# 基于 OpenCV+YOLOv4-Tiny 的选择性目标检测系统

**学号：10185101288 姓名：马一凡**

## 摘要

本文提出并实现了一种基于YOLOv4-Tiny模型的选择性道路目标检测系统。该系统通过设置图像上的交互按钮，使用户能够选择性地检测和显示视频中的特定目标类别，包括车辆、卡车、公交车和行人。系统采用OpenCV实现，利用鼠标事件响应机制，实现了对视频中不同目标的实时检测和可视化。本文详细描述了系统的设计与实现过程，并通过实际视频测试验证了系统的有效性和交互性。

## 1. 引言

目标检测是计算机视觉中的一个重要任务，它在安防监控、自动驾驶等领域有广泛应用。YOLOv4-Tiny是一种高效的目标检测算法，具有检测速度快、精度高的特点。实际应用中，用户常常需要针对特定目标进行检测。为此，本文设计并实现了一种基于YOLOv4-Tiny的选择性道路目标检测系统，用户可以通过点击按钮选择需要检测的目标类别，包括车辆、卡车、公交车和行人。

## 2. 系统设计与实现

### 2.1 系统架构

系统主要由以下几个模块组成：

- 模型加载与初始化

- 视频捕获与处理

- 交互按钮绘制与鼠标事件处理

- 目标检测与结果显示

### 2.2 技术理解

本系统利用YOLOv4-Tiny模型进行目标检测，该模型在保证检测精度的同时，具有较快的推理速度。YOLO（You Only Look Once）系列模型的设计理念是将目标检测问题转换为回归问题，直接从图像中预测出边界框和类别概率。YOLOv4-Tiny是YOLOv4的轻量化版本，适用于计算资源有限的场景。

1. 模型加载与初始化：在这一部分，系统首先加载了YOLOv4-Tiny的模型结构和预训练权重。模型的输入参数设置为图像大小416x416，归一化系数为1/255，以适应YOLO的输入要求。
2. 视频捕获与处理：系统利用OpenCV的VideoCapture功能读取视频文件，并对每一帧图像进行处理。视频帧的尺寸被缩小至1280x720，以加快处理速度并节省计算资源。
3. 交互按钮绘制与鼠标事件处理：系统设计了一个drawButton类，用于在图像上绘制交互按钮。用户可以通过点击按钮选择检测特定类别的道路目标，包括车辆、卡车、公交车和行人。cv2.setMouseCallback函数用于设置鼠标事件的回调函数，当用户点击按钮时，系统记录下所选按钮的索引。
4. 目标检测与结果显示：在每一帧图像中，系统利用YOLOv4-Tiny模型进行目标检测，获取检测框的位置信息和类别信息。根据用户选择，系统仅显示特定类别的检测结果，并通过绘制矩形框和分类标签进行可视化。同时，系统对选中的按钮进行高亮显示，以提供更好的用户体验。

该系统的技术实现中，关键技术点包括YOLOv4-Tiny模型的高效目标检测、OpenCV的视频处理与图像绘制功能，以及交互按钮的设计与事件响应机制。

### 2.3 模型加载与初始化

首先，我们加载YOLOv4-Tiny模型的结构和权重，并设置模型的输入参数。具体代码如下：

import cv2

import numpy as np

# 导入YOLOv4-Tiny网络模型结构和权重

net = cv2.dnn.readNet('dnn\_model/yolov4-tiny.cfg', 'dnn\_model/yolov4-tiny.weights')

# 定义目标检测模型

model = cv2.dnn\_DetectionModel(net)

model.setInputParams(size=(416, 416), scale=1/255)

### 2.4 获取分类文本信息

读取存储目标类别的文本文件，将每个类别的名称保存到列表中。

classes = []

with open('dnn\_model/classes.txt') as file\_obj:

for class\_name in file\_obj.readlines():

classes.append(class\_name.strip())

### 2.5 视频捕获与处理

使用OpenCV捕获视频流，并对每一帧进行处理。视频帧处理包括缩放、目标检测和按钮绘制。

filepath = 'C:\\GameDownload\\Deep Learning\\trafficvideo3.mp4'

cap = cv2.VideoCapture(filepath)

### 2.6 交互按钮绘制与鼠标事件处理

通过定义鼠标回调函数，实现鼠标点击事件的响应。根据点击位置判断用户选择的目标类别，并在视频中绘制相应的检测框。

```python

button\_index = None # 存放哪个按键被点亮了

def click\_button(event, x, y, flags, params):

global button\_index

if event == cv2.EVENT\_LBUTTONDOWN:

for index, pt in enumerate(np.array(buttonList)):

is\_inside = cv2.pointPolygonTest(pt, (x,y), False)

if is\_inside > 0:

button\_index = index

cv2.namedWindow('Image')

cv2.setMouseCallback('Image', click\_button)

usenames = ['all', 'person', 'car', 'bus', 'truck']

button = drawButton(usenames)

```

### 2.7 目标检测与结果显示

对每一帧视频进行目标检测，根据用户选择绘制检测框，并显示检测结果。具体代码如下：

def drawbbx(img, x, y, w, h, predName, score):

cv2.rectangle(img, (x, y), (x+w, y+h), (255,255,0), 1)

cv2.putText(img, predName, (x,y+h+20), cv2.FONT\_HERSHEY\_COMPLEX, 1, (0,255,0), 2)

cv2.putText(img, str(int(score\*100))+'%', (x,y-5), cv2.FONT\_HERSHEY\_COMPLEX, 1, (0,255,255), 2)

while True:

ret, frame = cap.read()

frame = cv2.resize(frame, (1280,720))

if cap.get(cv2.CAP\_PROP\_POS\_FRAMES) == cap.get(cv2.CAP\_PROP\_FRAME\_COUNT):

cap.set(cv2.CAP\_PROP\_POS\_FRAMES, 0)

classids, scores, bboxes = model.detect(frame, 0.5, 0.3)

button.drawRec\_many(frame)

buttonList = button.recList

for class\_id, score, bbox in zip(classids, scores, bboxes):

x, y, w, h = bbox

class\_name = classes[class\_id]

for index, name in enumerate(usenames):

if class\_name == name and index == button\_index:

drawbbx(frame, x, y, w, h, class\_name, score)

cv2.rectangle(frame, buttonList[index][0], buttonList[index][2], (0,0,255), 5)

elif name == 'all' and index == button\_index:

drawbbx(frame, x, y, w, h, class\_name, score)

cv2.rectangle(frame, buttonList[index][0], buttonList[index][2], (0,0,255), 5)

cv2.imshow('Image', frame)

if cv2.waitKey(30) & 0xFF == 27:

break

cap.release()

cv2.destroyAllWindows()

## 总结

本文设计并实现了一种基于YOLOv4-Tiny的选择性目标检测系统，通过交互按钮的方式，用户可以方便地选择需要检测的目标类别，如车辆、卡车、公交车和行人。系统在视频处理和目标检测方面表现出色，具有较高的应用价值，特别是在道路监控和交通管理领域。

在项目的实施过程中，不仅完成了预期的技术目标，还通过实际操作和实践，深化了对相关技术的理解和掌握。未来，可以在此基础上，进一步优化系统性能，扩展更多的功能，如实现多目标跟踪、动态目标预测等，提升系统的智能化水平和应用范围。

## 参考文献

1. Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection", arXiv:2004.10934.

2. OpenCV Documentation, https://docs.opencv.org/

3. "YOLOv4-Tiny: Real-Time Object Detection", https://github.com/AlexeyAB/darknet