پروژه ی ماشین لرنینگ

دانشگاه خوارزمی

تهیه کننده: لعیا اعتصام

استاد: دکتر کیوان برنا

زمستان ۱۴۰۲

چکیده:

در زندگی امروزه، با توجه به اهمیت امنیت در جامعه و وجود ابزارهای متنوع مخرب، نیاز به تشخیص این ابزارها بیش از پیش حس میشود. مسئله ی اهمیت این تشخیص در مکانهای عمومی، مکانهای زیارتی و ادارات دولتی بیشتر است. در این پروژه قصد داریم یک مدل تشخیص شی بسازیم که ابزارهای مخرب را شناسایی کند.

مقدمه:

در این پروژه میخواهیم یک مدل تشخیص شی با استفاده از کتابخانه ی تنسورفلو پایتون بسازیم. «تنسورفلو» (TensorFlow)، یک کتابخانه رایگان و «متن‌باز» (Open Source) برای «برنامه‌نویسی جریان داده» (Dataflow Programming) و «برنامه‌نویسی متمایزگر» (Differentiable Programming)، جهت انجام طیف وسیعی از وظایف است. تنسورفلو، کتابخانه‌ای برای «ریاضیات نمادین» (Symbolic Math) محسوب می شود و کاربردهای گوناگونی در «یادگیری ماشین» (Machine Learning) دارد که از آن جمله می‌توان به پیاده‌سازی «شبکه‌های عصبی» (Neural Networks) اشاره کرد. لازم به ذکر است که تا نسخه ی ۳,۱۰ پایتون را پشتیبانی میکند.

یکی دیگر از کتابخانه های مورد استفاده در این کد، کتابخانه ی opencv است. اوپن‌سی‌وی (به انگلیسی: OpenCV) یا همان Open Computer Vision Library مجموعه ای از کتابخانه‌های برنامه‌نویسی پردازش تصویر و یادگیری ماشین است. این مجموعه بیشتر بر پردازش تصویر بی درنگ (به انگلیسی: Real Time) تمرکز دارد.

سناریو:

Subject: a security company wants to develop an object detection model that can detect and locate suspicious object in security camera footage. The model should be able to identify objects such as weapons, bags and suspicious packages. The company plans to use the model to enhance its security systems and prevent potential security threats.

جزئیات:

این پروژه را با استفاده از تصاویر اشیا مخرب و الگوریتم های یادگیری عمیق اموزش می دهیم. دیتاست به صورت دستی اماده شده و از هیچ سایتی برگرفته نشده است. البته میتوان برای دیدن نتایج واقعی تر به ان دیتاست تزریق کرد. پس از اموزش، با استفاده از این مدل اشیا مخرب موجود در فیلم دوربین های امنیتی را به صورت live تشخیص می دهیم.

LabelImg، ابزار محبوب حاشیه نویسی تصویر که توسط Tzutalin با کمک ده ها مشارکت کننده ایجاد شده است، دیگر به طور فعال توسعه نمی یابد و به بخشی از جامعه Label Studio تبدیل شده است. Label Studio، انعطاف‌پذیرترین ابزار برچسب‌گذاری داده‌های منبع باز برای تصاویر، متن، فرامتن، صدا، ویدئو و داده‌های سری زمانی را بررسی کنید. LabelImg یک ابزار حاشیه نویسی تصویر گرافیکی است. به زبان پایتون نوشته شده است و از Qt برای رابط گرافیکی خود استفاده می کند. حاشیه نویسی ها به عنوان فایل های XML در قالب PASCAL VOC، فرمتی که توسط ImageNet استفاده می شود، ذخیره می شوند. علاوه بر این، از فرمت های YOLO و CreateML نیز پشتیبانی می کند.

مدل استفاده شده در این پروژه Tensorflow model Garden است. TensorFlow Model Garden یک مخزن با تعدادی از پیاده سازی های مختلف از مدل های پیشرفته (SOTA) و راه حل های مدل سازی برای کاربران TensorFlow است که به طور رسمی با آخرین API های TensorFlow 2 توسط TensorFlow نگهداری، پشتیبانی و به روز نگه داشته شده است و به طور منطقی برای عملکرد سریع بهینه شده است در حالی که هنوز خواندن آن آسان میباشد و از اجرا بر روی انواع دستگاه های مختلف (CPU، GPU و TPU) پشتیبانی می کند.

Ssd mobnet: معماری SSD (Single Shot Detection) که برای تشخیص اشیا استفاده می شود و یک خط لوله (pipeline) تشخیص شی را در یک ویدیو اعمال میکند.

- روش کار:

ابتدا با دستور python در CMD چک میکنیم که ورژن پایتون از ۳,۱۰ بیشتر نباشد.

با نوشتن دستور py -m venv my-env یک محیط مجازی ایجاد میکنیم و آن را فعال میکنیم.

با دستور زیر kernel را فعال میکنیم.

Pip install ipykernel

Py –m ipykernel install - -user - -name=my_kernel

سپس پروژه را در jupyter notebook اجرا میکنیم.

- Open 1. Image Collection.ipynb

در این قسمت از پروژه، عکس های مورد نیاز پروژه اعم از کیف، اسلحه، چاقو و... را به پروژه تزریق میکنیم.

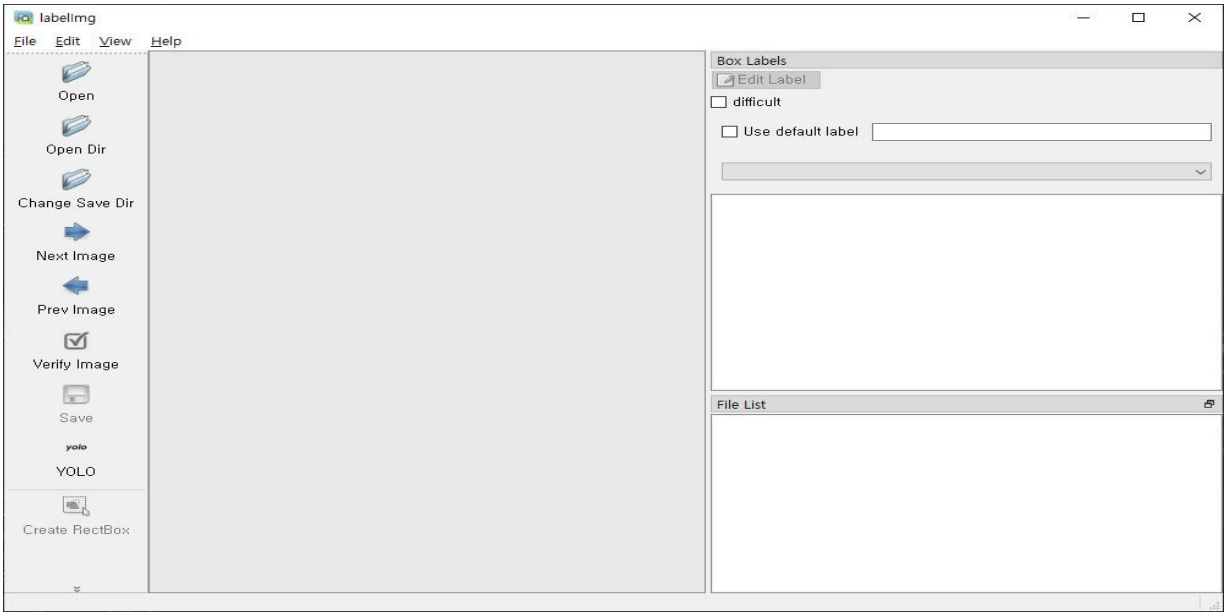
برای این کار دو راه وجود دارد: ۱. ایتدا عکس هارا به صورت دستی وارد فولدر های موجود در collectedimages کنیم ۲. روش دیگر این است که با استفاده از وبکم (clause 4) عکس های مورد نیاز را نشان دهیم. (ما در این پروژه از روش اول استفاده کردیم)

```
for label in labels:
    cap = cv2.VideoCapture(0)
    print('Collecting images for {}'.format(label))
    time.sleep(5)
    for imgnum in range(number_imgs):
        print('Collecting image {}'.format(imgnum))
        ret, frame = cap.read()
        imgname = os.path.join(IMAGES_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1()))) #save images in the directory
        cv2.imwrite(imgname, frame)
        cv2.imshow('frame', frame)
        time.sleep(2)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
cap.release() #exit webcam
cv2.destroyAllWindows()  #exit all windows
#open    tensorflow -> workspace -> images -> collected images
```

شکل.۱

پس از طی این مراحل عکس ها را در labelimg باز میکنیم و سوژه ی مورد نظر را مشخص کرده و سیو میکنیم. توجه شود که حالت yolo را به pascal تغییر میدهیم.



شکل.۲ محیط labelImg

تصاویر جمع آوری شده را به صورت دستی به دو پوشه تقسیم کرده و تست کنید. بنابراین اکنون همه پوشه ها و حاشیه نویسی ها باید بین دو پوشه زیر تقسیم شوند.

\Tensorflow\workspace\images\train

Tensorflow\workspace\images\test

- Open 2. Training and Detection.ipynb

Wget که در این پروژه استفاده شده است برای دانلود اسان فایل ها از اینترنت است.

```
In [7]: #for download file from the internet easily
        if os.name=='nt':
            !pip install wget
            import wget
```

شکل.۳

بافرهای پروتکل (با نام مستعار، پروتوباف) مکانیسم Google برای سریال‌سازی داده‌های ساخت‌یافته است.

```
In [9]: # Install Tensorflow Object Detection
if os.name=='posix':
    !apt-get install protobuf-compiler
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && cp object_detection/packages/tf2/s

if os.name=='nt': #download protoc buffers for storing data
    url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
    os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && copy object_detection\\packages\\t
    !cd Tensorflow/models/research/slim && pip install -e .  #install from a local project path
```

شکل.۴

پس از اجرای قسمت ۶ و ۷ میتوانیم با استفاده از تنسوربورد، میزان loss و accuracy را در گراف میبینیم.

In CMD:

Cd Tensorflow\workspace\models\my_ssd_mobnet\train

And cd Tensorflow\workspace\models\my_ssd_mobnet\eval

Run tensorboard –logdir=.

با استفاده از قسمت ۹ میتوانیم وبکم را باز کنیم و به صورت Real Time وسایل مخرب را تشخیص دهیم.

```
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

while cap.isOpened():
    ret, frame = cap.read()
    image_np = np.array(frame)

    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
    detections = detect_fn(input_tensor)

    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
    detections['num_detections'] = num_detections

    # detection_classes should be ints.
    detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

    label_id_offset = 1
    image_np_with_detections = image_np.copy()

    viz_utils.visualize_boxes_and_labels_on_image_array(
                image_np_with_detections,
                detections['detection_boxes'],
                detections['detection_classes']+label_id_offset,
                detections['detection_scores'],
                category_index,
                use_normalized_coordinates=True,
                max_boxes_to_draw=5,
                min_score_thresh=.8,
                agnostic_mode=False)

    cv2.imshow('object detection',  cv2.resize(image_np_with_detections, (800, 600)))

    if cv2.waitKey(10) & 0xFF == ord('q'):
        cap.release()
```

شکل.۵

نتیجه:

دانستیم که با استفاده از تنسورفلو و مدل های آن میتوانیم یک مدل تشخیص شی بسازیم که تقریبا با توجه به دیتاست محدودی که داریم دقت بالایی دارد. (loss حدودا ۱۸ درصد است).

چالش ها:

در روند کد زدن به چالش هایی برخورد کردم که تعدادی از آنها را به اختصار بیان میکنم.

- یکی از بزرگترین چالش های پیش رو، عدم هماهنگ بودن تنسورفلو با اخرین نسخه ی پایتون بود.
- اخرین ورژن های تنسورفلو از جی پی یو پشتیبانی نمیکند که مجور به ران کردن کد در سی پی یو بودم و زمان زیادی صرف ران میشد.
- با توجه به دیتاست محدودی که داشتم انتخاب عکس ها بسیار مهم بود زیرا ممکن است میزان دقت به واقعیت نزدیک نباشد.

مراجع:

https://github.com/nicknochnack/TFODCourse/tree/main

https://github.com/HumanSignal/labelImg

https://www.youtube.com/watch?v=yqkISICHH-U

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjh4cCVyJSEAxUe1QIHHSOBD0QQFnoECAYQAQ&url=https%3A%2F%2Fwww.tensorflow.org%2F&usg=AOvVaw0TGZBeXHx2CVPI2FiDZclR&opi=89978449

https://github.com/tensorflow/models

https://github.com/protocolbuffers/protobuf

https://github.com/abhileshborode/SSD-MobileNet

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

Real-Time Object Detection Based on UAV Remote Sensing: A Systematic Literature Review(٠١-١٠-٢٠٢٣)

Small-Object Detection for UAV-Based Images Using a Distance Metric Method (٠١-١٠-٢٠٢٢)

Real-time multi-camera video analytics system on(٠١-٠٣-٢٠١٦)