Person tracking and re-identification in video

Compre report

CS F376
DESIGN PROJECT
Under Prof. Vinayak Naik
BITS Pilani K.K. Birla Goa campus

By
NIRVAN ANJIRBAG
2016A3PS0711G
f20160711@goa.bits-pilani.ac.in

GitHub Link

1. Introduction

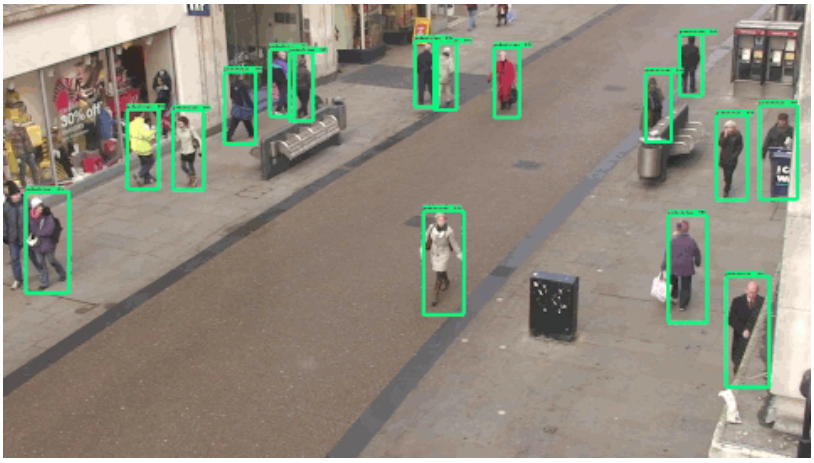
The objective of this project is the detection and tracking of people in a video to output a list of all unique people that have appeared in the video along with their individual cropped video sequences. The model must detect people and track them throughout the video. It must then count how many different people have appeared and store the identities in a list. If a person appears for some frames and then leaves the video for some time and then reappears later, the model should recognize it's the same person and not count this person a second time.

2. Application

The application of this project is the automated surveillance of cctv video footage. If we have a video footage from a CCTV camera and we want to know what people appeared in it, we have to watch the entire video manually. This is inconvenient as the video can be several hours or days long. The project automatically watches the video and outputs a list of what people appeared in the video. It also outputs a cropped video sequence for each person. These individual video sequences are useful if we want to a particular person’s actions. Hence, this project can automatically tell us what people appeared in a video and can show us their individual actions.

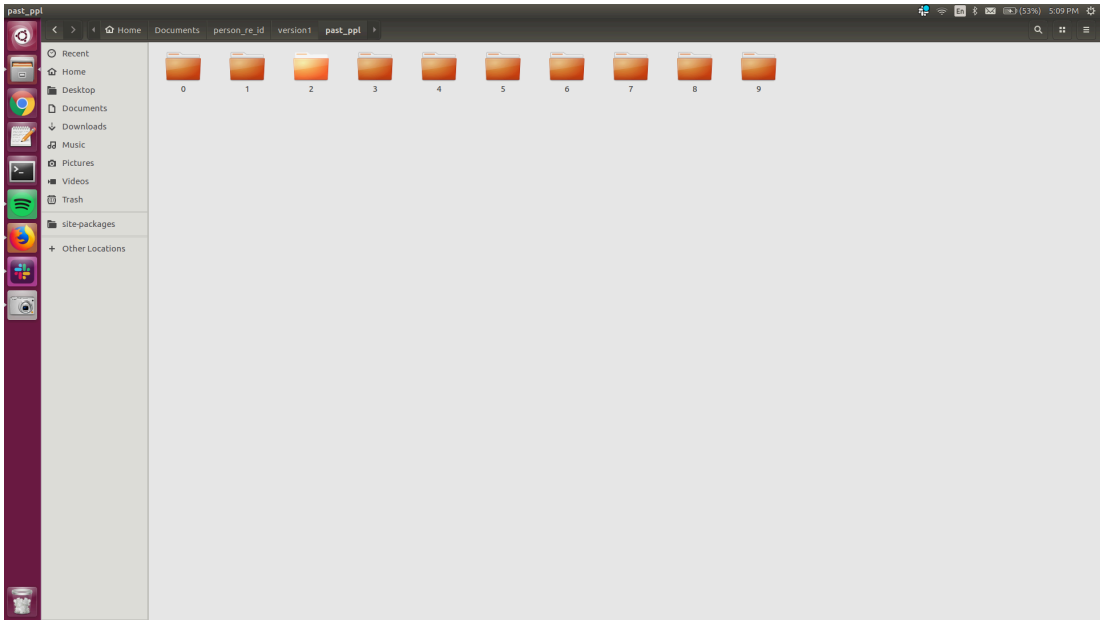
A CCTV video of a street or a room often contains multiple people. These people move around in the frame of the video in arbitrary paths. They may also cross paths and overlap with each other many times. They may also leave the video and reenter later. This must all be handled well.

This code will be connected to a live cctv video footage. It will observe the video in real-time and keep a track of how many people have appeared in the video until now and will store their identities in a list. This way, it will automatically keep a track of the people in the video and we can later simply check the generated list to see what people had appeared in the video. This way, a human doesn’t need to manually watch the entire video.

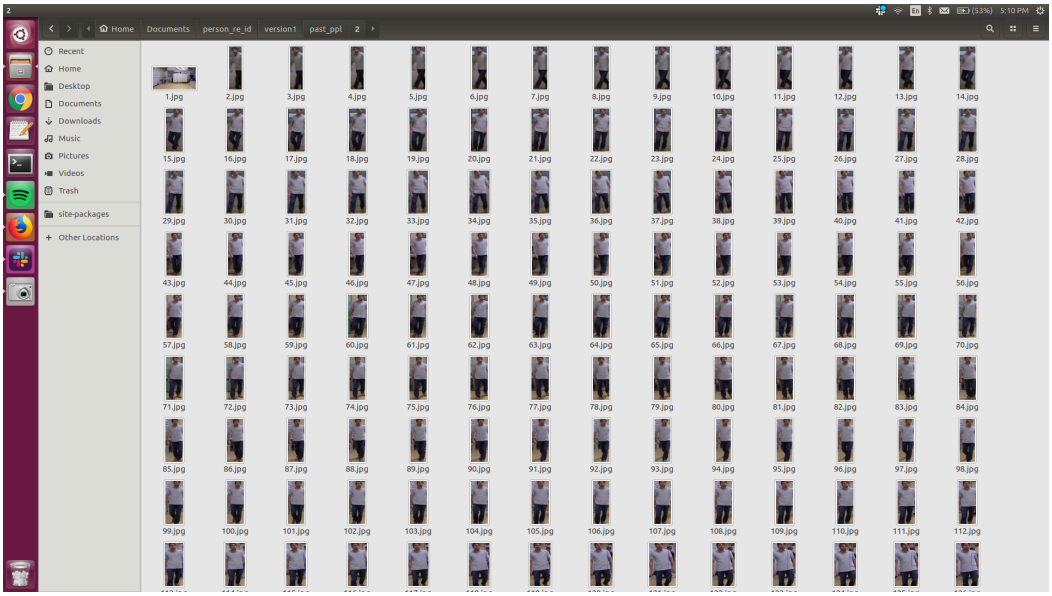


3. Output

This model is run on a video which contains people in it. If a total of N people are seen in the video, the output is a list of N folders, each belonging to one person. Each folder contains cropped images of that person from all frames in the video.
In the following screenshot, there are 10 folders as 10 people have been detected in the video.



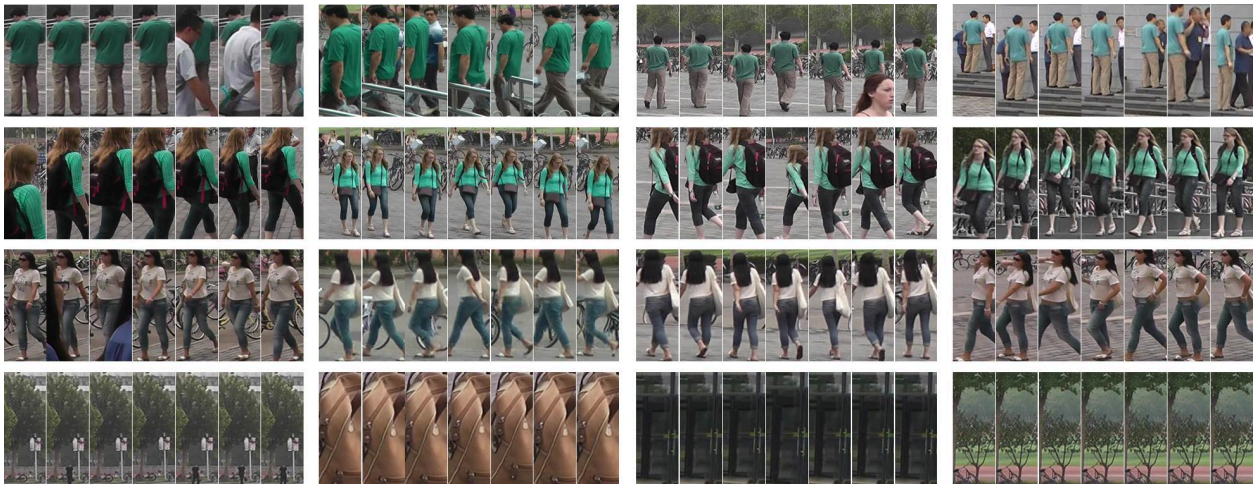
The screenshot below shows the folder for the 3rd person (0 indexing). This folder contains cropped images of this person from the entire video.



4. Related work

A part of this project is a mechanism called person re-identification. There has been a lot of research on person re-identification, both in videos and images. Person re-identification is the deep learning problem of receiving 2 images or 2 video sequences and determining whether they are of the same person or not.
One of the biggest problems in person re-identification is occlusions. If a person is partially occluded by another person in front of him it makes it difficult to re-identify him. Similarly, if many people are very close to each other there will be many occlusions and this makes the task of re-identifying each of them difficult.
Re-identification is solved by using a neural network. It takes as input 2 images or 2 video sequences which are subjected to convolution and recurrent layers (in the case of video sequences as input). The output is often binary, stating whether the 2 people are same or not.

Re-identification models are trained on special person re-identification datasets such as the MARS dataset [1]. The MARS dataset consists of 1,261 different pedestrians who are captured by multiple cameras. Each person has multiple different images.



This project uses the re-identification model described in the paper An Improved Deep Learning Architecture for Person Re-Identification [2]. The code is picked up from here [3]. This model takes a pair of different images as input and outputs either True or False. Each image contains only 1 person. If the images are of the same person, it outputs true. If the people are different, it outputs false. This model is for images, however, I have adapted it for videos.

5. Methodology

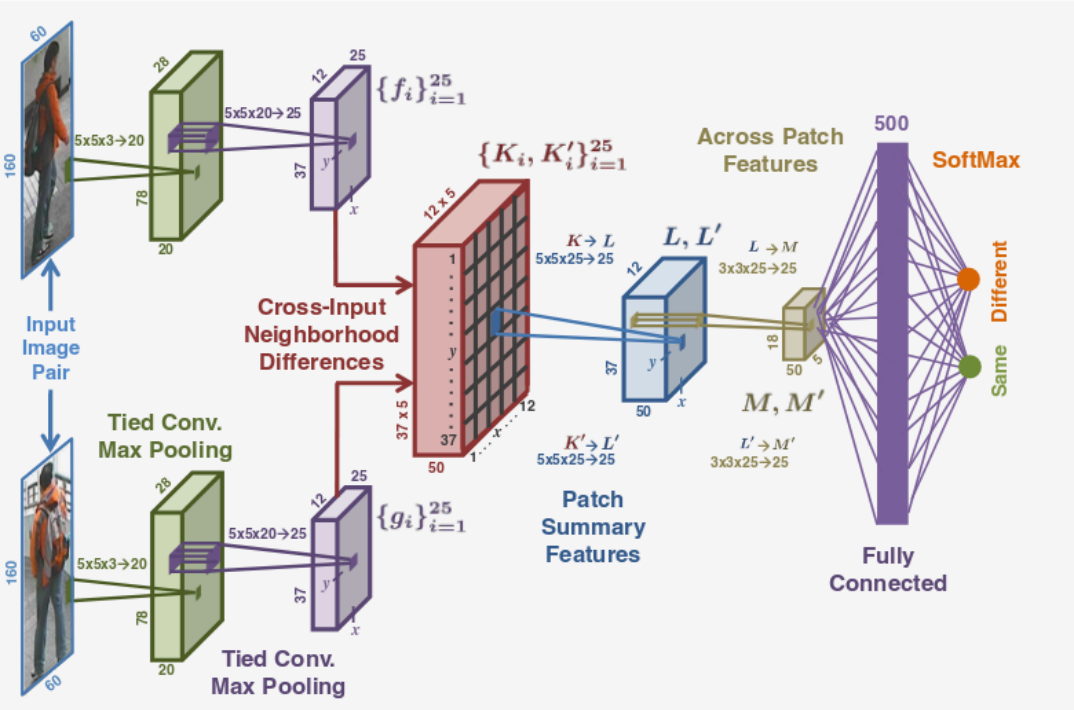
I propose a deep learning approach called person re-identification along with a tracking mechanism. I have used an object detection model found here [4] to detect people in each frame. I have also used the re-identification model found here [3].

Each frame is processed separately. Every frame is passed to the tensorflow object detection model [4], which detects people in it. Bounding boxes for the detected people are returned. These detected people are then checked with the list of previously identified people (the list of folders as explained in section 3) to check if they had appeared earlier. If they have not appeared earlier means that they are being seen in the video for the first time, in which case that person is added to the list of folders as a new person. Hence, a new folder is made for that person. But it could also be that this person was seen before but then left the frame of the video and is now reappearing. It is very important that the model makes this distinction and realizes where it happens because otherwise a lot of people will be counted multiple times.

5.1 Re-identification neural network

As stated previously, the model uses the neural network from [2]. This network takes 2 images of people and returns True if they are the same person and false otherwise.

The model takes as input 2 images of shape (160,60,3). Using convolution layers they are down-sampled to (37,12,25). A module called Cross-Input Neighborhood Differences converts this to (37,12,50). This is eventually flattened to a fully connected layer (500,). The output is a softmax layer with 2 neurons: same and different. The output shape is (2,). If the images are of the same person, same is 1 and different is 0. If the people are different, same is 0 and different is 1.



I have created 5 versions of my model with improvements in the algorithm.

5.2 Version 0: Only re-identification

As stated above, I used the person re-identification model described in the paper An Improved Deep Learning Architecture for Person Re-Identification [2] and I used the code from [3].

The model proposed in the paper uses a CNN to perform person re-identification on static images. Given 2 images, the model predicts whether they are of the same person or not. I then used the Tensorflow object detection model [4] to adapt this model for videos.

My code processes each frame independently. It runs the Tensorflow object detector [4] on each frame and obtains the bounding boxes for people in that frame. Using the obtained bounding boxes, the people are cropped out and sent to the person re-identification model [2]. The people in the current frame are compared pairwise with all the previously detected unique people in the list so far. This determines whether the person was previously detected or not. If the person is not matched with any of the previous people, then he is added to the list as a new unique person. His image is also added to the list of previously detected people so people in the future frames can be compared with this person.

There was a small code optimization I made to make this faster. Initially, I was building the re-identification model again every time I called it. This was making the execution slow. To prevent this, I encapsulated the model in a class. I created an object of that class and then the model would only be created once in the constructor.

5.2.1 Details of the re-identification module

The re-identification module in Version0 is passed an image of a person from the current frame and it needs to find a match in the list of people already identified. Hence, it needs to check the images in the list of folders for a match. If there is no match, it needs to add this person to the list as a new individual, that is, it should make a new folder for this person.

```
//find a match for img, which is an image of a person that needs to be identified
def find(self, img):

        folders = list of folders of past people. If n people have been identified so far, there will be n
folders

        for folder_i in folders:
           files = list of images in ith folder. These are all images of the ith person recorded so far.
           for f in files:
                 //compare img with every image in every folder
              ret = self.reid.compare(img,  f)

                  //if there is a match, store this person in the matched folder
               If ret is True:
                     person_no = len(files) + 1
                     cv2.imwrite(past_ppl + '/' + folder + '/' + str(person_no) + '.jpg',img)
                     return

           // if there is no match with any previous image, this must be a new person
           // make a new folder for this person
        l = len(folders)
        os.makedirs(past_ppl + '/' + str( l )  )
           //save this image of the person in this new folder
        cv2.imwrite(past_ppl + '/' + str( l ) + '/1.jpg',img)
```

5.3 Version1: Tracking using only previous frame + re-identification

The Version0 model calls the re-identification model for every detected person in every frame. This is a very naive and ineffective way to track and identify people in a video. Firstly, it is very slow as the neural network is being called so many times. Secondly, it is inaccurate as it doesn't consider the existing connections between frames. In a video, consecutive frames share a lot of information and most times the people in previous frame are also there in the current frame. Version0 doesn't consider this and treats every frame as a separate image and so loses out on a lot of accuracy.

In version1, I implemented a tracking mechanism and combined it with the re-identification model. Previously, the version0 model would iterate over every frame. In

each frame, the object detection module would return bounding boxes for the people and each detected person in each frame was compared with all the previously detected people using re-identification.

In version1, the model iterates over all frames and in each frame it obtains the person bounding boxes. But it doesn't call the re-identification module every time, only when there is an ambiguity.

The idea is that between 2 consecutive frames, the person would move very little and his bounding boxes from those 2 frames will overlap a lot. I used this property to track people. People once identified are simply tracked across the video by checking the degree of overlap with the bounding boxes of the previous frame. Hence the re-identification neural network doesn't have to be called every time, hence it's faster. However, sometimes this tracking breaks, such as when the person enters the video for the first time or leaves and re-enters the frame. In such a case only the re-identification model is used.

The first time a new person appears, the model will detect it for the first time and store it in the list of people. So in frame number 1, the bounding box positions are known and it is also known which people they belong to. In frame number 2, the object detection model return bounding boxes, but it is not yet know who these people are. So each current box is checked for overlapping with all the previous frame's boxes. In this way, each box is matched with its box from the previous frame. Since the identities are known in the previous frame, they are now also know in the current frame. This way, for the successive frames the model will simply track the path of this person by comparing with the previous frame. Tracking doesn't involve the execution of the neural network and so it is faster.

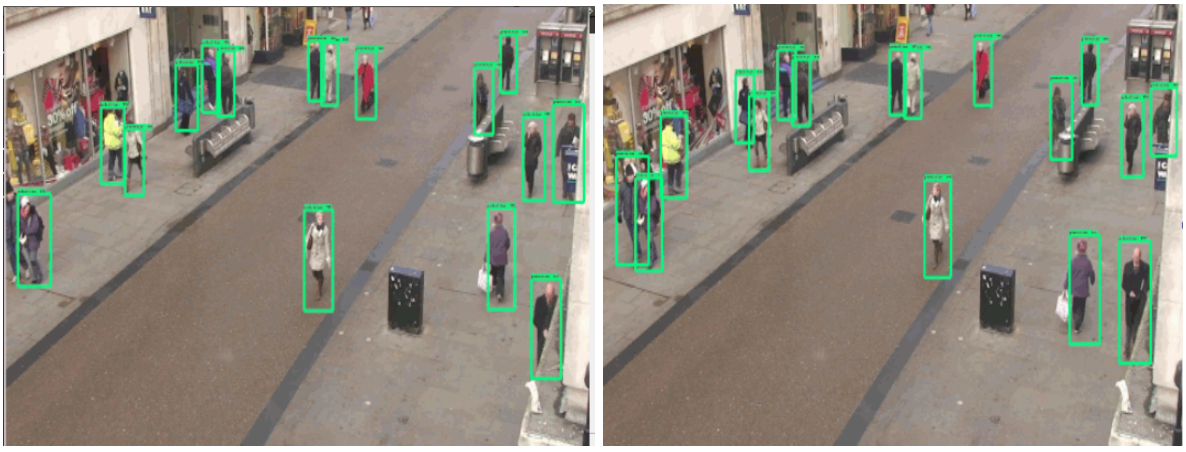
The model stores a list of bounding box positions from the previous frame. It matches the bounding boxes from the current frame to the previous frame boxes that are very close in position. In other words, if the current bounding box is very close to a bounding box from a previous frame- that means that it's the same person who has just slightly moved between frames. This way the model identifies each person only once (when they're first seen) using the neural network. For the next frames, it just tracks the person.

To determine that 2 bounding boxes from consecutive frames are very close, it uses intersection-over-union. IOU is a measure of the overlap between 2 boxes. If the boxes overlap a lot, IOU is close to 1. If the 2 boxes don't overlap at all, IOU is 0.

IOU between 2 boxes = Area of the intersection of boxes / Area of the union of boxes

So, for each bounding box from the current frame, the model tries to find a bounding box from the previous frame which greatly overlaps with it (IOU > 0.9). If such a box is found, the model assigns the previous box's person to the new bounding box. In this way, people are identified without actually running the neural network on them. If a bounding box is not able to match with any previous box, it means that this person just entered the frame and was not there in the previous frame. In this case, the re-identification neural network is run on the person to determine if he has appeared before or if he is a new person totally, in which case he is added to the list of unique people detected. This new person is compared with all the other previously detected people using the re-identification neural net. If a match is found, then that means the person had appeared before but then disappeared for an intermediate period. If a match is not found, that means the person is appearing in the video for the first time and needs to be added to the list of unique people.

The following 2 screenshots are very few frames apart. Hence, for every person, his bounding box in the next frame is very close in position to his bounding box in his previous frame. Hence for every person, the current and previous bounding boxes overlap a lot.



This overlapping is measure by intersection-over-union. For 2 consecutive frames, the 2 bounding boxes of the same person from both frames are extremely close and hence they have an IOU close to 1.

Intersection-over-union is directly proportional to overlapping. If 2 boxes overlap a lot, intersection-over-union is close to 1.

To implement this, I added a intersection-over-union function. This would take 2 boxes and return their intersection-over-union (iou) value.

I also made a list *boxes_prev[]* which would keep a list of the box positions in the previous frame. I also made a list *boxes_cur[]* which would store the boxes in the current frame. The identities of the bounding boxes in *boxes_prev[]* are known and we need to identify the people of the current bounding boxes *boxes_cur[].* Each box in the current frame is compared with the boxes in the previous frame to match it with. 2 boxes match if they are very overlapping, which indicates that they belong to the same person.

5.3.1 Details of Re-identification model

There are some changes from the code in Version0. In Version1 we have implemented a tracking mechanism. This maintains a list boxes_cur[] of boxes in each frame. This list also needs to be updated during re-identification.

```
//find a match for img, which is an image of a person that needs to be identified
def find(self, img, boxes_cur, box):
        folders = list of folders of past people

        for folder_i in folders:
          files = list of images in ith folder. These are images of the ith person recorded so far
          for f in files:
            ret = self.reid.compare(img , f)

              //If there is a match, save the person in the folder.
```

```
        if(ret == True):
                person_no = len(files) + 1
                cv2.imwrite(past_ppl + '/' + folder + '/' + str(person_no) + '.jpg',img)
            //add this person to the list of current people. This is used for tracking in the next frame
                boxes_cur[ int(folder_i) ] = box
                return

l = len(folders)
  //make new folder for this new person and save his image in it
os.makedirs(past_ppl + '/' + str( l )  )
cv2.imwrite(past_ppl + '/' + str( l ) + '/1.jpg',img)

   //add this person to the list of current people. This is used for tracking in the next frame
boxes_cur.append( box )
return
```

5.3.2 Problems in version1

This model only checks the previous frame while tracking. Bounding boxes for each frame are returned by the object detection module. But this module is not 100% accurate all the time. It misses people many times. Many times, the object detection module fails to detect a person in some frame in between. It may detect the person in the frame just before and the frame just after, but it fails to detect the person in the frame in between the two. Hence there are many breaks in the object detection. This causes the tracking to be interrupted and the person is assumed to be a different person, which causes the re-identification module to be called. Due to this, tracking does not happen continuously and the same person is fragmented into multiple folders.

5.4 Version2 : Tracking using k previous frames + re-identification

In Version1, the tracking mechanism was getting interrupted because for many frames in-between, the object detector misses people. This causes a break in the tracking. To prevent this we can use the last k frames for tracking instead of just the previous frame. We can check the k previous frames and see if they contain any bounding boxes that can be matched with the bounding boxes in the current frames.

This way, even if in some previous frames the person was not detected by the object detector, that's okay because the person would have been detected in atleast one of the last k frames. So the tracking will not be interrupted due to missing frames. This is the improvement in the Version1 model.

Just like the previous model, in each frame we obtain bounding boxes. Every bounding box in the current frame is compared with k=25 previous frames to find the most overlapping bounding box. The box in the previous k frames which is most overlapping with the current frame box is said to belong to the same person. This is logical because the bounding box of the same person will only shift a little bit between k frames if k is small and hence will be very overlapping. Using intersection-over-union, the person is tracked across frames. If a new person enters, he is not tracked to anyone in the previous frame and hence a new folder is created for him.

To implement this, I maintain a 2D list *boxes_prev[][].* Each *boxes_prev[i][]* is a list of bounding boxes in the *ith* previous frame. For each box in this list, the model already knows which person it belongs to. Now for the current frame we have *boxes_cur[],* which is a list of the boxes in the current frame. The model needs to identify these people. So it compares each current box with each previous box in *boxes_prev[][]* and the box with the greatest overlap in *boxes_prev[][]* is declared to belong to the same person. Like this, the current frame boxes are tracked and identified.

But sometimes the model is not able to find a sufficiently overlapping box in *boxes_prev[][].* This means that the person was not present in the last k frames. There are 2 possibilities here: the person had appeared before but then disappeared for an intermediate period or the person is being seen in the video for the first time. To determine which case it is, the re-identification module is called. The re-identification module compares this person with each person in the list made so far. If there is a match, it means that this person was seen before. In that case, this new image of the person is just added to his pre-existing folder which already contains many images of

him from previous frames. If there isn't a match it means that the person is being seen for the first time, in which case, he is added to the list of people.

I also made a small change in the re-identification module. The re-identification module iterates over all the past images of people in the list to find a match. Earlier I was only looking for the first match, but now the code only declares a match if more than 70% of the images belonging to a specific person (70% of the images in that folder) match the current image.

5.4.1 Details of re-identification module

In version2 we maintain a 2D list boxes_prev[][] for tracking. Due to this there are some changes in the code below.

```
def find(self, img, boxes_cur, boxes_prev, box):
   folders = list of folders of past people. The ith folder contains images of the ith person.

   for folder in folders:
      files = list of images in each folder. These are all images of the ith person
      same = 0
      diff = 0
      numOfFiles = len(files)
      for f in range(numOfFiles):
          ret = self.reid.compare(img , f)

         //count the matches
         if(ret == True):
                        same += 1
         else:
                        diff += 1

      p = 100 * float(same) / float(same + diff)

      //if this person matches with more than 70% of the images in this folder,
        //it is declared that it's the same person.


     if( p > 70 ):
      person_no = len(files) + 1
      cv2.imwrite(past_ppl + '/' + folder + '/' + str(person_no) + '.jpg',img)
      boxes_cur[ int(folder) ][0] = box
      boxes_prev[ int(folder) ] = -1
      return
```

```
//no match found so create a new folder for this person and add him to it
 l = len(folders)
os.makedirs(past_ppl + '/' + str( l )  )
cv2.imwrite(past_ppl + '/' + str( l ) + '/1.jpg',img)

//add person to list of current boxes
boxes_cur.append( [box] )
```

5.5 Version 3 : Only tracking, no re-identification

This model uses only tracking. This model is obtained by removing the re-identification module from the Version2 model. It tracks people by using the k-previous frame approach. But when a new person is seen (tracking fails to match), the model assumes that he was not seen in the video before and creates a new folder for the person. If person is appearing for the second time, the model will count him twice.

5.6 Version 4: Tracking with k previous frames + re-identification + image processing function to reduce errors in tracking

This model is obtained by adding an image processing function to Version2. In the output of version2, I noticed that some people were getting clubbed into the same folder. Hence the model was determining that these 2 people were the same. This was happening due to a flaw in the tracking mechanism. If at some moment in the video, 2 people cross paths or otherwise overlap, their bounding boxes come very close together. This causes the tracking mechanism to think that they belong to the same person.
To solve this I used an openCV function called matchTemplate().

This function takes 2 images and returns a number indicating how similar they are. This is a pure image processing operation and is not very accurate. However, it can still be used to double check the result of tracking.

Cur_img: The image in the current frame that needs to be identified
Old_img: If the tracking mechanism has identified the current person as the ith person, old_img is an image *of the ith person from a previous frame*

```
res = cv2.matchTemplate(old_img,  cur_img,  cv2.TM_CCOEFF_NORMED)
if(res[0][0] < 0.45)
    Tracking was incorrect, perform re-identification instead
```

So what is happening here is that if 2 people in the video come very close to each other or cross paths with each other, the tracking mechanism will still believe that they are the same person. But this method, matchTemplate(), double checks this. If the 2 people are in fact different, then it will return a low similarity score and this will cause the algorithm to use the re-identification mechanism instead.

6. Results

6.1 Accuracy Metric

I implemented 2 methods to measure accuracy. The first one proved to be an incorrect way of doing so, but the second method reflects the performance of the model well.

6.1.1 Accuracy metric #1

The code iterates over each frame. First it executes the version0 model on the frame. This was the model using only re-identification. This model identifies people by itself and maintains a separate list of people. Then the code executes the version1 model and this produces a separate list of people. Then this list is compared to the version0 list which serves as baseline or ground truth. And the number of matches is the number of correct identifications. So this method runs the version0 and version1 models in parallel and compares their outputs to find accuracy. Like this, we count how many bounding boxes are correctly identified over all the frames. We divide this by the total number of boxes to get the accuracy percentage.

However, this doesn't work. It outputs the wrong accuracy often. I realized that this was because the baseline model itself was quite inaccurate. It was misclassifying many people and grouping together different people. So, even though the latter models were more accurate, they're accuracy was coming low because I was using an inaccurate baseline model as ground truth.

6.1.2 Accuracy metric #2

This code doesn't run the baseline version0 model separately in parallel. It only uses the re-identification module to check the output of the version1 model.

The code iterates over all the frames. In each frame the detected people are identified or re-identified. Let's say there are *n* people in a frame. These n people are identified by a combined effort of the tracking and re-identification mechanisms. Now, each of these detected people is sent to the re-identification module. The other input is another photo from the database. So lets say that the *n* people in the current frame are assigned some *n* identities. Supposing one person in the frame is assigned to person *k* from the list. This means that the model has predicted that this person is identity *k* in the list of already detected people. Now, to check if this is correct, we pass the image of the person from the current frame and a picture of person *k* from the previously saved pictures to the re-identification model. If the model returns true then we say that it's a correct output. Like this, we count how many bounding boxes are correctly identified over all the frames. We divide this by the total number of boxes to get the accuracy percentage.

6.2 Time metric

To calculate the time taken I simply ran the model on a video and measured the time taken for 240 frames.

6.3 Results

Version	**Accuracy**	**Time (240 frames)**
Version0	100%	10 hours
Version1	93% but only 8% of bounding boxes are identified by tracking	10 hours
Version2	98%	40 minutes
Version3	96%	5 minutes
Version4	98%	50 minutes

It is important to realize that there is no absolute ground truth here that we can use to measure accuracy. We are simply using the re-identification module to check this and we are assuming that the re-identification module is the ground truth (which is not always true). Still, this method of measuring accuracy gives us a fairly good idea of how well the algorithm is working.

Let us now interpret the results. Version0 only uses re-identification. It uses the re-identification module to match people to the previously detected people in the list. Now to measure accuracy, if we again use the re-id module to check this, it is obvious that it will return a positive result every time. This is why the accuracy is 100%. This just means that the output is 100% same as the re-identification module.

In version1, ideally most of the bounding boxes should be identified by the tracking mechanism and only a few ambiguous cases should go to the re-identification module. However, the accuracy metric tells us that only 7% of the boxes are being identified through tracking, while 93% are still being sent to the re-identification module. This is not good. As previously stated, the reason this is happening is that the object detection mechanism occasionally fails to detect people. It may detect a person in the k-1th frame and detect the same person in the k+1th frame, but still fail to detect him in the kth frame. These random failures in the object detection cause the tracking mechanism to be interrupted and so tracking fails. This problem is overcome by adopting the k-previous frames approach of Version2. So it is still calling the re-identification neural network most of the time, which is why the time is the same as Version0.

In Version2, the time is reduced to around 40 minutes because most of the time the re-identification neural network is not run. Most of the time only the tracking mechanism runs, which is fast as it's not deep learning. The accuracy is also very high because tracking always returns the correct person and the ones which are not tracked are identified using the re-identification module.

In Version3 we only use tracking. This only takes 5 minutes as no deep learning model is run here. However, this model cannot recognize when a previously seen person reappears in the video. If this happens the model will count the person twice. That is why we need the re-identification based model.

The time and accuracy for version4 is almost equal to that of version2. However, when you look at the output in the folders you will see that different people are not clubbed together in the same folder as it sometimes happens in version2. So version4 is the most accurate and fastest model uptil now that takes re-identification into account.

7. Future work

The biggest problem in this task or any tracking or re-identification task is handling occlusions. The tracking mechanism still fails when there are a lot of occlusions: when people are very close to each other and overlapping each other or when people cross paths with each other and thereby overlapping momentarily. So, occlusion handling is one area which needs improvement.

The use of a tracking mechanism has definitely made the execution faster, but it's uncertain whether this is fast enough for real-time person tracking and re-identification. Hence, the execution speed is also an area which needs to be improved.

Currently, I am using an image re-identification model. This model was made for images, but I have adopted it for videos. Still, the use of a full-fledged video re-identification model instead will improve the accuracy even more. The existing re-identification neural network needs to be replaced with a neural network that can receive video sequences of 2 people and output whether it's the same person or not. This kind of a neural net can make use of the temporal information stored in a video, such as, a person's gait, his speed of walking, etc and will definitely improve the re-identification accuracy.

8. Conclusion

In this project I made 5 different versions of the model. I implemented a tracking mechanism and made use of object detection and person re-identification. The results have been quite positive. The latest model, Version4, can receive a video and output how many people are in the video, a list of their individual identities and their individual video sequences. I have handled the problem of people leaving the video and reappearing later by introducing a re-identification module. I have optimized speed by using a tracking mechanism that doesn't use deep learning. I have also solved the problem of multiple people being clubbed in the same folder by introducing a check after the tracking mechanism using the matchTemplate() function. The performance and speed of the model now are quite good. After this point, I just need to work on increasing the speed and the accuracy using the ideas mentioned in the 'Future Work' section.

9. References

[1] MARS dataset: http://www.liangzheng.com.cn/Project/project_mars.html

[2] Re-identification model paper https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Ahmed_An_Improved_Deep_2015_CVPR_paper.pdf

[3] Re-identification model code on GitHub https://github.com/digitalbrain79/person-reid

[4] Tensorflow object detection models: https://github.com/tensorflow/models/tree/master/research/object_detection

Intersection-over-union: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Github repo: https://github.com/Nirvan101/Person-Re-identification