EE 046746 - Technion - Computer Vision

Homework 3 - Segmentation

Due Date: 01.06.2020

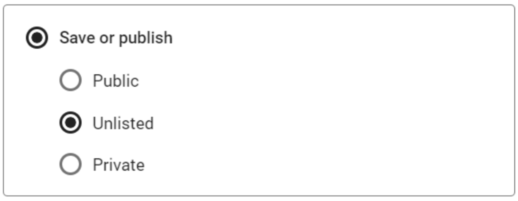
Submission Guidelines

READ THIS CAREFULLY

- Submission only in **pairs**.
- **No handwritten submissions**.
- You can choose your working environment:
  - You can work in a `Jupyter Notebook`, locally with [Anaconda (https://www.anaconda.com/distribution/)](https://www.anaconda.com/distribution/) or online on [Google Colab (https://colab.research.google.com/)](https://colab.research.google.com/)
    - Colab also supports running code on GPU, so if you don't have one, Colab is the way to go. To enable GPU on Colab, in the menu: `Runtime` $\rightarrow$ `Change Runtime Type` $\rightarrow$ `GPU`.
  - You can work in a Python IDE such as [PyCharm (https://www.jetbrains.com/pycharm/)](https://www.jetbrains.com/pycharm/) or [Visual Studio Code (https://code.visualstudio.com/)](https://code.visualstudio.com/).
    - Both also allow opening/editing Jupyter Notebooks.
- You should submit two **separated** files:
  - A compressed `.zip` file, with the name: `ee046746_hw3_id1_id2.zip` which contains:
    - A folder named `code` with all the code files inside (`.py` or `.ipynb` ONLY!), and all the files required for the code to run (your own images/videos) inside `my_data` folder.
      - **The code should run both on CPU and GPU without manual modifications**, require no special preparation and run on every computer.
    - A folder named `output` with all the output files that are not required to run the code. This includes videos and visualizations that are not included in your PDF report.
  - A report file (visualizations, discussing the results and answering the questions) in a `.pdf` format, with the name `ee046746_hw3_id1_id2.pdf`.
    - Be precise, we expect on point answers.
  - No other file-types (`.docx`, `.html`, ...) will be accepted.
- Submission on the course website (Moodle).

Video Submission

- In this exercise you are going to produce a video.
- In addition to submititng this video in the `output` folder, we also ask you to upload it to YouTube and submit the link in a different section on the website (just below the original submission section).
    - This will benefit you when you explain your expertise in Computer Vision (e.g., in a job interview).
    - We also want to make something nice with your works at the end of the course.
- If you don't want to make your video public, you can change its visibility to "Unlisted", and then it can only be accessed via link:



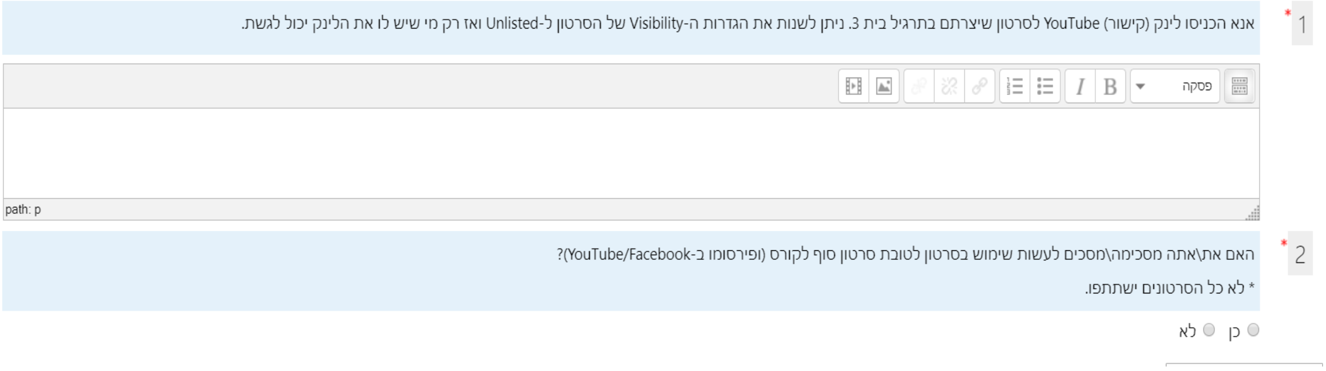
● Save or publish

○ Public

● Unlisted

○ Private

- Finally, submit the link on the course website and answer if you are okay with making this video public:



1 * אנא הכניסו לינק (קישור) YouTube לסרטון שיצרתם בתרגיל בית 3. ניתן לשנות את הגדרות ה-Visibility של הסרטון ל-Unlisted ואז רק מי שיש לו את הלינק יכול לגשת.

2 * האם את\אתה מסכימה\מסכים לעשות שימוש בסרטון לטובת סרטון סוף לקורס (ופירסומו ב-YouTube/Facebook)?

* לא כל הסרטונים ישתתפו.

○ כן ○ לא

Python Libraries

- `numpy`
- `matplotlib`
- `pytorch` (and `torchvision`)
- `opencv` (or `scikit-image`)
- `scikit-learn`
- Anything else you need (`PIL` , `os` , `pandas` , `csv` , `json` ,...)

Tasks

- In all tasks, you should document your process and results in a report file (which will be saved as `.pdf`).
- You can reference your code in the report file, but no need for actual code in this file, the code is submitted in a seprate folder as explained above.

Part 1 - Classic Vs. Deep Learning-based Semantic Segmentation

In this part you are going to compare classic methods for segmentation to deep learning-based methods. You are also going to see the effect of the background on classification.

1. Load the images in the `./data/frogs` and `./data/horses` folders and display them.
2. Pick 1 classic method for segmentation and 1 deep learning-based method and segment the given images. Display the results.
   - Briefly summarize each method you picked and discuss the advantages and disadvantages of each method. In your answer, relate to the results you received in this section.
   - Note: the classic method **must not** use any neural network.
3. Pick 3 images (download from the internet or take them yourself) that satisfy the following, and display them:
   - One image of a living being (human, animal,...).
   - One image of commonly-used object (car, chair, smartphone, glasses,...).
   - One image of not-so-commonly-used object (fire extinguisher, satellite,... **BE CREATIVE**).
4. Apply each method (one classic and one deep learning-based) on the 3 images. Display the results.
   - Which method performed better on each image? Describe your thoughts on why one method is better than the other.
5. As you probably have noticed, segmentation can be rough around the edges, i.e., the mask is not perfect and may be noisy around the edges. What can be done to fix or at least alleviate this problem? Your suggestions can be in pre-processing, inside the segmentation algorithm or in post-processing.
6. Load a pre-trained classifier (which was trained on ImageNet) from: https://pytorch.org/docs/stable/torchvision/models.html#classification (https://pytorch.org/docs/stable/torchvision/models.html#classification). In the previous assignment you used VGG16, but you can choose a different one for this assignment.
7. Pick an image of an animal in its natural habitat (e.g., cow on grass fields, zebra in the safari, ...). Display the image you chose and feed-forward it to the pre-trained network. What is the network's prediction? To convert from class index to label, use the supplied `./data/imagenet1000_clsidx_to_labels.txt` file. You can also use the supplied `./data/cow.jpg` or `./data/sheep.jpg` .
8. Segement the animal using one of the methods (classic or deep) and display the result.
9. Put the the animal in a different habitat, i.e., use the mask to place the animal on a different background. You can choose any background you want (which is not the animal's natural habitat). For example, put the cow on a beach (`./data/beach.jpg` for example) or the elaphant/sheep in a room (`./data/room.jpg` for example). Display the result.

   - You should submit the final image in the `output` folder, in addition to putting in your report.
10. Feed forward the the new image (e.g. the sheep in the room) to the pre-trained network. Was the prediction different than Q7? If so, discuss the reasons for that to happen.

Part 2 - Jurrasic Fishbach

In this part you are going to apply segmentation on a video, and integrate with other elements.

1. Film a short video of yourself (you can use your phone/webcam for that), but without too much camera movement. You on the other hand, can move -- walk/move you hands... (we expect you to). Convert the video to frames and resize the images for a reasonable not too high resolution (lower than 720p ~ 1280x720 pixles). You can use the function in `frame_video_convert.py` to help you. Display 2 frames in the report.
2. Segment yourself out of the video (frame-by-frame) using one of the methods (classic or deep). Display 2 frames in the report.
3. Pick one of the objects in the supplied videos (`./data/dancing_man_model.mp4` , `./data/dinosaur_model.mp4` , `./data/jet_model.mp4`), convert it to images and segement it out using one of the methods (classic or deep). Display 2 frames in the report. You can choose another object from: [https://pixabay.com/videos/search/green%20screen/ (https://pixabay.com/videos/search/green%20screen/)](https://pixabay.com/videos/search/green%20screen/).

   - Explain how you performed the sementation for this specific type of video (i.e., green-screen videos). Did you use a simple/classic method? Deep method? Combined both?
4. Put it all together - pick a background (can be a video or static image), put yourself (the segmented-self) and the segemented object on the background. Stitch it frame-by-frame (don't make the video too long or it will take a lot of time, 10secs maximum). Display 2 frames of the result. Convert the frames back to video. You can use the function in `frame_video_convert.py` to help you.

   - Tip: To make it look good, you can resize the images, create a mapping from pixel locations in the original image to pixels locations in the new image.
   - You should submit the final video in the `output` folder, and upload it to YouTube as instructed above.
   - We expect some creative results, this can benefit you a lot when you want to demonstrate your Computer Vision abilities.

Credits

- Images from Imagenet (http://www.image-net.org/)
- Videos from Pixabay (https://pixabay.com/videos/search/green%20screen/)
  - Dinosaur video from Modern Media (https://sites.google.com/a/sau17.net/modern-media/home/green-screen-animations)
- Icons from Icon8.com (https://icons8.com/) - https://icons8.com (https://icons8.com)