTS = ["dassenhoritas@terra.com.br", "contato@dassenhoritas.com.br"]
ALERT_DIR = "/content/drive/MyDrive/alertas"
os.makedirs(ALERT_DIR, exist_ok=True)

# ================= CONTROLE DE ALERTAS =================
alert_history = deque(maxlen=10)  # Máximo 10 alertas/minuto

def should_send_alert():
    """Previne flood de e-mails"""
    now = time.time()
    if len(alert_history) >= 10 and (now - alert_history[0]) < 60:
        print("⚠️ Muitos alertas recentes - Modo silencioso ativado")
        return False
    alert_history.append(now)
    return True

def send_alert(frame, detected_objects, confidence):
    """Envia e-mail com imagem anexada"""
    if not should_send_alert():
        return

    try:
        # 1. Salvar imagem temporária
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        img_path = f"{ALERT_DIR}/alerta_{timestamp}.jpg"
        cv2.imwrite(img_path, frame)

        # 2. Criar e-mail
        msg = MIMEMultipart()
        msg['Subject'] = f"🚨 ALERTA: {detected_objects} detectado (Conf: {confidence:.0%})"
        msg['From'] = EMAIL_USER
        msg['To'] = ", ".join(ALERT_RECIPIENTS)

        # Corpo do e-mail (HTML)
        html = f"""
        <h2>VisionGuard - Detecção de Objeto Perigoso</h2>
        <p><b>Objeto:</b> {detected_objects}</p>
        <p><b>Confiança:</b> {confidence:.0%}</p>
        <p><b>Horário:</b> {timestamp.replace('_', ' ')}</p>
        <img src="cid:alerta_image" width="800">
        """
        msg.attach(MIMEText(html, 'html'))

        # Anexar imagem
        with open(img_path, 'rb') as f:
            img_data = f.read()
        image = MIMEImage(img_data, name=os.path.basename(img_path))
        image.add_header('Content-ID', '<alerta_image>')
        msg.attach(image)

        # 3. Enviar
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
            server.starttls()
            server.login(EMAIL_USER, EMAIL_PASSWORD)
            server.send_message(msg)

        print(f"📧 Alerta enviado: {detected_objects} ({confidence:.0%})")

    except Exception as e:
        print(f"❌ Erro no envio: {str(e)}")

# ================= PROCESSAMENTO DE VÍDEO =================
def process_video(video_path, model_path="runs/detect/train/weights/best.pt"):
    cap = cv2.VideoCapture(video_path)
    model = YOLO(model_path)

    # Configuração do vídeo de saída
    frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    fps = int(cap.get(cv2.CAP_PROP_FPS))
    output_path = os.path.join(ALERT_DIR, "video_processado.mp4")
    out = cv2.VideoWriter(output_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (frame_width, frame_height))

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

        results = model.predict(frame, conf=0.5)
        annotated_frame = results[0].plot()  # Frame com marcações
        # Salva cada frame processado
        out.write(annotated_frame)

        for box in results[0].boxes:
            class_name = model.names[int(box.cls)]
            confidence = float(box.conf)
            if class_name in ['Knife', 'knife', 'scissors', 'weapon'] and confidence > 0.6:
                send_alert(annotated_frame, class_name, confidence)

    # Libera recursos
    cap.release()
    out.release()
    print(f"🎥 Vídeo processado salvo em: {output_path}")

# ================= EXECUÇÃO =================
if __name__ == "__main__":
    process_video("/content/drive/MyDrive/Hackaton FIAP5/video.mp4")
    print("✅ Processamento concluído! Verifique:")
    print(f"- Vídeo com marcações: {ALERT_DIR}/video_processado.mp4")
    print(f"Alertas por e-mail e imagens em: {ALERT_DIR}/alerta_*.jpg")  # Linha corrigida
```

```
0: 384x640 1 knife, 38.5ms
Speed: 2.6ms preprocess, 38.5ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)
📨 Alerta enviado: knife (89%)

0: 384x640 1 knife, 9.7ms
Speed: 3.8ms preprocess, 9.7ms inference, 1.3ms postprocess per image at shape (1, 3, 384, 640)
📨 Alerta enviado: knife (89%)

0: 384x640 1 knife, 20.4ms
Speed: 5.7ms preprocess, 20.4ms inference, 2.0ms postprocess per image at shape (1, 3, 384, 640)
📨 Alerta enviado: knife (88%)

0: 384x640 1 knife, 14.1ms
Speed: 3.7ms preprocess, 14.1ms inference, 1.4ms postprocess per image at shape (1, 3, 384, 640)
📨 Alerta enviado: knife (86%)

0: 384x640 1 knife, 8.9ms
Speed: 3.2ms preprocess, 8.9ms inference, 1.3ms postprocess per image at shape (1, 3, 384, 640)
📨 Alerta enviado: knife (86%)

0: 384x640 1 knife, 7.2ms
Speed: 3.0ms preprocess, 7.2ms inference, 1.3ms postprocess per image at shape (1, 3, 384, 640)
📨 Alerta enviado: knife (83%)

0: 384x640 1 knife, 8.8ms
Speed: 5.6ms preprocess, 8.8ms inference, 1.4ms postprocess per image at shape (1, 3, 384, 640)
📨 Alerta enviado: knife (82%)

0: 384x640 1 knife, 14.0ms
Speed: 5.2ms preprocess, 14.0ms inference, 2.3ms postprocess per image at shape (1, 3, 384, 640)
📨 Alerta enviado: knife (86%)

0: 384x640 1 knife, 11.6ms
Speed: 6.0ms preprocess, 11.6ms inference, 1.3ms postprocess per image at shape (1, 3, 384, 640)
📨 Alerta enviado: knife (87%)

0: 384x640 1 knife, 10.7ms
Speed: 4.8ms preprocess, 10.7ms inference, 1.4ms postprocess per image at shape (1, 3, 384, 640)
📨 Alerta enviado: knife (88%)

0: 384x640 1 knife, 9.0ms
Speed: 4.0ms preprocess, 9.0ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)
⚠️ Muitos alertas recentes - Modo silencioso ativado

0: 384x640 1 knife, 10.2ms
Speed: 3.6ms preprocess, 10.2ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)
⚠️ Muitos alertas recentes - Modo silencioso ativado

0: 384x640 1 knife, 9.5ms
Speed: 7.0ms preprocess, 9.5ms inference, 1.6ms postprocess per image at shape (1, 3, 384, 640)
⚠️ Muitos alertas recentes - Modo silencioso ativado

0: 384x640 1 knife, 14.0ms
Speed: 3.4ms preprocess, 14.0ms inference, 1.7ms postprocess per image at shape (1, 3, 384, 640)
⚠️ Muitos alertas recentes - Modo silencioso ativado

0: 384x640 1 knife, 15.3ms
```

```
from google.colab import drive
import shutil

# 1️⃣ Montar seu Google Drive
#drive.mount('/content/drive')
```

```
# 2️⃣ Copiar a pasta runs para uma pasta no seu Drive
shutil.copytree("/content/runs", "/content/drive/MyDrive/Hackaton FIAP5/runs", dirs_exist_ok=True)

print("✅ Pasta 'runs/' copiada para seu Drive com sucesso!")
```

```
✅ Pasta 'runs/' copiada para seu Drive com sucesso!
```