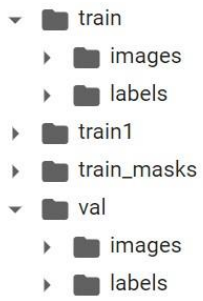
سوال اول

در ابتدا با استفاده از gdown فایل را دانلود و بعد unzip میکنیم.و بعد همه فایل های عکس را در دارکتوری trian1 و همه فایل های mask را در train_masks میریزیم.

```
# بررسی فایل‌های موجود در هر فولدر و کپی کردن آنها در فولدر مقصد
for folder in subfolders:
    for root, dirs, files in os.walk(folder):
        for file in files:
            source_file_path = os.path.join(root, file)
            file_name = os.path.basename(source_file_path)
            folder_name=os.path.basename(folder)
            img = cv2.imread(source_file_path)
            if "label" in file_name:
              s=file_name.split('.')
              finalname=s[0]+"_"+folder_name+'.png'
              cv2.imwrite(os.path.join(label_root, finalname),img)
            else:
              s=file_name.split('.')
              finalname=s[0]+"_"+folder_name+'.png'
              cv2.imwrite(os.path.join(image_root, finalname),img)
```

بعد داده ها را به دو قسمت train,val تقسیم می کنیم من به این صورت در نظر گرفتم در هر دو پوشه یعنی train,val دو پوشه دیگر داریم به نام images,labels در واقع شکل فولدرهای ما به صورت زیر است هم چنین ۲۰ درصد داده ها را به val اختصاص دادم:

```
├── yolov7
##  └── train
####└── images (folder including all training images)
####└── labels (folder including all training labels)
##  └── test
####└── images (folder including all testing images)
####└── labels (folder including all testing labels)
```



یک تابع به نام writeintxt داریم که در ابتدا یک عکس و مسیر فایل تکست را میگیرد روی عکس canny میزند و بعد کانتور های ان را به دست می اورد و بعد x,y,w,h مستطیل برای هر کانتور را به دست می اوریم و بعد نرمال می کنیم و در فایل تکست مینویسیم کد ان به صورت زیر است:

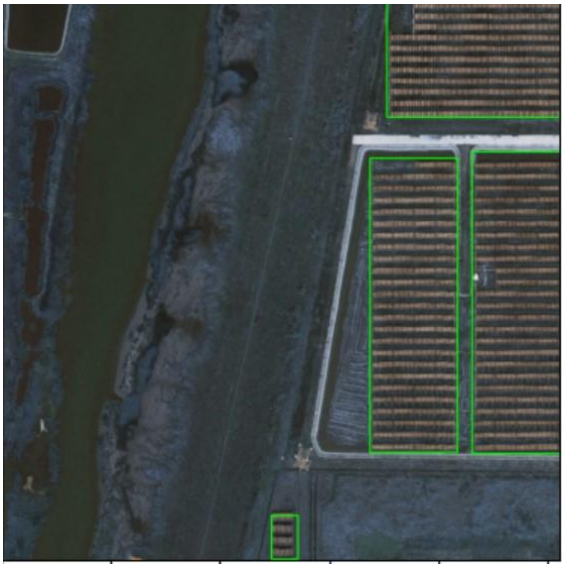
```python
def writeintxt(img,txt):
  m1gray=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  image_height,image_width=m1gray.shape
  edgesimg = cv2.Canny(m1gray,10,150)
  # Find contours in the binary image
  contours, hierarchy = cv2.findContours(edgesimg, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
  # Draw rectangles on the image for each contour
  id=0
  with open(txt, 'w') as f:
    for contour in contours:
      # Find the bounding rectangle for the contour
      x, y, w, h = cv2.boundingRect(contour)
      center_x = (x + w / 2) / image_width
      center_y = (y + h / 2) / image_height
      norm_width = w / image_width
      norm_height = h / image_height
      print(center_x,center_y,norm_width,norm_height)
      f.write('{} {} {} {} {}\n'.format(id,center_x, center_y, norm_width, norm_height))
      id+=1
  f.close()
    # Draw the rectangle on the image
```

یک نمونه فایل تکست هم به صورت زیر است:

100_0.txt 0_2.txt 0_0.txt × •••

```
1 0 0.96435546875 0.67919921875 0.0712890625 0.3115234375
2 0 0.8388671875 0.6328125 0.09765625 0.419921875
3 0 0.8388671875 0.6328125 0.09765625 0.419921875
4 0 0.98681640625 0.25634765625 0.0263671875 0.2314453125
5 0 0.908203125 0.03515625 0.18359375 0.0703125
6
```

در ادامه تابع plot_bbox_on_img که ۱۰ تصویر را باکس های مستطیلی را بر روی پنل های خورشیدی رسم میکند.



فایل را از صفحه گیت هاب کلون کردم و بعد requirememnts ها را نصب کردم و در ادامه فایل custom.yaml را درون پوشه data ایجاد میکنم و متن زیر را در ان قرار می دهم:

custom.yaml

```
train: /content/train
val: /content/val

nc: 1
names: ['object']
```

و بعد مدل را با استفاده از command زیر train می کنیم:

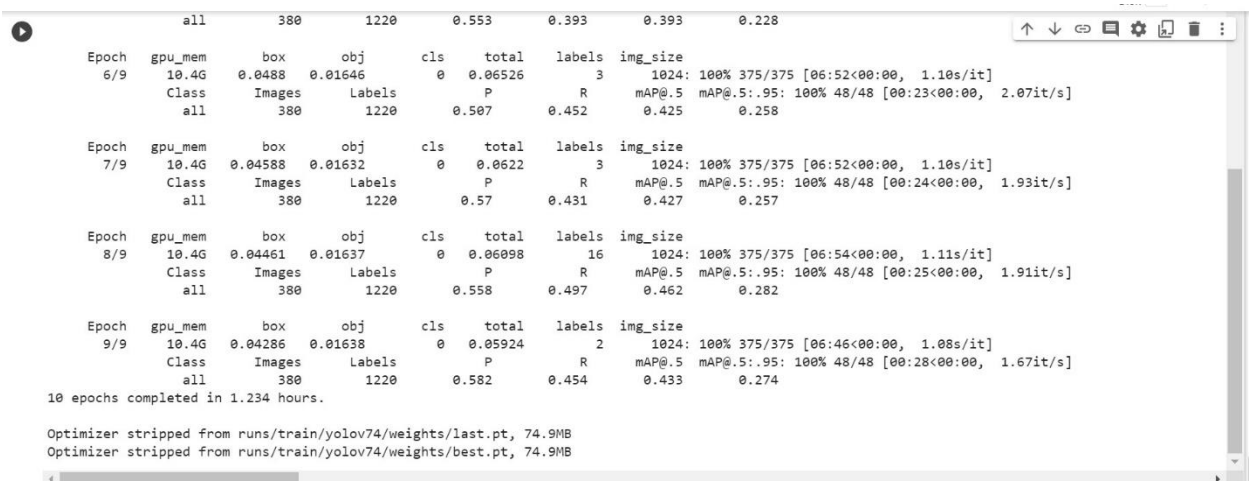
```
!python3 train.py --weights yolov7.pt --data "data/custom.yaml" --workers 4 --batch-size 4 --img 1024 --cfg cfg/training/yolov7.yaml --name yolov7 --hyp data/hyp.scratch.p5.yaml --epochs 50
```

```
Image sizes 416 train, 416 test
Using 2 dataloader workers
Logging results to runs/train/yolov73
Starting training for 50 epochs...

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      0/49    0.908G   0.06931   0.01287         0   0.08218         8       416: 100% 375/375 [05:45<00:00,
               Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 48/48 [00:
                 all         380        1220       0.228       0.289       0.136      0.0463

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
      1/49     1.91G    0.0569   0.01095         0   0.06785        26       416:   2% 9/375 [00:05<05:08,  1
```

ولی در نهایت به علت طولانی بودن زمان train تا ۱۰ تا epoch رفتم که نتیجه به صورت زیر شد:



```
               all        380       1220      0.553      0.393      0.393      0.228

      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
        6/9     10.4G   0.04488   0.01646         0   0.06526         3      1024: 100% 375/375 [06:52<00:00,  1.10s/it]
                Class      Images      Labels          P          R      mAP@.5  mAP@.5:.95: 100% 48/48 [00:23<00:00,  2.07it/s]
                  all        380       1220      0.507      0.452      0.425      0.258

      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
        7/9     10.4G   0.04588   0.01632         0    0.0622         3      1024: 100% 375/375 [06:52<00:00,  1.10s/it]
                Class      Images      Labels          P          R      mAP@.5  mAP@.5:.95: 100% 48/48 [00:24<00:00,  1.93it/s]
                  all        380       1220       0.57      0.431      0.427      0.257

      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
        8/9     10.4G   0.04461   0.01637         0   0.06098         16      1024: 100% 375/375 [06:54<00:00,  1.11s/it]
                Class      Images      Labels          P          R      mAP@.5  mAP@.5:.95: 100% 48/48 [00:25<00:00,  1.91it/s]
                  all        380       1220      0.558      0.497      0.462      0.282

      Epoch   gpu_mem       box       obj       cls     total    labels  img_size
        9/9     10.4G   0.04286   0.01638         0   0.05924         2      1024: 100% 375/375 [06:46<00:00,  1.08s/it]
                Class      Images      Labels          P          R      mAP@.5  mAP@.5:.95: 100% 48/48 [00:28<00:00,  1.67it/s]
                  all        380       1220      0.582      0.454      0.433      0.274
10 epochs completed in 1.234 hours.

Optimizer stripped from runs/train/yolov74/weights/last.pt, 74.9MB
Optimizer stripped from runs/train/yolov74/weights/best.pt, 74.9MB
```

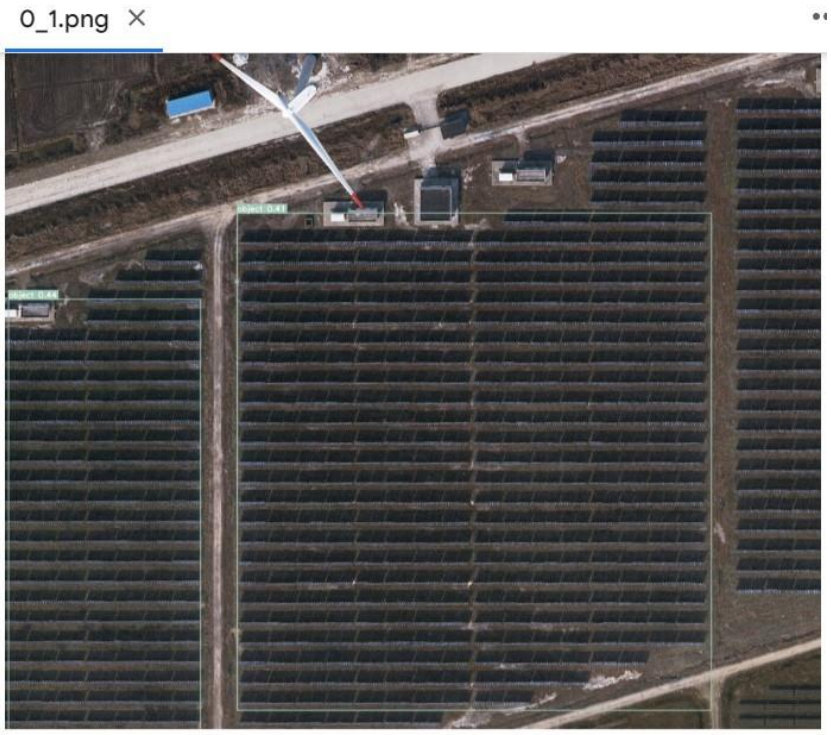
و بعد برای Inference روی یک تصویر تست کردم و نتیجه ان به صورت زیر شد:

```
/content/yolov7
Namespace(weights=['/content/yolov7/runs/train/yolov74/weights/best.pt'], source='/content/train/images/0_1.png
YOLOR 🚀 v0.1-126-g84932d7 torch 2.0.1+cu118 CUDA:0 (Tesla T4, 15101.8125MB)

Fusing layers...
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
IDetect.fuse
/usr/local/lib/python3.10/dist-packages/torch/functional.py:504: UserWarning: torch.meshgrid: in an upcoming re
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
Model Summary: 314 layers, 36481772 parameters, 6194944 gradients, 103.2 GFLOPS
 Convert model to Traced-model...
 traced_script_module saved!
 model is traced!

2 objects, Done. (56.1ms) Inference, (162.2ms) NMS
 The image with the result is saved in: runs/detect/exp4/0_1.png
Done. (0.490s)
```

که خروجی تصویر به صورت زیر شد:



طبعا چون تعداد اپک ها کم بود نتونست همه رو تشخیص بده.

سوال دوم

الف) فیلتر کالمن (Kalman Filter) یک روش ترکیبی است که برای تخمین حالت یک سیستم کنترل خطی متغیر در زمان (LTV) به کار می‌رود. این روش از اطلاعات مشاهده شده در گذشته و مدل دینامیک سیستم برای تقریب حالت فعلی سیستم استفاده می‌کند. فیلتر کالمن به طور خلاصه متشکل از دو مرحله زیر است:

۱. مرحله پیش‌بینی (Prediction): در این مرحله، حالت بعدی سیستم بر اساس حالت کنونی و مدل دینامیک سیستم پیش‌بینی می‌شود.
۲. مرحله به‌روزرسانی (Update): در این مرحله، تخمین حالت بعدی بر اساس مشاهدات جدید به‌روزرسانی می‌شود.

برای پیش‌بینی یک پارامتر با استفاده از فیلتر کالمن، ابتدا باید مدل دینامیک سیستم را بیان کنیم. برای سادگی، فرض کنید که داریم یک متغیر خطی در زمان را پیش‌بینی می‌کنیم:

$$x_k = A x_{k-1} + w_k$$

که در اینجا x_k حالت سیستم در زمان k، A ماتریس تبدیل حالت و w_k نویز فرآیند است. همچنین، داریم مشاهده‌ی z_k که به صورت زیر با x_k ارتباط دارد:

$$z_k = H x_k + v_k$$

که در اینجا Hماتریس تبدیل مشاهده و v_kنویز مشاهده است.حال برای تخمین پارامتر مورد نظر با استفاده از فیلتر کالمن، مراحل زیر را طی می‌کنیم:

مرحله پیش‌بینی:

$$x_{k|k-1} = A x_{k-1|k-1}$$

$$P_{k|k-1} = A P_{k-1|k-1} A^T + Q$$

که در اینجا x_k|k-1تخمین حالت x_kبر اساس اطلاعات تا زمان k-1و P_k|k-1کوواریانس خطای تخمین است Q نیز ماتریس کوواریانس نویز فرآیند است.

مرحله به‌روزرسانی:

$$K_k = P_{k|k-1} H^T (H P_{k|k-1} H^T + R)^{-1}$$

$$x_{k|k} = x_{k|k-1} + K_k (z_k - H x_{k|k-1})$$

$$P_{k|k} = (I - K_k H) P_{k|k-1}$$

که در اینجا K_kبردار کالمن گین(Kalman Gain) ، Rماتریس کوواریانس نویز مشاهده و Iماتریس همانی است.با انجام این دو مرحله به صورت تکراری، فیلتر کالمن به طور خودکار به یک تخمین بهینه برای پارامتر مورد نظر می‌رسد. در هر مرحله‌ی پیش‌بینی، فیلتر کالمن حالت بعدی سیستم را بر اساس مدل دینامیک سیستم و حالت کنونی سیستم پیش‌بینی می‌کند. سپس در مرحله‌ی به‌روزرسانی، فیلتر کالمن حالت بعدی را بر اساس مشاهدات جدید به‌روزرسانی می‌کند. این روند به صورت تکراری ادامه می‌یابد و هر بار تخمینی بهتربرای پارامتر مورد نظر به دست می‌آید.

ب) برای استفاده از فیلتر Kalman برای پیش‌بینی حرکت لیبل‌ها با ۴ ویژگی مختلف در یک شبکه با ۷ گره، باید ابتدا یک ماتریس پیش‌بینی Kalman و یک ماتریس اندازه‌گیری تعریف کنیم.ماتریس پیش‌بینی Kalman برای این حالت به شکل زیر خواهد بود:

[1 0 dt 0 0 0 0]

[0 1 0 dt 0 0 0]

[0 0 1 0 0 0 0]

[0 0 0 1 0 0 0]

[0 0 0 0 1 0 0]

[0 0 0 0 0 1 0]

[0 0 0 0 0 0 1]

در این ماتریس، هر سطر متعلق به یک لیبل خاص است و هر ستون متناظر با یکی از ویژگی‌های مختلف است. ستون‌های ۱ تا ۴ مربوط به موقعیت x و y و سرعت vx و vy هستند. ستون‌های ۵ تا ۷ هم مربوط به ویژگی‌های دیگر هستند (مثلاً اندازه لیبل).

[۰ ۰ ۰ ۰ ۰ ۰ ۱]

[۰ ۰ ۰ ۰ ۰ ۱ ۰]

[۰ ۰ ۰ ۰ ۱ ۰ ۰]

[۰ ۰ ۰ ۱ ۰ ۰ ۰]

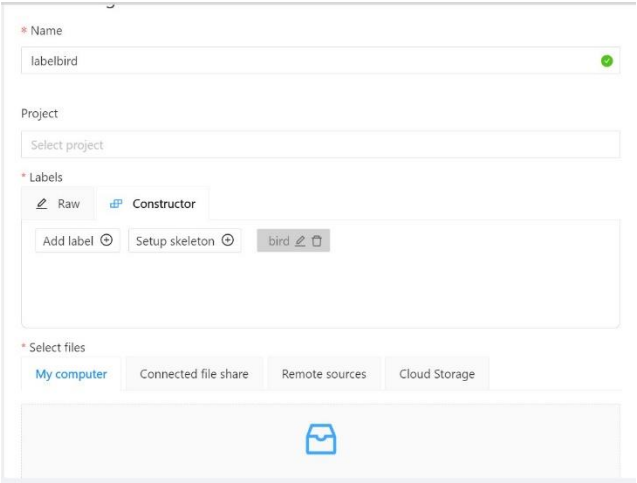
در اینجا، هر سطر نیز متعلق به یک لیبل خاص است و ستون‌های ۱ تا ۴ با ویژگی‌های موقعیت و سرعت مرتبط هستند، در حالی که ستون‌های ۵ تا ۷ برای ویژگی‌های دیگر استفاده می‌شوند.با استفاده از این دو ماتریس و داده‌های مشاهده شده، می‌توانیم فیلتر Kalman را به شکل ماتریسی برای پیش‌بینی حرکت لیبل‌ها در یک شبکه با ۷ گره اعمال کنیم. این عملیات در دو مراحل مختلفی از جمله پیش‌بینی، تصحیح و به‌روزرسانی انجام می‌شود. در هر مرحله، ماتریس‌های جدیدی برای پیش‌بینی و تصحیح حالت لیبل‌ها محاسبه می‌شوند و با استفاده از آن‌ها، حالت لیبل‌ها در زمان جدید تخمین زده می‌شود. با ادامه این عملیات در هر فریم، می‌توانیم حرکت لیبل‌ها را در زمان واقعی پیش‌بینی کنیم.

ج) الگوریتم SORT یک روش ساده برای پیش‌بینی موقعیت اشیاء در فریم‌های بعدی در ویدیوها است. این الگوریتم برای تشخیص و ردیابی اشیاء در ویدیوهای با سرعت بالا و دارای چالش بالایی مانند تلاش در محیط‌های پرتلاطم و با تغییرات شدید نورپردازی به کار می‌رود.الگوریتم DeepSORT که توسعه‌یافته‌ی الگوریتم SORT است، در عین سادگی و انعطاف‌پذیری الگوریتم SORT، از شبکه‌های عصبی عمیق برای استخراج ویژگی‌های قوی‌تر برای اشیاء استفاده می‌کند. از این رو، این الگوریتم قادر است تا دقت و سرعت بیشتری در ردیابی اشیاء داشته باشد.شبکه‌های عصبی عمیق مانند شبکه‌های CNN با استفاده از لایه‌های مختلف، ویژگی‌های متفاوتی را از تصاویر استخراج می‌کنند. این ویژگی‌ها به عنوان ورودی به الگوریتم DeepSORT داده می‌شوند تا الگوریتم

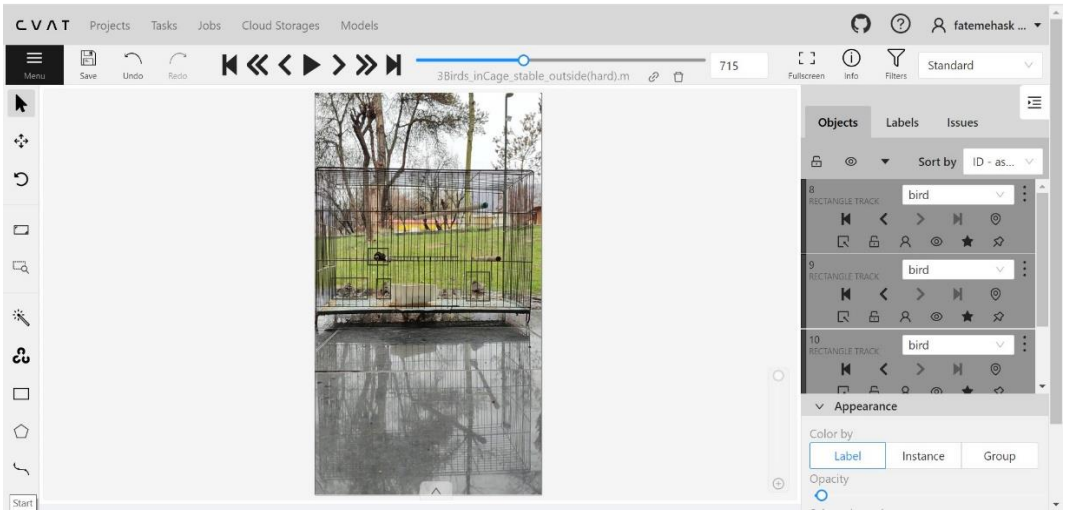
بتواند موقعیت و گره‌های مختلف اشیاء را در فریم‌های بعدی ردیابی کند. استفاده از شبکه‌های عمیق در الگوریتم DeepSORT باعث می‌شود تا الگوریتم قادر به تشخیص و ردیابی اشیاء با دقت بالاتری باشد و در مواجه با حالات پیچیده‌تری مانند اشیاء با حرکت سریع و تغییرات شدید نورپردازی، دقت بیشتری را ارائه دهد.

سوال سوم

الف)در ابتدا task خود را در سایت cvat مطابق شکل زیر تعریف می کنیم:



و در ادامه پرندگان را از فریم ۷۰۰ تا فریم ۱۴۰۰ برچسب گذاری می کنیم:



در فولدر job_216955-2023_07_05_12_19_19-yolo 1.1 مجموعه برچسب ها موجود است.

ب) در این صورت، مدلی که بر روی داده‌هایی آموزش دیده است که پرندگان در محیط‌های آزاد هستند، ممکن است برای تشخیص پرندگان در قفس‌ها به خوبی عمل نکند. دلیل این امر این است که تفاوت بین داده‌هایی که

پرندگان در محیط‌های آزاد و داده‌هایی که پرندگان در قفس هستند، بسیار مهم است.در داده‌هایی که پرندگان در قفس هستند، شرایط نورپردازی، پس‌زمینه و اندازه پرندگان با داده‌هایی که در محیط‌های آزاد به تصویر کشیده شده‌اند، متفاوت است. بنابراین، آموزش مدل بر روی داده‌هایی که پرندگان در محیط‌های آزاد هستند، به تنهایی کافی نیست

ج) یکی از راهکارهایی که می‌توان برای آموزش مدل بر روی داده‌هایی استفاده کرد که شامل پرندگان در قفس هستند، استفاده از روش Transfer Learning است. در این روش، از یک مدل پیش آموزش دیده برای تشخیص اشیاء در مجموعه داده‌هایی مانند COCO استفاده می‌شود و سپس این مدل با داده‌هایی که پرندگان در قفس به تصویر کشیده شده‌اند، تنظیم می‌شود.برای استفاده از این روش، می‌توانید با استفاده از یک مدل پیش آموزش دیده مانند مدل‌هایی مثل ResNet، VGG، Inception و ...، یک مدل جدید برای تشخیص پرندگان در قفس آموزش دهید. برای انجام این کار، می‌توانید از داده‌هایی استفاده کنید که پرندگان در قفس به تصویر کشیده شده‌اند و با استفاده از روش Transfer Learning، مدل را با داده‌های جدید آموزش دهید. در این روش، مدل پیش آموزش دیده، به عنوان یک استخراج‌کننده ویژگی برای داده‌های جدید استفاده می‌شود و سپس لایه‌های دیگر به آن اضافه می‌شوند و مدل برای تشخیص پرندگان در قفس تنظیم می‌شود.با این روش، نیازی به برچسب زنی داده‌هایی که پرندگان در قفس هستند، نیست و می‌توانید از داده‌هایی که پرندگان در محیط‌های آزاد به تصویر کشیده شده‌اند و همچنین داده‌های شامل پرندگان در قفس در آموزش مدل استفاده کنید. همچنین، با استفاده از مدل پیش آموزش دیده، می‌توانید دقت و عملکرد مدل را بهبود ببخشید و به سرعت به مدلی با دقت بالا برای تشخیص پرندگان در قفس برسید.

سوال چهارم

الف) معماری شبکه SiamFC (یا Siamese Fully Convolutional Network) یکی از روش‌های پرکاربرد در وظایف بینایی کامپیوتر مانند تشخیص و ردیابی اشیاء در ویدیوها است. این شبکه با استفاده از یک شبکه عصبی عمیق با ساختار کاملاً پیچشی، امکان تشخیص و ردیابی اشیاء در ویدیوها را با دقت بالا فراهم می‌کند. در ادامه به بررسی مزایا و محدودیت‌های این شبکه پرداخته می‌شود:

مزایا:

دارای دقت بالایی در تشخیص و ردیابی اشیاء در ویدیوها است.

این شبکه با استفاده از تکنیک‌های پیچیده مانند آموزش دوتایی (pairwise training)، قادر است تا با دقت بالایی اشیاء را در بین فریم‌های مختلف ویدیوها ردیابی کند.

شبکه SiamFC با ساختار کاملاً پیچشی، دارای سرعت بالایی در پردازش تصاویر است و از این رو، برای پردازش ویدیوهای بلند نیز قابل استفاده است.

این شبکه علاوه بر تشخیص و ردیابی اشیاء، برای وظایف دیگری مانند تطبیق تصاویر نیز مورد استفاده قرار می‌گیرد.

محدودیت‌ها:

این شبکه تنها برای تشخیص و ردیابی اشیاء به کار می‌رود و قابلیت استفاده در وظایف دیگری مانند تشخیص چهره وجود ندارد.

شبکه SiamFC در مواجه با تصاویر با شرایط نورپردازی پیچیده و تغییرات شدید در محیط، دچار مشکل می‌شود و دقت پایین‌تری در تشخیص و ردیابی اشیاء ارائه می‌دهد.

این شبکه نیاز به داده‌های بسیار زیاد و آموزش دوتایی دارد تا بتواند دقت بالایی در تشخیص و ردیابی اشیاء ارائه دهد.

ب) توضیح اجزای اصلی این شبکه به صورت زیر است:

۱. معماری شبکه:

شبکه سیامی شامل دو شاخه مجزا است که هر کدام از آن‌ها یک شبکه عصبی عمیق با ساختار کاملاً پیچشی است. این دو شبکه به صورت مشابه با هم آموزش داده می‌شوند و هدف آن‌ها یادگیری ویژگی‌های مشترک بین اشیاء موجود در دو فریم متوالی ویدئو است.

۲. ورودی شبکه:

ورودی شبکه شامل دو تصویر از دو فریم متوالی ویدئو است که به عنوان تصاویر ورودی برای دو شاخه مجزای شبکه سیامی استفاده می‌شود.

۳. پیش‌پردازش تصاویر:

تصاویر ورودی به شبکه پس از پیش‌پردازش‌هایی مانند تبدیل به سیاه و سفید و استفاده از فیلترهای گوسی، به شکلی تمیز و آماده برای ورود به شبکه تبدیل می‌شوند.

۴. استخراج ویژگی:

هر یک از دو شاخه شبکه، ویژگی‌های تصویر ورودی را با استفاده از لایه‌های پیچشی و لایه‌های تکاملی (convolutional layer) استخراج می‌کند. پس از استخراج ویژگی‌ها، یک لایه‌ی تماماً متصل (fully connected layer) نیز برای ترکیب ویژگی‌ها در هر شاخه استفاده می‌شود.

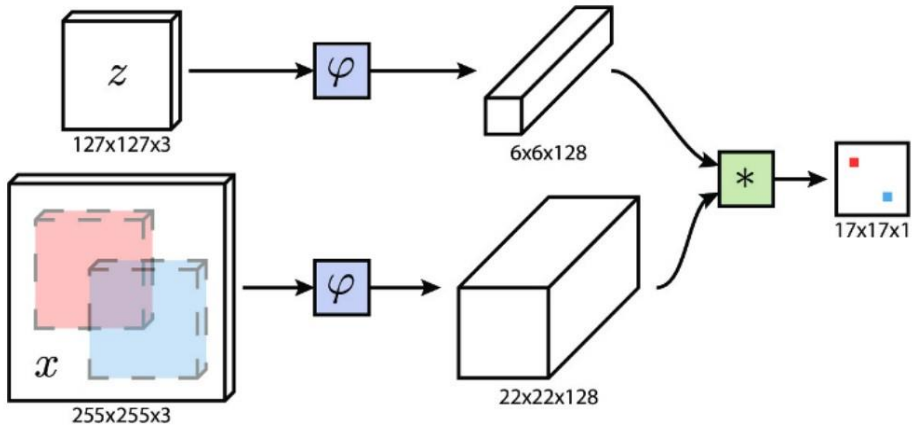
۵. محاسبه شباهت:

پس از استخراج ویژگی‌ها از دو تصویر متوالی، شباهت بین آن‌ها محاسبه می‌شود. برای این منظور از روش تولید شده در SiamFC استفاده می‌شود که ابتدا با استفاده از یک شبکه عصبی عمیق، ویژگی‌های تصاویر ورودی استخراج می‌شوند. سپس با استفاده از یک تابع شباهت، شباهت بین دو تصویر متوالی محاسبه می‌شود.

۶. ردیابی:

در نهایت، با استفاده از شباهت محاسبه شده در مرحله قبل، موقعیت اشی مورد نظر در فریم بعدی ویدئو پیش‌بینی می‌شود. برای این کار، از فیلتر کالمن استفاده می‌شود که با استفاده از اطلاعات موقعیت قبلی اشیا، موقعیت آن‌ها در فریم بعدی ویدئو را پیش‌بینی می‌کند.

به طور خلاصه، شبکه سیامی (SiamFC) با استفاده از دو شاخه مجزای شبکه، استخراج ویژگی از دو تصویر متوالی ویدئو، محاسبه شباهت بین آن‌ها و استفاده از فیلتر کالمن برای ردیابی اشیا، امکان ردیابی اشیا در ویدئوها با دقت بالا را فراهم می‌کند.



ج) ردیابی شی یکی از مسائل پیچیده در حوزه بینایی ماشین است و با چالش‌هایی مانند تغییرات نورپردازی، تغییر در اندازه و شکل شی، انسداد شی توسط سایر اشیاء و ... مواجه می‌شود. در ادامه به توضیح چالش‌های مرتبط با ردیابی شی و نحوه برطرف کردن آن‌ها توسط شبکه سیامی (SiamFC) پرداخته شده است:

۱. تغییر در اندازه و شکل شی:

یکی از چالش‌های مهم در ردیابی شی، تغییر در اندازه و شکل شی است که با تغییر در فاصله شی از دوربین، زاویه دید، وضوح تصویر و ... همراه است. برای رفع این چالش، شبکه سیامی با استفاده از ویژگی‌های ارزشمندی که از تصاویر استخراج می‌کند، می‌تواند شی را با دقت بالایی در تصاویر با اندازه و شکل مختلف، ردیابی کند.

۲. انسداد شی توسط سایر اشیاء:

در برخی موارد، شی مورد نظر توسط سایر اشیاء یا موانعی مانند دیوار، درخت، ماشین و ... انسداد می‌شود که باعث بروز مشکل در ردیابی شی می‌شود. برای رفع این چالش، شبکه سیامی با استفاده از ویژگی‌هایی که از تصاویر استخراج می‌شود، قادر به ردیابی شی در صورت انسداد قسمتی از آن است.شبکه سیامی (SiamFC) با استفاده از ویژگی‌های قدرتمندی که از تصاویر استخراج می‌کند، قادر به ردیابی شی با دقت بالا در مواجهه با چالش‌های مختلف در ردیابی شی است. همچنین با استفاده از روش آموزش دوتایی، شبکه سیامی قادر به آموزش با تعداد داده‌های کمتر است و امکان پرداختیاری به رویکرد Transfer Learning را نیز فراهم می‌کند. در این روش، شبکه از پیش آموزش داده شده در دامنه دیگری (مثلاً مجموعه داده‌های ImageNet) استفاده می‌کند و برای ردیابی شی در دامنه مورد نظر (مثلاً مجموعه داده‌های ردیابی شی)، با استفاده از انتقال یادگیری، وزن‌های شبکه از دامنه پیشین به دامنه جدید انتقال داده می‌شود. این روش باعث بهبود دقت و سرعت ردیابی شی شده و مشکلاتی مانند بروز بیشتر از Overfitting و نیاز به داده‌های آموزش بیشتر را کاهش می‌دهد.

د) معماری سیامی در واقع یک رویکرد برای مقایسه دو تصویر است. در این رویکرد، دو تصویر به دو شاخه مجزا از شبکه سیامی داده می‌شوند و ویژگی‌هایی که از هر تصویر استخراج می‌شوند، با یکدیگر مقایسه می‌شوند. این رویکرد در مسائلی که نیاز به مقایسه دو تصویر یا داده دارند، مانند ردیابی شی، تشخیص چهره، تشخیص اثر انگشت و ... مورد استفاده قرار می‌گیرد.به علاوه، معماری سیامی به عنوان یک روش آموزشی، به مسائل دیگری نیز می‌تواند اعمال شود. به عنوان مثال، در حوزه یادگیری با یک داده (One-Shot Learning)، معماری سیامی می‌تواند به عنوان یک روش کارآمد برای یادگیری با یک داده استفاده شود. در این رویکرد، یک تصویر از کلاس مورد نظر به عنوان تصویر اصلی و تصاویر دیگر از کلاس‌های مختلف به عنوان تصاویر مقایسه‌ای استفاده می‌شوند. با استفاده از شبکه سیامی، ویژگی‌های هر تصویر استخراج شده و با هم مقایسه شده و تصویری که بیشترین شباهت را با تصویر اصلی دارد، به عنوان پاسخ انتخاب می‌شود.علاوه بر این، معماری سیامی در حوزه تشخیص تقلب در کارت‌های اعتباری، تشخیص تقلب در شناسنامه‌ها، تشخیص تقلب در بیمه‌های مربوط به حوادث و ... نیز مورد استفاده قرار می‌گیرد. به عنوان مثال، در حوزه تشخیص تقلب در کارت‌های اعتباری، دو تصویر از کارت اعتباری که قرار است مقایسه شوند، به دو شاخه مجزا از شبکه سیامی داده می‌شوند. سپس ویژگی‌های هر تصویر

استخراج شده و با یکدیگر مقایسه می‌شوند تا در صورت وجود تقلب، شناسایی شود.بنابراین، معماری سیامی علاوه بر ردیابی شی، می‌تواند در حوزه‌های مختلفی از جمله یادگیری با یک داده، تشخیص تقلب و ... مورد استفاده قرار گیرد.