**Etapas do Projeto:**

1. **Preparação do Ambiente de Desenvolvimento**
2. **Coleta e Preparação de Dados**
3. **Treinamento do Modelo YOLOv9**
4. **Implementação do Script de Reconhecimento em Tempo Real**
5. **Desenvolvimento da Aplicação Mobile (Opcional)**

**1. Preparação do Ambiente de Desenvolvimento**

Primeiro, precisamos garantir que você tenha todas as dependências necessárias instaladas. Vamos usar o YOLOv9, que é uma evolução do YOLO (You Only Look Once) para detecção de objetos.

1. **Instalar Python e dependências**:
   * Certifique-se de ter Python instalado. Recomendo Python 3.7 ou superior.
   * Instale os pacotes necessários usando pip:

pip install torch torchvision opencv-python

pip install pycocotools

pip install matplotlib

1. **Clonar o repositório YOLOv9**:
   * Clone o repositório oficial do YOLOv9 (substitua YOLOv9-repo-url com a URL correta quando estiver disponível):

git clone YOLOv9-repo-url

cd YOLOv9

**2. Coleta e Preparação de Dados**

Para treinar um modelo eficaz, você precisará de um conjunto de dados bem rotulado contendo imagens de animais em estradas. Algumas fontes possíveis incluem:

* **COCO dataset**: Contém uma variedade de imagens de animais, mas você pode precisar filtrar as imagens relevantes.
* **Dataset próprio**: Coletar imagens usando câmeras de monitoramento de estradas ou de bancos de imagens.

**Anotação dos Dados**:

* Use ferramentas como LabelImg para anotar suas imagens. Cada imagem deve ter caixas delimitadoras ao redor dos animais.

**3. Treinamento do Modelo YOLOv9**

1. **Organização dos dados**:
   * Estruture seus dados de acordo com o formato esperado pelo YOLOv9.

├── dataset/

│ ├── images/

│ │ ├── train/

│ │ ├── val/

│ ├── labels/

│ │ ├── train/

│ │ ├── val/

1. **Configuração e treinamento**:
   * Modifique o arquivo de configuração do YOLOv9 para refletir o número de classes e caminhos de seus dados.
   * Execute o treinamento:

python train.py --img 640 --batch 16 --epochs 50 --data dataset.yaml --cfg yolov9.yaml --weights yolov9.pth

**4. Implementação do Script de Reconhecimento em Tempo Real**

Uma vez que o modelo está treinado, você pode usar o modelo para detetar animais em um feed de vídeo em tempo real.

1. **Script para deteção em tempo real**:

import cv2

import torch

from yolov9.models.experimental import attempt\_load

from yolov9.utils.general import non\_max\_suppression, scale\_coords

from yolov9.utils.datasets import LoadStreams, LoadImages

from yolov9.utils.plots import plot\_one\_box

# Carregar o modelo treinado

model = attempt\_load('path/to/best.pt', map\_location='cpu')

# Configuração da câmera

cap = cv2.VideoCapture(0)

while True:

ret, frame = cap.read()

if not ret:

break

img = frame[..., ::-1] # Convert BGR to RGB

img = torch.from\_numpy(img).float().div(255.0).unsqueeze(0)

# Inference

pred = model(img)[0]

pred = non\_max\_suppression(pred, 0.25, 0.45, classes=None, agnostic=False)

# Process and visualize results

for det in pred:

if len(det):

scale\_coords(img.shape[2:], det[:, :4], frame.shape).round()

for \*xyxy, conf, cls in det:

label = f'{names[int(cls)]} {conf:.2f}'

plot\_one\_box(xyxy, frame, label=label, color=[255, 0, 0], line\_thickness=2)

cv2.imshow('Frame', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()

**5. Desenvolvimento da Aplicação Mobile (Opcional)**

Para a aplicação mobile, você pode usar frameworks como o Flutter ou React Native junto com uma biblioteca que permita a execução de modelos PyTorch, como o PyTorch Mobile ou TensorFlow Lite (convertendo o modelo).

**Recursos Adicionais**

* **Documentação do YOLOv9**: Consulte a documentação oficial para detalhes específicos sobre a instalação e configuração.
* **Tutoriais online**: Existem muitos tutoriais disponíveis que podem ajudá-lo a entender melhor o processo de treinamento e inferência com YOLO.