Chapter – 2 (SSD implementation: part 2)

**Computer Vision**

**SSD: Object Detection**

Implementation

**2.8 SSD - Step 8 (TRANSFORM the inputs to feed into NN)**

We've created the SSD-NN and our "frame" is ready. Before applying the **detect()** function we need to do the transformation.

* We're going to create a new object of the **BaseTransform()** class.
* This object will do the transformation on the image so that the image is compatible with the neural network. It will make sure that the **frame** can get in to the neural network.

# *Creating the transformation*

    # *We create an object of the BaseTransform class, a class that will do the required transformations*

        # *so that the image can be the input of the neural network.*

transform = **BaseTransform**(net.size, (104/256.0, 117/256.0, 123/256.0))

* **net.size:** It is the target size of the images to feed to the neural network.
* **(104/256.0, 117/256.0, 123/256.0):** It's a triplet of three numbers that will allow to put the **color values** at the **right scale**.
* We have to input the right scale under which the SSD-NN was trained (since we're using the pre-trained SSD-NN).
* The weight was trained under some certain convention and we have to put "104/256.0, 117/256.0, 123/256.0".

**2.9 SSD - Step 9**

* In this step we'll actually open the video then we will iterate on the frames of this video to do a frame by frame detection to apply the **detect()** function on each frame of the video.
* Note that, a 2 second video can have 67 to 68 frames.
* We'll apply the **detect()** function on all 68 frames.
* Extract: First we'll get all these frames.
* Detect: Then we'll apply the **detect()** function on each of these frames to detect some *dogs*, *humans* or *other* *objects* on the frames.
* We'll will print the rectangles on each of these frames and then
* Reassemble: We will reassemble the whole frames to make a new video.
* That is the original video with the detector rectangles detecting the objects.

# *Doing some OBJECT DETECTION on a video*

reader = **imageio.get\_reader**('funny\_dog.mp4')    # *We open the video.*

fps = **reader.get\_meta\_data**()['fps']     # *We get the 'fps' frequence (frames per second).*

# *We create an output video with this 'same fps' frequence.*

writer = **imageio.get\_writer**('output.mp4', fps = fps)

**for** i, frame **in** **enumerate**(reader):      # *We 'ITERATE' on the frames of the output video:*

    # *We call our detect() function (defined above) to detect the object on the frame.*

    frame = **detect**(frame, **net.eval**(), transform)

**writer.append\_data**(frame)   # *We add the next frame in the output video.*

**print**(i)    # *We print the number of the processed frame.*

**writer.close**()  # *We close the process that handles the creation of the output video.*

* We have the frames from the video. Our neural network is ready. And we have our transform transformation. So we are ready to do some object detection on a video.
* The video is about a bouncing dog and there are humans in the background. We'll try to detect the dog and the humans.
* Open the video: The reader object is created with **imageio.get\_reader()**. reader = **imageio.get\_reader**('funny\_dog.mp4')
* **imageio** is a great library to process videos. There is another great library called **PIL**. But in this case **imageio** gives the good result.
* Frequency of the Frames (fps: frames per second): We need to get the **fps** of the video so that we can extract all the frames from the video.

fps = **reader.get\_meta\_data**()['fps']

* Output video: We'll use the fps to create an output video with same frequency. We'll use **imageio** again.

writer = **imageio.get\_writer**('output.mp4', fps = fps)

* **get\_writer**() basically creates something like an object that will contain a video.
* Detect (FOR-loop): We use a FOR-loop to detect objects for each frame of the video. On each frame we're going to apply **detect()** function and print the rectangles.
* We use enumerate to iterate the list-of-frames, because we want to print the number of processed frame.

**for** i, frame **in** **enumerate**(reader):      # *We 'ITERATE' on the frames of the output video:*

    # *We call our detect() function (defined above) to detect the object on the frame.*

    frame = **detect**(frame, **net.eval**(), transform)

**writer.append\_data**(frame)   # *We add the next frame in the output video.*

**print**(i)    # *We print the number of the processed frame.*

**writer.close**()  # *We close the process that handles the creation of the output video.*

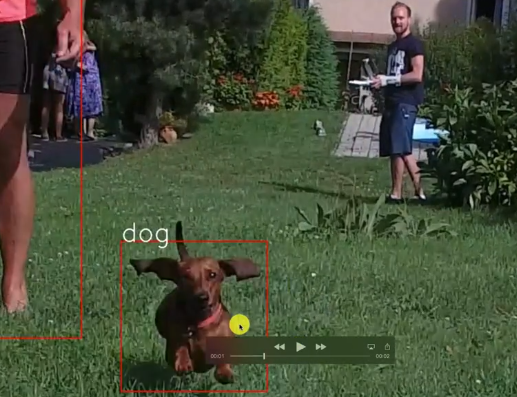
* Then we apply **detect()** method on **frame** to detect some object by the **net** (**net** is our SSD-NN). We use **net.eval()** function from the SSD-NN.
* It align with the way the **build\_ssd()** function was made. Basically **net.eval()** represents our neural network from which we get the output **y** and therefore the detections on each frame.
* We also use **transform** to make sure that this frame can be accepted into this **net**.
* Now **detect()** will return the processed frame with the detected object and we store that in a new variable called "**frame**" (we're just overwriting the old **frame** variable).
* Previous frame has no detection and new frame has detector rectangle.
* Each time we get a new process frame with the objects detected we need to append this frame to our **writer** *output video*.

**writer.append\_data**(frame)   # *We add the next frame in the output video.*

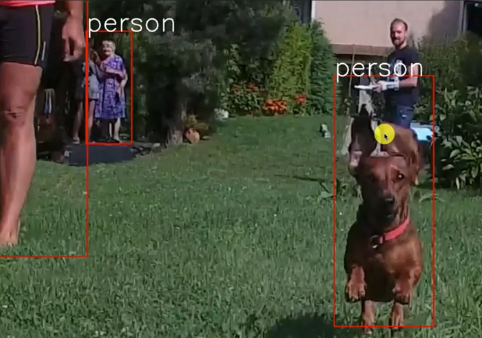
* We use print(i) just to see during the detection which frame we reached. The number of the processed frame will be displayed during the detection.
* Finally we need to close the process that manages the creation of this video.

**writer.close**()

will close the process and we'll get the output video in that same directory.



* This special frame is very interesting because it detected the humans but no dog.



* We lost the detection on the dog because the dog merged with the person behind it.
* The model things that one same person and that's why it detected the person.