International Institute of Information Technology, Hyderabad
Spring 2024 CS7.505: Computer Vision
Assignment 3: Object Detection

22 March 2024

Instructions:

- The goal of this assignment is to get familiar with object detection.
- You should upload the assignment as a jupyter notebook with appropriate cells (markdown and code) containing: (1) code that you wrote, (2) keep relevant outputs, and (3) your report and observations (markdown cells). The file should be uploaded in the courses portal.
- We recommend Python.
- You may want to use Google Colab for the second question in the assignment.
- Include the assignment number, your name, and roll number in the first cell of the notebook submission.
- **Make sure that the assignment that you submit is your own work. Any breach of this rule could result in serious actions including an F grade in the course.**
- The experiments and writing it all together can take time. Start your work early and do not wait till the deadline.

Submission: Any time before *5th Apr 2024, 23:59 IST*

1 Assignment

This assignment requires you to understand and implement object detection methods. Please use relevant libraries, do not implement from scratch. You are expected to solve both questions.

Q1: Face detection and association-based tracking [4.5 points]

1. [0.5 points] **Data preparation.** We will implement face detection and tracking on a famous scene from the movie Forrest Gump. To prepare the dataset, please download the video clip from `https://www.youtube.com/watch?v=bSMxl1V8FSg` (the mp4 at 480p resolution) and burst the first 30 seconds into frames (you should get about 719-720 frames).

   Hint 1: `https://github.com/ytdl-org/youtube-dl` is a great tool to download Youtube videos. Use `-F` flag to identify which format to download.

   Hint 2: `ffmpeg` is a wonderful tool to burst the video into frames. But you may also use `decord` or other libraries for video manipulation (be wary of different frame rates!).

2. [1.5 points] **Face detection.** Use the Viola-Jones Haar cascades based face detector from OpenCV to detect faces in each frame. How long does it take to process each frame? Identify some key factors of the algorithm that could change the time.

   Hint: you may need to look within the `xml` config file.

3. [1 point] **Face detection visualization.** Visualize the face detections made over the first 30s frames as a new video. Link to the video from your google drive. Watch the video and draw **three** conclusions about when does the face detector work or fail. Why do you think this is the case?

   Hint: You can use `cv2.rectangle` to draw boxes on the image and then save them back to disk. Then `ffmpeg` can be used again to stitch together the frames into a new video.

4. [1.5 point] **Association-based tracking.** Tracking can be used to associate face detections across time and understand that it is the same character appearing across multiple frames of the movie. We will explore a simple way to perform tracking.

   (i) Generate face tracks by comparing face detections in two consecutive frames and associating them based on IoU scores. You may want to associate faces only when IoU > 0.5. Do consider what happens when there are multiple face detections in both frames. Start new tracks for faces not seen in the previous frame. End existing tracks when faces are not visible in the next frame. How many unique tracks did you create in the first 30 seconds?

   (ii) Update the video visualization above to now include a unique track identifier (an integer number is fine), shown inside each box. Link to the video from your google drive.

   Hint: You may use `cv2.putText` to write these numbers. Make sure they are readable after stitching together the frames into a video.

   (iii) Comment about the quality of the face tracks. Do different people get associated in one track? Is a unique character associated with one unique track id? Note the timestamps of some failure cases and explain why.

Q2: YOLO Object Detection [5.5 points]

1. [0.5 points] **Data preparation.** Download a sample of the Open Images Dataset v7, specially catering to ducks `https://www.kaggle.com/datasets/haziqasajid5122/yolov8-finetuning-dataset-ducks`. We will use this to fine-tune a Yolo v8 nano detector. Unzip the archive and note how the data is stored separately in the training and validation sets.

2. [1 point] **Understanding YOLO object detector.** Study the YOLO object detector in detail. Understand and explain (in your own words, no plagiarism!) how single-shot detectors such as YOLO are different from the R-CNN series that we learned in class. There are more than 8 versions of the YOLO series. Pick any 3 and understand and explain (in your words) the differences between them.

3. [1 points] **Hands on with ultralytics.** Familiarize yourself with the ultralytics library (`https://docs.ultralytics.com/`). Learn more about the `train` and `predict` functions of the `YOLO` class.

   (i) Create a `yolov8n` (yolo-v8-nano) model from scratch. How many parameters does this model have? How many convolutional layers does it use?

   (ii) Compare the size (# parameters and # conv layers) of this model against `yolov8m` (medium) model.

   Hint: See `https://docs.ultralytics.com/models/yolov8/`.

4. [2 points] **Training YOLO variants.**

   (i) Create two versions of the training dataset. `train1` containing 100 images. `train2` containing all 400 images. You may slice the dataset randomly. Create copies of the `config.yaml` that came with the dataset zip file accordingly (point to the right directories).

   (ii) On both training dataset variants above, train three variants of the Yolo v8 models: (a) `yolov8n` initialized from scratch; (b) `yolov8n` from pretrained weights; and (c) `yolov8m` from pretrained weights.

   (iii) In total, you should have 6 "best" model checkpoints after the training. Train all models for the same number of epochs (e.g., 20). Report and compare the results (AP50) across the 6 variants on both the training and validation set. (a) How does increasing dataset size impact performance? (b) Does the bigger model perform better? Why or why not?

   (iv) Visualize the results of one of the above model variants on 4 images from the validation set. Pick your own diverse samples to qualitatively analyze how well the model performs.

5. [1 point] **Impact of augmentations.**

   (i) Does the default run use augmentations? Which ones and what do they do?

   Hint: analyze the output train logs (watch out for `albumentations`, an augmentations library).

(ii) For the `yolov8n` model with pretrained weights (trained on 100 images), remove all augmentations and compare performance before and after. Is the change significant? If yes, why? If not, why?

(iii) Which augmentation is the most important? How did you determine this?

2 Submission

We recommend that you submit a report as a single jupyter notebook with relevant cell outputs as mentioned at the top. In case you are not using Python, please share code (with instructions on how to execute) and a separate pdf report.

Submit the file in the courses / moodle portal before the deadline: **5th Apr 2024 23:59 IST**. The moodle portal may show a different date due to the grace period, do not get confused.

The report/notebook should contain:

- A description of the problem, algorithms, results and comparison of methods based on the experiments you performed.
- Challenges you faced and learnings from the experiments.

Remember, you are expected to write the complete code for the assignment yourselves. DO NOT COPY ANY PART FROM ANY SOURCE not limited to your friends, seniors or the internet.