18BCE108

by Dhaval Makwana

Submission date: 17-May-2022 01:20PM (UTC+0530)
Submission ID: 1838216236
File name: 18BCE108.docx (6.11M)
Word count: 4402
Character count: 25669

LIST OF FIGURES

CONTENTS

1. Introduction

1.1 Video Summarization

Video summarization is the process of providing a brief summary of the content of a long video by selecting and showing the most informative or intriguing items to potential consumers. The output video summary is often made up of a collection of keyframes or video clips taken from the source video by some sort of editing procedure.

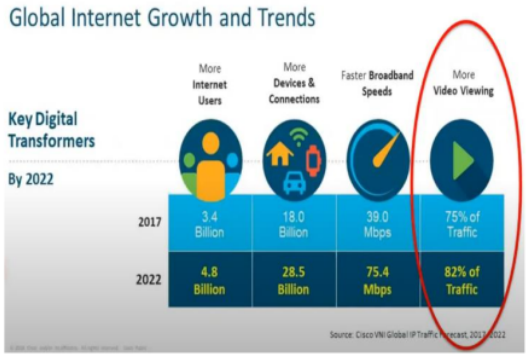


Figure 1.1: (Source of image: https://www.youtube.com/watch?v=dHp5I0m9_zA)

Here are some statistics made by Cisco that is by the year of 2022 more than 80% of internet traffics are going to be real data except in particular could be video data. So, the majority of data flowing around the internet is our videos but currently, we don't have very good tools to browse and analyses those data.

Another estimate is that there are more than 25 million CCTV cameras globally. All cameras are recording videos 24 x 7 days a week. So, to maintain this all recording we still hire a people and try to send an alarm if any bad activities happened. It is extremely time-consuming and expensive. So, if we could have a video summarization tool like we could record a video in one day and summarize that recording in another day.

The main goal of video summarization is to view a vast collection of video information while also achieving efficient access and representation of video data. Users may make rapid choices about the usefulness of the video by watching the synopsis. Depending on the application and intended users, summary evaluation may include usability tests to assess the content informativeness and quality of a summary.

1.2 Type of Video Summarization

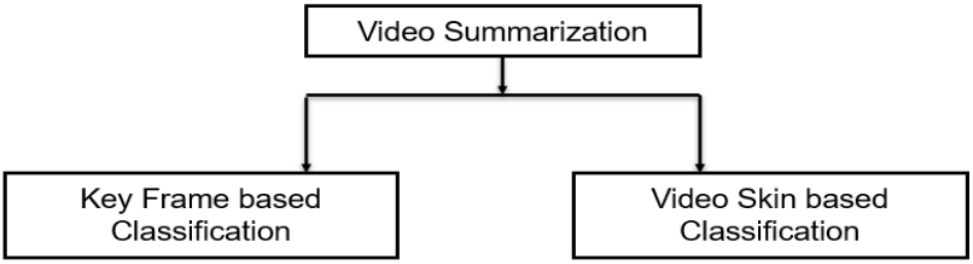


Fig 1.2 Types of Video Summarization

1.2.1 Key Frame-based Classification:

This strategy depends on the spatial division of each frame to recognize significant events. Along these lines, keyframe detection is confronting a substantially more semantic standard so that each keyframe presents a significant event like the appearance and the vanishing of a relevant objects.

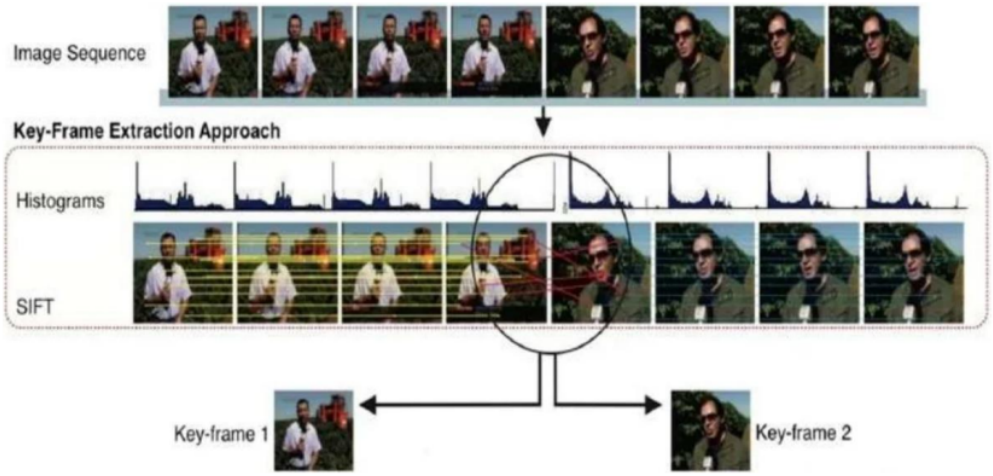


Fig 1.3 (Source of image: https://medium.com/@myworldsharma.jay/key-frame-extraction-from-video-9445564eb8ed)

1.2.2 Video Skin-based Classification:

Video skimming is also called a dynamic video summary, which makes a period condensed form of a video. Skimming can be accomplished by finding relevant components in either uni-modal or multimodal video characteristics. Because video skimming is dynamic in nature, it

provides for a deeper grasp of the video from its summary through temporal connectedness. With this benefit, video skimming has as of late stood out enough to be noticed by various academics, who benefit from the prepared accessibility of the essential computational assets. This is likewise called a moving-picture unique, moving storyboard, or summary grouping. This original video is segmented into various parts which is a video clip with a shorter duration.

Fig 1.4 (Source of image: https://www.researchgate.net/figure/Video-2-example-frame-and-its-classification-result-with-a-near-skin-color-background_fig2_220845268)

1.3 Problem Definition

In this project work we have to work on Video Summarization by Using Yolo Algorithm. We suppose to use video or web came as input based on that we need to summarize the video.

Another estimate is that there are more than 25 million CCTV cameras globally. All cameras are recording videos 24 x 7 days a week. So, to maintain this all recording we still hire a people and try to send an alarm if any bad activities happened. It is extremely time-consuming and expensive. So, if we could have a video summarization tool like we could record a video in one day and summarize that recording in another day.

2. R-CNN

The R-CNN algorithm is a two-stage detection technique. The first stage selects a selection of picture areas that may contain an item. The item is classified in each region in the second stage. Autonomous driving is one use for R-CNN object detectors.

2.1 Algorithm Structure

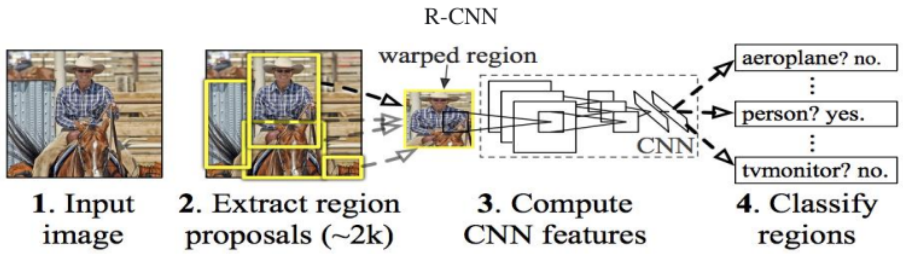


Fig 2.1 (Source of image: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection algorithms-36d53571365e)

Rather than dealing with large regions, the RCNN algorithm recommends various boxes in the image and decides whether any of these boxes contain any objects. To extract these boxes from a picture, RCNN employs selective search (these boxes are called regions)

R-CNNs (Region-based Convolutional Neural Networks) is a kind of ML model that is utilized in picture handling and computer vision. The first objective of any R-CNN, which is explicitly intended for object identification, is to detect objects in any input picture and characterize boundaries around them.

Follow this link to learn more about the selective search algorithm. These 2000 competitor region recommendations are squared and input to a CNN, which yields a 4096-layered include vector. The Conventional Neural Network capacities as an element extractor and the result dense layer are comprised of the extricated highlights from the image, which are then input into a Support Vector Machine to group the presence of the thing inside that applicant locale idea. As well as foreseeing the presence of an object inside the area determined, the framework predicts four offset values to help the bounding box's precision.

An item is composed of four distinct regions: varied sizes, colors, textures, and enclosure. Selective search recognizes these patterns in the image and offers various regions depending on them. Here's a quick rundown of how:

2.2 Algorithm's Steps

Step-1: It first takes an image as input.

Step-2: Then it produces initial sub-segments, resulting in various areas from this image.

Step-3: The approach then joins the related sections to create a bigger zone (based on color, texture, size, and shape compatibility).

Step-4: Finally, the final object placements are generated by these areas (Region of Interest).

2.3 Problem with RCNN

So far, we've seen how RCNN can help with object detection. However, this approach has its own set of restrictions. Training an RCNN model is costly and time-consuming due to the following steps:

1. Using a selective search, 2,000 zones are extracted for each image.
2. For each picture area, features are extracted using CNN. If we have N photos, the number of CNN features is N*2,000.
3. The complete object detection procedure using RCNN is divided into three models:
    - CNN is used to extract features.
    - Object recognition with a linear SVM classifier
    - The regression model is used to tighten the bounding boundaries.

All of these processes work together to make RCNN very sluggish. When presented with a massive dataset, it takes roughly 40-50 seconds to produce predictions for each new image, making the model complicated and nearly difficult to create. The good news is that we now have another object detection algorithm that addresses the majority of the issues we encountered with RCNN.

3. Faster R-CNN

Faster R-CNN is a single-stage model that is trained from start to finish. It generates region suggestions using a unique region proposal network (RPN), which saves time when compared to classic algorithms like Selective Search. It extracts a fixed-length feature vector from each area proposed using the ROI Pooling layer.

3.1 What is the Faster R-CNN algorithm?

What more can we do to lower the usual computing time of an RCNN algorithm? Rather than running a CNN 2,000 times per picture, we can run it once and retrieve all of the regions of interest (regions containing some object).

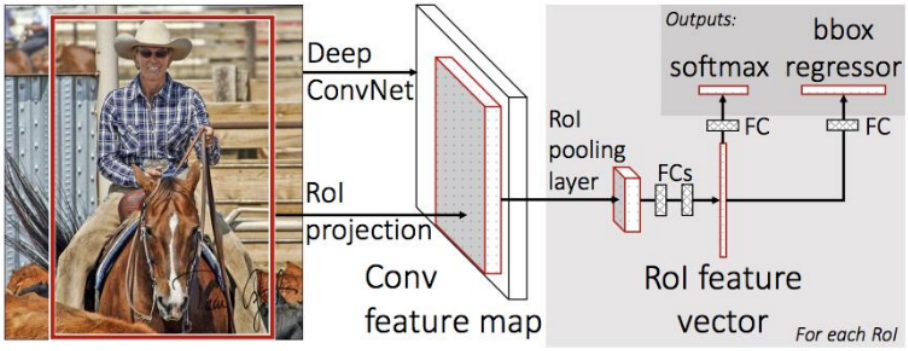


Fig 3.1.1 (Source of image: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e)

The same author of the previous study (R-CNN) give some of the R-shortcomings CNNs in order to create a faster object detection system, which was dubbed Fast R-CNN. The method is comparable to the R-CNN algorithm.

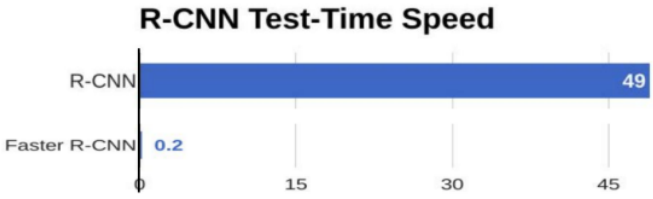


Fig 3.1.2 (Source of image: https://www.researchgate.net/figure/Comparison-of-Test-Time-speed-of-different-Object-detection-algorithms_fig5_350372828)

The graph above shows that Quicker R-CNN is significantly faster than its predecessors. As a result, it may be utilized for real-time object detection.

The creator of R-CNN, Ross Girshick, came up with the notion of executing the CNN only once per picture and then finding a mechanism to spread that computation over the 2,000 areas. We pass the input picture to the CNN in Fast RCNN, which creates the convolutional feature maps. The regions of suggestions are extracted using these maps.

3.2 Faster RCNN technique

1. We start with an image and give it to the ConvNet as a input, which returns the feature map for that image.
2. On these feature maps, a region proposal network is used. This method returns the object suggestions as well as their objectless score.
3. On these proposals, and RoI pooling layer is used to decrees the size of all proposals to the same size.
4. Finally, the suggestions are sent to a fully connected layer with a SoftMax layer and a linear regression layer on top of categories and output item bounding boxes.

3.4 Problem with Faster RCNN

The reason "Faster R-CNN" is quicker than R-CNN is that you don't have to give input to the CNN 2000 region suggestions every time. Instead, the convolution method is performed just once per picture, and a feature map is produced as a result.

4. YOLO

YOLO stands for You Only Look Once, and the fundamental idea is to place a grid (typically 19x19) on the image, with only one cell, the one bearing the center/midpoint of an item, responsible for identifying this object.

YOLO is a real-time object identification technique that uses neural networks. This algorithm is well-known for its speed and precision. It has been used to detect traffic signals, parking metrics, animals, etc.

4.1 Basic Introduction

To detect objects in real-time, the YOLO method leverages convolutional neural networks (CNN). To identify objects, this technique is required only a single forward propagation through a neural network, as the name suggests.

This indicates that the complete image is predicted in a single algorithm run. The CNN is used to forecast several class probabilities and bounding boxes at the same time. There are several variations of the YOLO algorithm.

4.2 Why YOLO algorithm is important?

1) Speed: Because it can forecast objects in real-time, this technique boosts detection speed.
2) High precision: YOLO is a prediction approach that produces precise findings with minimum background noise.
3) Learning skills: The algorithm has incredible learning capacities, permitting it to learn object representations and use them in object detection.

4.3 Application of YOLO:

1) Object Detection: Object detection is the technique of finding and identifying a variable number of objects on a picture. The main difference is the "variable" part. Conversely with issues like classification, the yield of object detection is variable in length, since the quantity of

objects detected may change from picture to picture. Performing Yolo model we can detect various objects, for example – chair, person, table, fruits, mobile, laptop, etc

2) Person detection: Individual detection has several uses in a variety of sectors. Security applications that track who's travelling where, who's arriving and departing, as well as safety systems designed to keep people safe, are common use cases.

To recognize the presence of people in an image, we use a method called object-detection in computer vision. Individuals are sometimes the only item that an object detection model is capable of detecting. Similarly, this technique differs from face recognition in that it does not identify a specific individual, but rather recognizes when a human is in the picture.

3). Autonomous driving: The YOLO algorithm can be used in self-driving cars to identify things in the environment such as automobiles, pedestrians, and parking signals. Object detection is used in self-driving automobiles to avoid collisions because no human driver is in charge of the vehicle.

4) Wildlife detection: This algorithm detects numerous sorts of animals in the woods. Wildlife rangers and journalists utilize this form of detection to identify animals in films (both recorded and real-time) and photos. Giraffes, elephants, and bears are among the creatures that can be recognized.

5) Security: YOLO may also be employed in security systems to enforce security in a certain region. Assume that individuals are not permitted to travel through a specific location for security reasons. If someone enters the restricted area, the YOLO algorithm will detect him/her, prompting security officers to take further action.

5. How YOLO Works

YOLO is a Deep Learning architecture proposed by joseph Redmon, Santosh Divvala, and Ross. YOLO is a Real-Time object detection that uses a totally different approach. It's a clever convolutional neural network (CNN) for object detection

5.1 Yolo Algorithm

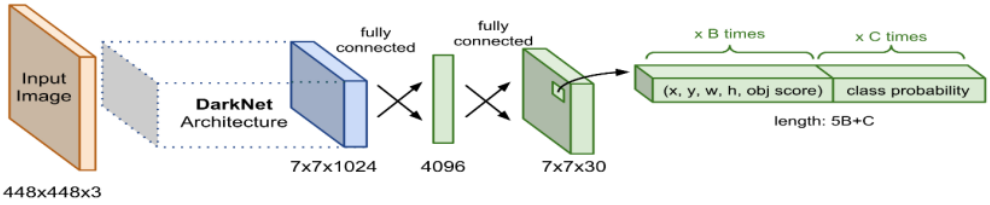


Fig 5.1 (Source of image: https://www.youtube.com/watch?v=dHp5I0m9_zA)

Yolo Algorithm works using **Residual blocks, Bounding box regression, and Intersection over Union (IOU).**

5.1.1 Residual Blocks:

To begin, the picture is separated into grids. Each grid has a S x S dimension. The graphic below demonstrates how an input image is separated into grids.

Fig 5.1.1 (Source of image: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/)

5.1.2 Bounding Box Regression:

A bounding box is an outline that draws attention to an object in a picture. Each bounding box in the picture has the following attributes: width (bw), height (bh)
The letter c represents a class (for example, a person, an automobile, a traffic light, and so on). The centered bounding box (bx, by) the graphic below depicts an example of a bounding box. A yellow outline has been used to depict the bounding box.

Fig 5.1.2 (Source of image: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/)

5.1.3 Intersection over Union (IOU):

Intersection over Union (IOU) is an object detection phenomenon that defines how boxes overlap. YOLO uses IOU to create an output box that properly surrounds the items.

Each grid cell is in charge of forecasting the bounding boxes as well as their confidence ratings. If the expected and actual bounding boxes are the same, the IOU is equal to one. This approach removes bounding boxes that aren't equivalent to the actual box. The graphic below is a simple illustration of how IOU works.

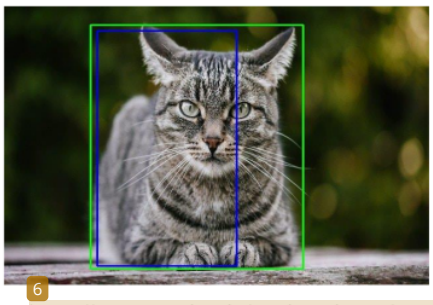


Fig 5.1.3 (Source of image: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/)

5.2 Combination of the three techniques:

We can see at least three types of items, for example, a vehicle, a dog, and a bicycle. A single convolutional neural network is used to make all of the predictions at the same time.

Intersection over union assures that the expected bounding boxes of the objects are equivalent to their actual boxes. This phenomenon removes unneeded bounding boxes that do not correspond to the properties of the objects (like height and width). The final detection will consist of distinct bounding boxes that exactly suit the objects.

The pink bounding box, for example, surrounds the automobile, whereas the yellow bounding box surrounds the bicycle. The blue bounding box has been used to highlight the dog.

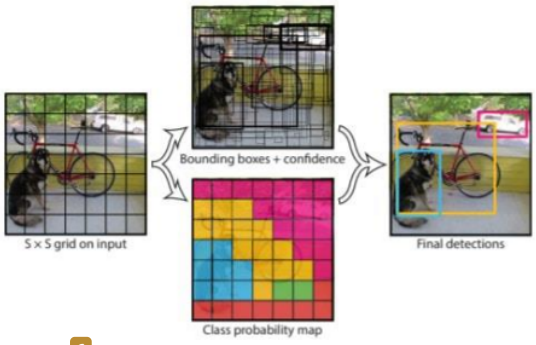


Fig 5.2 (Source of image: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/)

6. YOLOv4

It is a strong and efficient object detection model that allows anybody with a 1080 Ti or 2080 Ti GPU to train a super-fast and accurate object detector. The impact of cutting-edge "Bag-of-Freebies" and "Bag-of-Specials" item identification systems on detector training has been demonstrated.

State-of-the-art approaches, such as CBN (Cross-iteration Batch Normalization), PAN (Path Aggregation Network), and others, have been improved and are now suited for single GPU training. YOLOv4 is a one-stage object detection model that improves on YOLOv3 by including numerous techniques and modules from the literature.

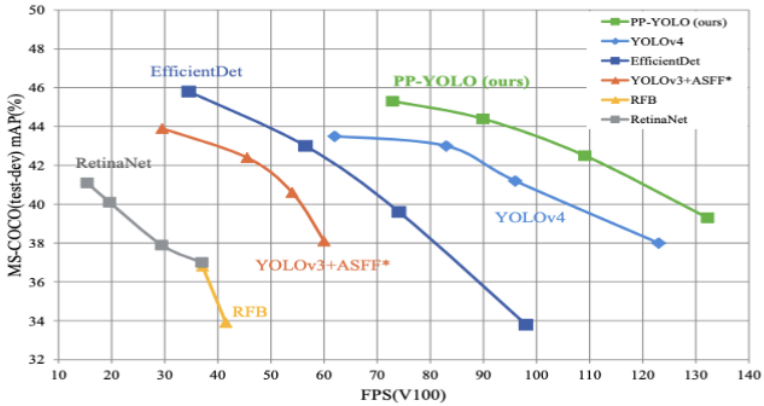


Fig 6.1 (Source of image: https://blog.roboflow.com/pp-yolo-beats-yolov4-object-detection/)

YOLOv4 was essentially an amalgamation of several well-known computer vision technologies that were merged and tested during the research process.

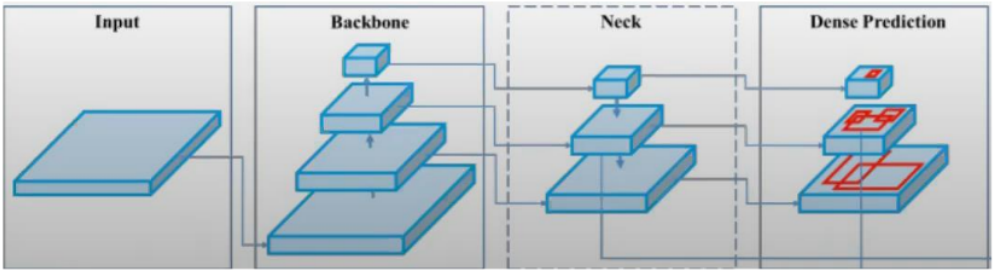


Fig 6.2 (Source of image: https://www.youtube.com/watch?v=dHp5I0m9_zA)

7. Image Augmentation

Image augmentation is a technique for modifying existing images in order to generate extra data for the model training process. To put it another way, it is the technique of artificially increasing the available dataset for training a deep learning model.

Zoom, shear, rotation, preprocessing function, and other image augmentation parameters are commonly employed to enhance the data sample count. When these parameters are used, pictures with these qualities are generated during Deep Learning model training. Image samples created by image augmentation often result in a 3x to 4x increase in the size of the current data sample collection.

Image augmentation in Keras is accomplished through the use of a function named Image Data Generator. The following is a basic outline of the function definition.

```
import Augmentor
```

```
p = Augmentor.Pipeline(r"C:\Users\Dhaval Makvana\Desktop\books\sem 8\obj")
p.zoom(probability=0.3, min_factor = 0.8, max_factor = 1.5)
p.flip_top_bottom(probability =0.4)
p.random_brightness(probability=0.3, min_factor = 0.3, max_factor = 1.2)
p.random_distortion(probability=1, grid_width=4, grid_height=4,magnitude=8)
p.sample(1000)
```

Using this code, we are generating around 3000 images for labeling and training. Here, we are giving a simple example of a person and showing how it will be worked.

Fig 7.1 Image Augmentation Result

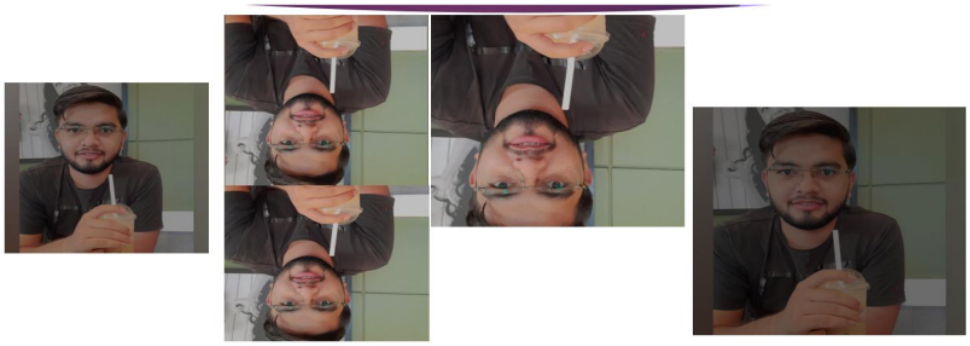


Fig 7.2 Image Augmentation Result

8. Image Labeling

The process of recognizing and categorizing distinct elements in a picture is known as image labeling. Image tagging is extremely handy when it comes to automating the process of producing metadata or providing suggestions to consumers based on features in their photographs.

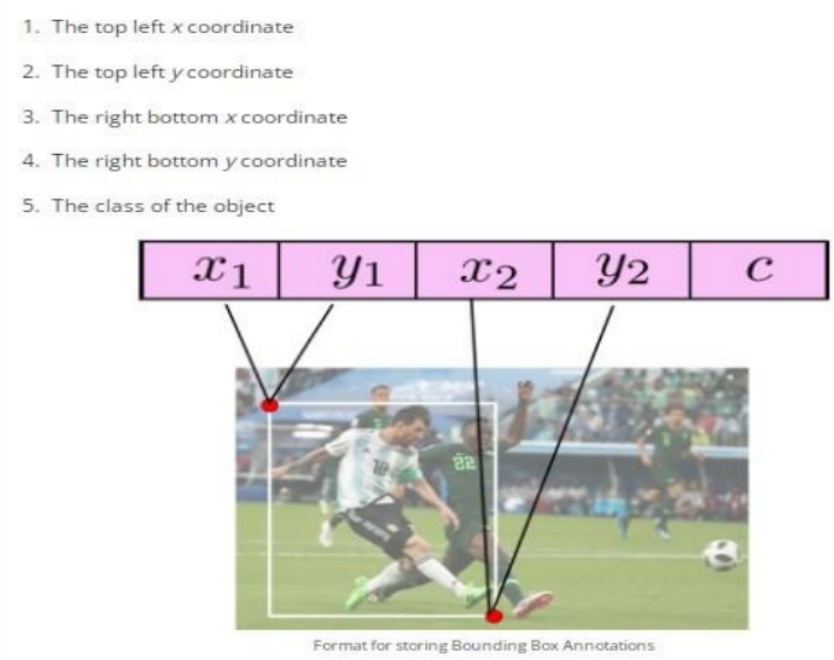


Fig 8.1 Image Labeling

Components linked together Labeling analyses a picture and divide its pixels into components based on pixel connectedness, which means that all pixels in a linked component have comparable pixel intensity values and are related in some way.

Labeling Images is done using 7 ways:

1. Label Every Object of Interest in Every Image.
2. Label the Entirety of an Object.
3. Label Occluded Objects.
4. Create Tight Bounding Boxes.
5. Create Specific Label Names.
6. Maintain Clear Labeling Instructions.
7. Use These Labeling Tools.

Fig 8.2 labeling the image using tool

9. Methodology

9.1 Training the YOLO Model:

To train the models, 3000 photos are used for training. From 3000 photographs from each class's 1200 (1200+1200 for each classes) images were utilized as training data, while the remaining images were used as testing/validation data. Simply said, the data was split in an 8:2 ratio, or 80:20, with 80 percent utilized for training and 20 percent used for validation.

The models were trained until the loss fell below one. Another benefit of YOLO is that it uses many fewer photos to train on an item than other object detection methods. 3000 photographs for one type of produces extraordinarily excellent accuracy.

9.2 Face Detection

Simple YOLO algorithm is detecting 80 different classes. What we did that, we performed object detection only for two classes SAGAR and DHAVAL. First, we have only 200 images, so, to increase image for better training model we did Image augmentation to create 3000 images earlier. After that, we did image labeling for each image by create bounding boxes and assigning the class name SAGAR and DHAVAL. After that we used YOLOv4 for training the 3000 images and then we test two class SAGAR and DHAVAL to recognise the Face.

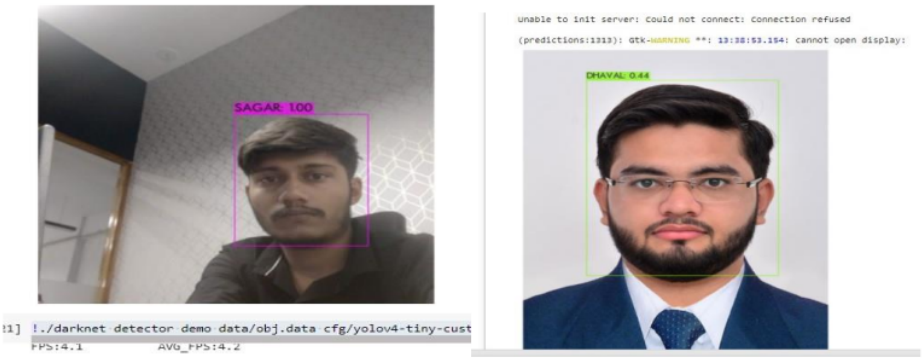


Fig 9.2 Face Detection Result

After training the images run the test image to check the accuracy. If Accuracy is low then increase the size of the test image.

10. Video Summarization

The goal of video summarizing is to create a short overview that summarizes the video content by selecting the most informative and important elements. The description of the video provides a summary of the material. It goes through the things that were presented in detail. It does not have to be particularly long. In reality, it might consist of five or more phrases. Because summarizing videos is not a subjective writing effort, writers should refrain from including their personal opinions.

10.1 Code

```
NMS_THRESH = 0.1
MIN_CONF = 0.1
import numpy as np
import cv2

# function to detect people
def detect_people(frame, net, ln, personIdx=0):
    # grab dimensions of the frame and initialize the list of results
    (H, W) = frame.shape[:2]
    results = []

    # construct a blob from the input frame and then perfrom a forward pass
    # of the YOLO object detector, giving us the bounding boxes and
    # associated probabilities
    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416), swapRB=True, crop=False)
    net.setInput(blob)
    layerOutputs = net.forward(ln)

    # initialize lists of detected bounding boxes, centroids, and confidence
    boxes = []
    centroids = []
    confidences = []

    # loop over each of the layer outputs
    for output in layerOutputs:
        # loop over each of the detections
        for detection in output:
            # extract teh class ID and confidence(probability) of the current object detection
            scores = detection[5:]
            classID = np.argmax(scores)
            confidence = scores[classID]

            # filter detections by (1) ensuring that the object detected was a person and
            # (2) that the minimum confidence is met
            if classID == personIdx and confidence > MIN_CONF:
```

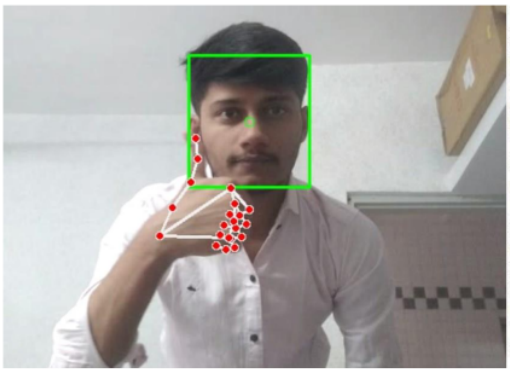
```
                # scale the bounding box coordinates back relative to the size of
                # the image, keeping in mind that YOLO actually returns the center (x, y)-coordinates
                # of the bounding box followed by the boxes' width and height
                box = detection[0:4] * np.array([W, H, W, H])
                (centerX, centerY, width, height) = box.astype("int")

                # use the center (x,y)-coordinates to derive the top and left corner of
                # the bounding box
                x = int(centerX - (width / 2))
                y = int(centerY - (height / 2))

                # update the list of bounding box coordinates, centroids and confidences
                boxes.append([x, y, int(width), int(height)])
                centroids.append((centerX, centerY))
                confidences.append(float(confidence))

    # apply non-maxima suppression to suppress weak, overlapping bounding boxes
    idxs = cv2.dnn.NMSBoxes(boxes, confidences, MIN_CONF, NMS_THRESH)

    # ensure at least one detection exists
    if len(idxs) > 0:
        # loop over the indexes being kept
        for i in idxs.flatten():
            # extract the bounding box coordinates
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])

            # update the results list to consist of the person prediction probability,
            # bounding box coordinates, and the centroid
            r = (confidences[i], (x, y, x + w, y + h), centroids[i])
            results.append(r)

    # return the list of results
    return results
```

```
# load the COCO class labels our YOLO model was trained on
labelsPath = os.path.sep.join(["A:/SAGAR/OIDv4_ToolKit-master - Copy/obj.names"])
LABELS = open(labelsPath).read().strip().split("\n")
CLASSES = ["SAGAR"]

# derive the paths to the YOLO weights and model configuration
weightsPath = os.path.sep.join(["A:/SAGAR/yolov4-tiny_final.weights"])
configPath = os.path.sep.join(["A:/SAGAR/yolov4-tiny.cfg"])

# load our YOLO object detector trained on COCO dataset (80 classes)
print("[INFO] loading YOLO from disk...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

# determine only the *output* layer names that we need from YOLO


# initialize the video stream and pointer to output video file
print("[INFO] accessing video stream...")
ln = net.getLayerNames()
ln = [ln[i - 1] for i in net.getUnconnectedOutLayers()]

# initialize the video stream and pointer to output video file
print("[INFO] accessing video stream...")
vs = cv2.VideoCapture('A:/SAGAR/OIDv4_ToolKit-master - Copy/pose_video.mp4')
writer = None

# loop over the frames from the video stream
while True:
    # read the next frame from the file
    (grabbed, frame) = vs.read()
    print("running")
    # if the frame was not grabbed, then we have reached the end
    # of the stream
    if not grabbed:
        break

    # resize the frame and then detect people (and only people) in it
    frame = imutils.resize(frame, width=700)
    results = detect_people(frame, net, ln,personIdx=LABELS.index("SAGAR"))

    print(results)
```

```
    # loop over the results
    for (i, (prob, bbox, centroid)) in enumerate(results):
        # extract the bounding box and centroid coordinates, then
        # initialize the color of the annotation
        (startX, startY, endX, endY) = bbox
        (cX, cY) = centroid
        color = (0, 255, 0)


        # draw (1) a bounding box around the person and (2) the
        # centroid coordinates of the person,
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
        cv2.circle(frame, (cX, cY), 5, color, 1)


    # check to see if the output frame should be displayed to our
    # screen
    if args["display"] > 0:
        # show the output frame
        #cv2.imshow("",frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
            break

    # if an output video file path has been supplied and the video
    # writer has not been initialized, do so now
    if args["output"] != "" and writer is None:
        # initialize our video writer
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(args["output"], fourcc, 25,
            (frame.shape[1], frame.shape[0]), True)

    mpHands = mp.solutions.hands
    hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.7)
    mpDraw = mp.solutions.drawing_utils
```

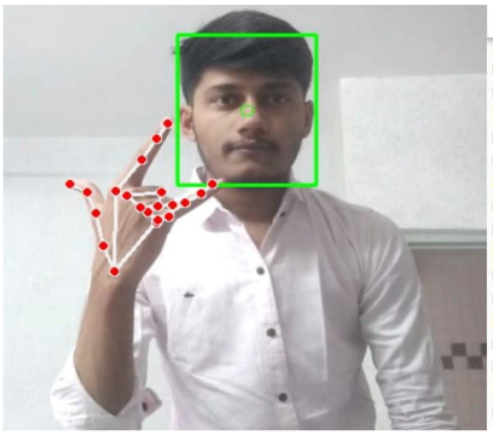
```
    # Load the gesture recognizer model
    model = load_model('mp_hand_gesture')

    # Load class names
    f = open('gesture.names', 'r')
    classNames = f.read().split('\n')
    f.close()

    x, y, c = frame.shape

    # Flip the frame vertically
    frame = cv2.flip(frame, 1)
    framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Get hand landmark prediction
    result = hands.process(framergb)

    # print(result)

    className = ''
    prediction  = []
    # post process the result
    if result.multi_hand_landmarks:
        landmarks = []
        for handslms in result.multi_hand_landmarks:
            for lm in handslms.landmark:
                # print(id, lm)
                lmx = int(lm.x * x)
                lmy = int(lm.y * y)

                landmarks.append([lmx, lmy])

            # Drawing landmarks on frames
            mpDraw.draw_landmarks(frame, handslms, mpHands.HAND_CONNECTIONS)

            # Predict gesture
            prediction = model.predict([landmarks])

            classID = np.argmax(prediction)
            className = classNames[classID]
            print(className)

    # show the prediction on the frame
    cv2.putText(frame, className, (10, 50), cv2.FONT_HERSHEY_SIMPLEX,
                   1, (0,0,255), 2, cv2.LINE_AA)

    # Show the final output
    #cv2.imshow("Output", frame)

    # if the video writer is not None, write the frame to the output
    # video file
    if writer is not None:
        if results and prediction:
            writer.write(frame)
```

10.2 Output



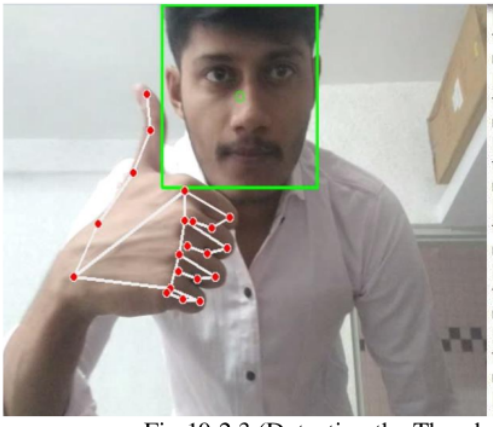
```
[(0.8805662989616394, (264, 76, 371, 202), (318, 139))]
call me
running
[(0.745705246925354, (265, 71, 372, 206), (319, 139))]
call me
running
[(0.8247552514076233, (263, 57, 377, 217), (320, 137))]
call me
running
[(0.8468935489654541, (265, 57, 385, 211), (325, 134))]
call me
running
[(0.9085741639137268, (263, 55, 386, 210), (325, 133))]
call me
running
[(0.9479851722717285, (265, 53, 387, 213), (326, 133))]
call me
```

Fig 10.2.1 (Detecting the call me class form Gesture Detection)



```
[(0.9589105844497681, (255, 48, 357, 180), (306, 114))]
rock
running
[(0.6763243675231934, (251, 43, 354, 181), (303, 112))]
rock
running
[(0.931380033493042, (249, 47, 351, 176), (300, 112))]
rock
running
[(0.9624115228652954, (244, 48, 348, 174), (296, 111))]
rock
running
[(0.9660375118255615, (241, 46, 348, 176), (295, 111))]
rock
running
[(0.9802072644233704, (243, 46, 349, 175), (296, 111))]
rock
```

Fig 10.2.2 (Detecting the Rock class form Gesture Detection)



```
[(0.9876931309700012, (287, 32, 391, 160), (339, 96))]
thumbs up
running
[(0.9800359010696411, (286, 22, 390, 150), (338, 86))]
thumbs up
running
[(0.980064332485199, (285, 22, 387, 146), (336, 84))]
thumbs up
running
[(0.9886481165885925, (282, 20, 385, 146), (334, 83))]
thumbs up
running
[(0.9907773733139038, (280, 18, 386, 143), (333, 81))]
thumbs up
running
[(0.9948617219924927, (279, 9, 384, 150), (332, 80))]
thumbs up
running
[(0.994825541973114, (277, 9, 384, 148), (331, 79))]
```

Fig 10.2.3 (Detecting the Thumbs up class form Gesture Detection)

Fig 10.2.4 (Detecting the Stop class form Gesture Detection)

11. Comparison

There are several object detection techniques, including R-CNN, Faster R-CNN, and YOLO. In this part, we'll go through why we chose YOLO as our default algorithm. This is accomplished by advising other algorithms on their limits as well as how YOLO counteracts/overcomes those constraints. R-CNN (Region-based Convolutional Neural Networks): The emphasis is on splitting the image into 2000 tiny parts and then locating the item in each region; an offset is also used to increase precision.

Limitation: It takes a long time to categorize 2000 sections. Because each image takes around 47 seconds to analyze, it is not practicable for real-time detections (classify). R-CNN is faster since the network itself learns the region suggestions. It is significantly quicker than R-CNN and Fast R-CNN, but it is not suited for real-time performance.

It can be shown that practically every algorithm is built on three tasks to detect objects: 1) Use the Region Proposal technique to build potential bounding boxes in an image. 2) Apply the classifier to these boxes. 3) After classification, do post-processing to tighten the bounding box borders and delete duplicates.

These challenges show to be exceedingly complicated and difficult to optimize at the same time since each component must be trained individually. This is when the phrase "YOLO" comes into play.

This technique differs from all previous object detection algorithms in that it views object detection as a regression problem rather than a classification one. To forecast the bounding box as well as the probability, a single convolutional neural network is employed. A prediction with the threshold (Maximum IOU) is regarded to be detected, while the remainder is rejected. The speed is on the order of 45 frames per second, which is currently the fastest accessible algorithm. Because of the method's spatial restrictions, the system cannot detect tiny things. Another shortcoming of the method is that it is less accurate than other algorithms such as Faster R-CNN.

Method	Advantage	Disadvantage
R-CNN	This network transforms the object detection problem into the classification problem and greatly improves the accuracy.	It generates a partially overlapping candidate area from each detection target.
Faster R-CNN	This network is faster than Fast R-CNN and no longer depends on a region selection algorithm.	The training process is complex, and there is still much room for optimization in the calculation.
YOLO	The network can meet the real-time requirements by using the full image as context information.	It is relatively sensitive to the scale of the objects, and the effect of small target detection is not good. That's why YOLOv4 comes.

12. YOLO Algorithm Application

- Duration adaptation
- Video ads customization
- Efficient content viewing
- Highlights creation
- Consumer Video Application to view the interesting parts quickly
- Traffic Monitoring

13. Summary

So, summary of this video summarization process is that we can reduce the size of the video of original video and store in the database. Main advantage of doing these activities is that we can save time, cost, memory etc.

Additional things are we also add face detection process in this algorithm. So, advantage of doing this is, if we want to remove particular person form the frame and store into another video, we can do that. If we want to reduce unnecessary person or frame in the video, we can do that also. We also can make a short movie using particular person's activity.

Video summarization are used in many fields like in camera's recording for Traffic Monitoring, create cricket/kabaddi/football matches highlights. In education, we can also use for seeing main topic in the video of 1 hr. Many photographers use the method to reduce extra or redundant frame for the video and make it attractive.

14. Conclusion

We show that Video Summarization concept are really useful in our daily life. Like in Education Section, Photography side, Entertainment side, Traffic cameras monitoring etc. The aim of video summarization is to speed up browsing of a large collection of video data, and achieve efficient access and representation of the video content. By watching the summary, users can make quick decisions on the usefulness of the video. User can also reduce is storage form large minute of video.

15. Future Work

We show that concept of Video Summarization is really useful in the coming life. In our YOLO model accuracy is less so we will try to improve the accuracy in the future. So, in current situation we are summarizing video with less activities like hand gestures (call me, stop, thumbs up, thumbs down etc.). In future we can add more activities and improve our video summarization technic.

16. References

1. ieeeexplore.ieee.org
2. M. Barbieri, L. Agnihotri, and N. Dimitrova, “Video summarization: methods and landscape”.
3. A. G. Money and H. Agius, “Video summarization: A conceptual framework and survey of the state of the art.”
4. M. Ajmal, M. H. Ashraf, M. Shakir, Y. Abbas, and F. A. Shah, “Video Summarization: Techniques and Classification
5. Dogra, Debi & Sk, Arif Ahmed Bhaskar, Harish. (2015). Smart Video Summarization using Mealy Machine based Trajectory Modelling. Multimedia Tools and Applications.10.1007/s11042-015-2576-7.
6. Video Summarization using Keyframe Extraction and Video Skimming Jadon, Shruti Jasim, Mahmood. (2019). Video Summarization. 10.13140/RG.2.2.23087.38564/1.https://www.eureka.co/python
7. https://www.github.com
8. https://www.youtube.com/watch?v=dHp5I0m9_zA
9. https://blog.roboflow.com/pp-yolo-beats-yolov4-object-detection/
10. https://www.analyticsvidhya.com/blog/2019/ 07/computer-vision-implementing-mask-r-cnn-image-segmentation/
11. https://www.quora.com/unanswered/What-is-the-difference-between-CNN-and-R-CNN
12. https://www.researchgate.net/figure/Architectur e-of-ResNet-101-network-with-Dense-Upsampling-Co nvolution-DUClayer_fig6_314115448
13. https://medium.com/@myworldsharma.jay/key-frame-extraction-from-video-9445564eb8ed
14. https://www.researchgate.net/figure/Video-2-example-frame-and-its-classification-result-with-a-near-skin-color-background_fig2_220845268
15. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection algorithms-36d53571365e
16. https://www.irjet.net/archives/V7/i6/IRJET-V7I6255.pdf
17. https://www.researchgate.net/figure/Comparison-of-Test-Time-speed-of-different-Object-detection-algorithms_fig5_350372828
18. https://www.youtube.com/watch?v=dHp5I0m9_zA

19. https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/
20. https://robocademy.com/2020/05/01/a-gentle-introduction-to-yolo-v4-for-object-detection-in-ubuntu-20-04/
21. https://blog.roboflow.com/pp-yolo-beats-yolov4-object-detection/
22. https://www.researchgate.net/figure/Video-5-example-frame-and-its-classification-result-There-are-4-different-skin-colors_fig3_220845268
23. https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_1026#:~:text=The%20aim%20of%20video%20summarization,the%20usefulness%20of%20the%20video.
24. https://www.researchgate.net/figure/Comparison-of-representative-image-object-detection-method_tbl2_330256270
25. https://pjreddie.com/media/files/papers/yolo.pdf
26. https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/

18BCE108

ORIGINALITY REPORT

16%	15%	10%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.analyticsvidhya.com Internet Source	5%
2	ddd.uab.cat Internet Source	3%
3	www.pixelsolutionz.com Internet Source	2%
4	link.springer.com Internet Source	2%
5	"Proceedings of the International e-Conference on Intelligent Systems and Signal Processing", Springer Science and Business Media LLC, 2022 Publication	1%
6	polynoe.lib.uniwa.gr Internet Source	1%
7	www.analyticssteps.com Internet Source	1%
8	mentorcruise.com Internet Source	1%

9 Gladys Jebakumari Gnanadurai, Arun Raaza, Rajendran Velayutham, Sathish Kumar Palani, Ebenezer Abishek Bramwell. "Detection of cardiac amyloidosis on electrocardiogram images using machine learning and deep learning techniques", Computational Intelligence, 2022

Publication

1 %

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		