Aplicações com o modelo yolo v4

Tema: Visão Computacional

Laboratório de Inovação e Automação 1 (LIA 1)

Autores: Gustavo Mota Barros e Pedro Ferreira Galvão Neto

O projeto desenvolvido consiste na utilização de modelos de inteligência artificial voltados a área de visão computacional, para o reconhecimento de objetos.

Este projeto apresentará como o modelo se comporta em relação a imagens e vídeos, visando mostrar a sua grande utilidade no processamento de imagens presentes no dia a dia.

Sobre o modelo YOLO V4:

O YOLO v4 é um modelo de visão computacional capaz de detectar objetos em imagens, por exemplo, se temos uma imagem que contem uma pessoa e um carro, o yolo é capaz de analisar essa imagem, detectando as classe dos objetos reconhecidos (que condizem com os dados de seu treinamento), que nesse exemplo seriam “car” e “person”, reconhecendo também a posição dos objetos na imagem, que são retornadas para o usuário junto com as detecções das classes e o score, o qual demonstra a “certeza” do modelo em relação a cada classificação dos objetos .

Será utilizado um modelo pré-treinado para o conjunto de dados MS COCO.

Diante disso, como utilizaremos o modelo yolo v4 para o reconhecimento de imagens e vídeos com a biblioteca cv2 do OpenCV, devemos baixar os arquivos yolov4.weights e yolov4.cfg que são disponibilizados no seguinte repositorio: https://github.com/AlexeyAB/darknet. Também, devemos baixar, neste repositório, o arquivo coco.names que se encontra na pasta cfg.

O processo geral do modelo para esse projeto será o carregamento da rede neural junto ao modelo de detecção do OpenCV e seus parâmetros de entrada.

Carregamento de dados para as detecções :

O carregamento de imagens realizado na aplicação de detecção de objetos em imagens foi realizado utilizando a api de busca do duckduckgo disponibilizada para python. Na forma de carregamento desenvolvida você escolhe a classe a ser testada, e serão processadas algumas imagens dessa classe para que o modelo possa detectá-las.

Na etapa de carregamento de vídeos, além de ser possível carregar um vídeo do diretório, foi implementada a função de se escolher um vídeo do youtube para que este seja detectado pelo modelo. Nessa forma de carregamento o vídeo referente ao url coletado é carregado no diretório do código sendo, posteriormente, aplicado ao modelo para que as detecções sejam realizadas. As ferramentas utilizadas nessa etapa foram a biblioteca pafy e a api youtube-dl, que no momento atual apresenta conflitos de conexão com o youtube, por conta disso, preferiu-se carregar o vídeo antes de rodá-lo ao invés de realizar a detecção da stream do vídeo.

Na última aplicação é realizada uma conexão simultânea à câmera de um celular conectado localmente à aplicação por meio do aplicativo droidcam.

Código implementado

Carregando as bibliotecas:

```
#Primeiro, deve-se importar as seguintes bibliotecas
import cv2
import time
import numpy as np
import random as rd
import pafy
import os
import csv
```

Carregando o modelo pré-treinado:

```
# carregando os pesos do modelo
net = cv2.dnn.readNet('yolov4-tiny.weights', 'yolov4-tiny.cfg') # para
a  versão tiny

# net = cv2.dnn.readNet('yolov4.weights', 'yolov4.cfg') # para a versão
padrão(mais pesada)

# criando o modelo
model = cv2.dnn.DetectionModel(net)

# setando os parâmetros de entrada para o modelo utilizado
model.setInputParams(size=(416,416), scale=(1/255)) # para o modelo
tiny

# model.setInputParams(size=(608,608), scale=(1/255)) # para o modelo
padrão
```

Carregando os dados para funcionamento do modelo:

```
# carregando os nomes para uma lista
class_names = []
with open('coco_names.txt','r') as f:
    class_names = [cname.strip() for cname in f.readlines()]
# definindo as cores de cada nome
name_color = []
for i in class_names:
    color
=(((sum(map(ord,i))*2)%255),((sum(map(ord,i))*3)%255),((sum(map(ord,i))
*5)%255))
    name_color.append(color)
```

Código para detecção de imagens:

```
# Primeiro devemos coletar a imagem a ser detectada
#iremos usar o duckduckgo para isso
from duckduckgo_search import ddg_images
import requests

def search_img(img_name, num_img=15):
    ddg_img = ddg_images(img_name, max_results=num_img)
    img_list = [dicio['image'] for dicio in ddg_img]
    return img_list

img_name = input("Digite em inglês o nome do objeto que deseja
testar:")

if img_name not in class_names:
    print("O modelo não é capaz de reconhecer este objeto. Tente
novamente!")
else:
    images = search_img(img_name)
    img_list = []
    for i in images:
        print("Carregando links válidos...")
        try:
            if requests.get(i).status_code in range(400,500):
                images.remove(i)
```

```
            else:
                response = requests.get(i)
                img_array = np.array(bytearray(response.content),
dtype=np.uint8)
                img = cv2.imdecode(img_array, cv2.IMREAD_COLOR)
                res_img = cv2.resize(img,(416,416))
                img_list.append(res_img)
        except:
            images.remove(i)

# agora iremos realizar a detecção na imagem
for i in img_list:
    model_image = i.copy()

    classes, scores, boxes = model.detect(model_image , 0.1, 0.2)

    for (classid, score, box) in zip(classes, scores, boxes):
        color = name_color[classid]
        label = f"{class_names[classid]}: {score}"

        cv2.rectangle(model_image, box, color, 2)
        cv2.putText(model_image, label, (box[0], box[1] + 15),
cv2.FONT_HERSHEY_COMPLEX, 0.5, color, 2)
        print(f"class: {class_names[classid]} ---> score: {score}")

    cv2.imshow("detections", model_image)
    cv2.waitKey(0)
```

Código para uso de webcam:

```
# primeiro devemos escolher o vídeo que será capturado
cap =  cv2.VideoCapture(0) # aqui vai o número referente ao dispositivo
#código de detecção
while cap.isOpened():

    _, frame = cap.read()
    start = time.time()
    classes, scores, boxes = model.detect(frame,0.1,0.2)
    end = time.time()

    for (classid, score, box) in zip(classes, scores, boxes):
        color = name_color[classid]
```

```
        label = f"{class_names[classid]}: {score}"

        cv2.rectangle(frame, box, color, 2)
        cv2.putText(frame, label, (box[0], box[1] - 10),
cv2.FONT_HERSHEY_COMPLEX, 0.5, color, 2)

    fps_label = f"FPS: {round((1/(end - start)),2)}"

    cv2.putText(frame, fps_label, (0,25), cv2.FONT_HERSHEY_COMPLEX, 1,
(0,0,0), 5)
    cv2.putText(frame, fps_label, (0,25), cv2.FONT_HERSHEY_COMPLEX, 1,
(0,255,0), 3)

    cv2.imshow("detections", frame)

    if cv2.waitKey(1) == 27:
        break

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

Código para leitura e detecção dos objetos em um vídeo:

```
''' Para que o recurso de vídeos do youtube funcione, é necessário
instalar a biblioteca youtube-dl e modificar o arquivo
youtube.py que se encontra na pasta extractor, indo na linha 1794 e
comentando-a. Após isso, acesse a biblioteca pafy e comente
as linhas 53, 54 e 55 do arquivo backend_youtube_dl.py '''

from tqdm.notebook import tqdm

url =
"https://www.youtube.com/watch?v=iBlRm6BzgqY&t=904s&ab_channel=LucasSou
sa" #basta colocar o url do seu vídeo aqui
videoPafy = pafy.new(url)

def getNormalQuality(videoPafy):
    for stream in videoPafy.videostreams:
        if stream.resolution == '1280x720' and stream.extension
=='mp4':
            return stream
    return videoPafy.getbest(preftype='mp4')
```

```
cap = cv2.VideoCapture(getNormalQuality(videoPafy).url)
filename = 'test_video.mp4'
codec = cv2.VideoWriter_fourcc(*'XVID')
test_out = cv2.VideoWriter(filename, codec, 30, (1280, 720))

duration = 60*1000 # duração do vídeo em segundos
pbar = tqdm(total=duration , desc="Carregando Vídeo: ")
n = 0
k = 0
while True:

    _, frame = cap.read()
    test_out.write(frame)

    vid_time = cap.get(cv2.CAP_PROP_POS_MSEC)
    n = vid_time - k
    k = vid_time
    pbar.update(n)

    if vid_time >= duration or not _:
        break

test_out.release()
cv2.destroyAllWindows()
cap.release()

cap = cv2.VideoCapture('test_video.mp4')
# Para que a fluidez do vídeo seja mantida iremos escrever um arquivo
mp4 com as detecções ao invés de fazê-la
# simultaneamente

codec = cv2.VideoWriter_fourcc(*'XVID')
vid_out = cv2.VideoWriter('result.mp4', codec, 30, (1280, 720))

file = open("detec.csv", 'w', newline='', encoding= 'utf-8')
writer = csv.writer(file)
writer.writerow(['Frame', 'Objeto', 'Score', 'x_pos', 'y_pos'])

while cap.isOpened():

    _, f = cap.read()
    start = time.time()
```

```
    classes, scores, boxes = model.detect(f,0.1,0.2)
    end = time.time()

    for (classid, score, box) in zip(classes, scores, boxes):
        color = name_color[classid]
        label = f"{class_names[classid]}: {score}"
        writer.writerow([cap.get(cv2.CAP_PROP_POS_MSEC),
class_names[classid], score, (box[0] + int(box[2]/2)),
                         (box[1] + int(box[3]/2))])

        cv2.rectangle(f, box, color, 2)
        cv2.putText(f, label, (box[0], box[1] - 10),
cv2.FONT_HERSHEY_COMPLEX, 0.5, color, 2)

    fps_label = f"FPS: {round((1/(end - start)),2)}"

    cv2.putText(f, fps_label, (0,25), cv2.FONT_HERSHEY_COMPLEX, 1,
(0,0,0), 5)
    cv2.putText(f, fps_label, (0,25), cv2.FONT_HERSHEY_COMPLEX, 1,
(0,255,0), 3)

    vid_out.write(f)
    cv2.imshow("Video Detection",f)

    if cv2.waitKey(1) == 27 or not _:
        break

file.close()
vid_out.release()
cap.release()
cv2.destroyAllWindows()
```