DATA.ML.300 Computer Vision
Exercise Round 3

For these exercises you will need Python and a webcam (task 3). Webcams are present in exercise classrooms during exercise sessions. Submit all your answers and figures included in a PDF file. In addition, submit runnable .py files, where you have filled in your codes to the given template files. Do not copy all the code from these runnable files into the PDF.

All submissions should be uploaded to Moodle. Exercise points will be granted after a teaching assistant has reviewed your answers. Submissions made before the deadline can earn up to 4 points. After the deadline, no partial points will be awarded; only submissions with fully correct solutions to all tasks will receive 1 point.

Load Python.zip containing all the necessary data/code for the following tasks.

Before continuing you have to install OpenCV library for Python. For TC303 computers, open command prompt by searching *cmd* and use the command below to install the package (note that there are two dashes before *user* and *upgrade*).

```
pip install --user --upgrade opencv-python
```

Task 3 also require Tensorflow. Everything should be installed for TC303, but if you'd like to install them to your own machine check the instructions here. Currently, the code works at least on python 3.10 and 3.11 with tensorflow 2.13.0 and tensorflow-estimator 2.13.0.

Task 1. HOG descriptors for people detection. (Programming exercise) (1 point)
We'll start by implementing a simple people detector using Histogram of Oriented Gradients as descriptor for our detector. Open hog_detector.py and follow the instructions written in the comments. **Include at least two screenshots in your PDF file, and also return your version of hog_detector.py**

Task 2. Basics of SSD object detector. (Programming exercise) (1 point)
We'll be looking at a CNN based object detector, Single Shot MultiBox Detector (SSD). The goal of this task is to learn the basics of SSD and deep learning implementation in Python.

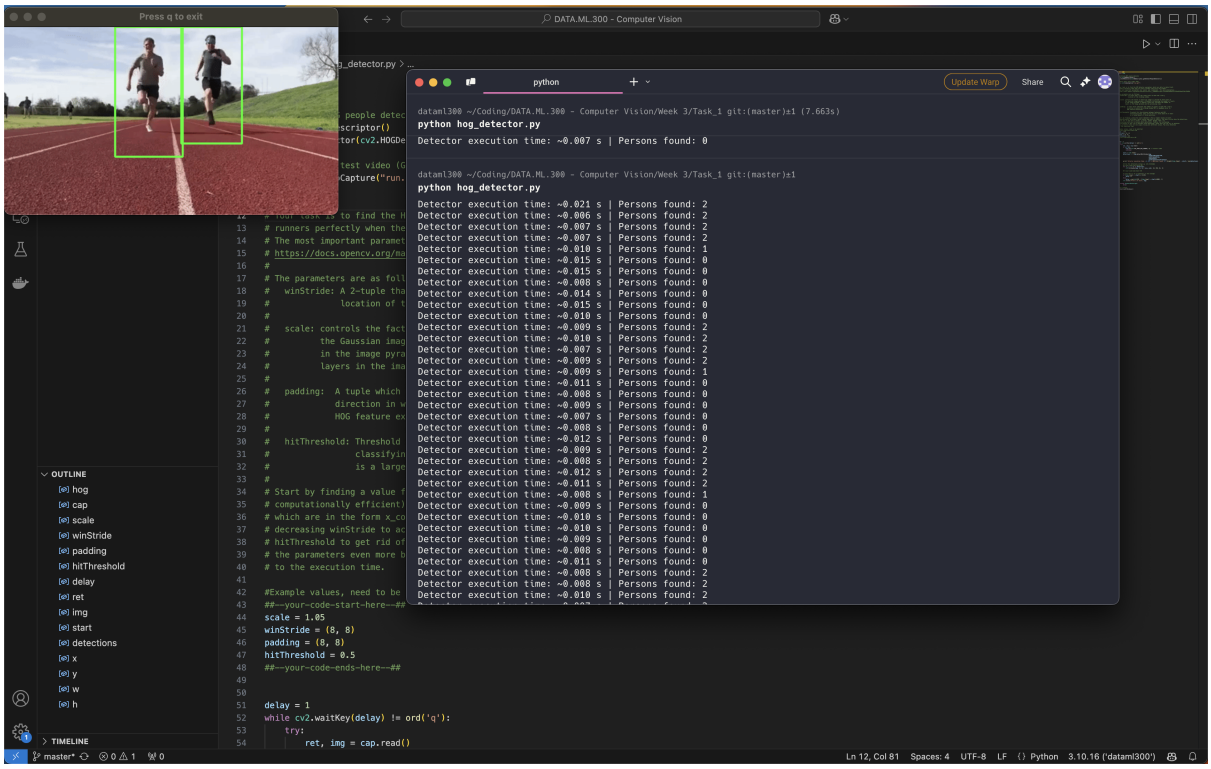

Figure 1: Task 1

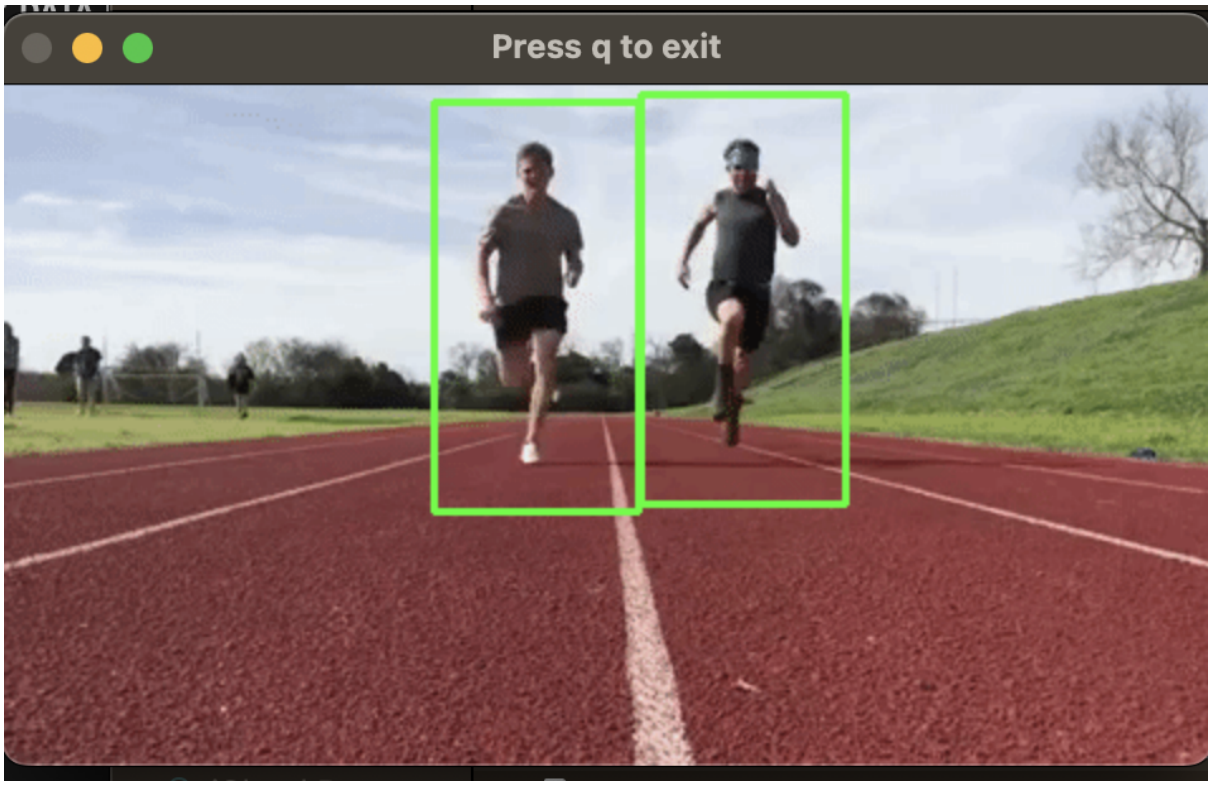

Figure 2: Task 1

Open the SSD exercise folder, follow the steps below and write down your observations. **Include your answers in your PDF file.**

1. We are using (a small portion of) the Udacity road traffic dataset as an example, where the target objects are vehicles and traffic lights. The dataset can be found in the *datasets* directory and the target values can be found in the .csv files. What is the form the targets are presented in? What is the difference between training and validation datasets in a general sense?

   The targets are presented as xmin, xmax, ymin, ymax and the respective class of the detected object. With this information we can draw appropriate boxes for the detected objects. Training dataset is the data we will train the model on. Essentially the model would learn a representation of the training data. Validation data is data that the model has not seen during the training process and we can use this to "validate" if the model is working on unseen data, meaning that has the model learned to generalize or has it overfit to the training data.

2. Next we'll take a look at the general architecture of the model. The keras_ssd7.py file implements a smaller version of the SSD detector. Open keras_ssd7.py under the *models*-directory, and locate the *build_model* function. Try to find where the first convolutional part (before the convolutional predictor layers) of the network is defined. How many convolutional "blocks" are there, and what kind of layers is each block build from?

   It seems like there are 7 convolutional blocks. Each of them contains a convolutional layer, a normalization layer, an exponential linear unit (ELU), and a max pooling layer, except the final convolutional block, which does not have the max pooling layer.

3. SSD has it's own loss function, defined in chapter 2.2 in the original publication. What are the two attributes this loss function observes? How are these defined (short explanation without any formulas is sufficient)? The publication can be found in the exercise folder or here.

   The loss function optimizes for 1) detecting objects and 2) detecting the classes of the objects. The first part, localization loss basically indicates the difference between the detection box parameters to the ground truth box parameters. The second part is a confidence in the class of the detected object, basically a classification loss, which is handled with a softmax over the multiple class confidences.

Task 3. SSD300 for real-time object detection. (Programming exercise) (2 points)
This time we'll be using a larger version of SSD with pretrained weights to implement real-time object detection for webcam feed. Webcams can be found in computer classroom during exercise sessions. **Download the pretrained weights here and save them to the *weights folder*.** Follow the instructions in ssd300_webcam.py and use the OpenCV documentation to find out how certain functions work. **Include at least two screenshots of the webcam feed with a detected object(s) in your PDF, e.g. a monitor or a chair. Also return your version of ssd300_webcam.py**

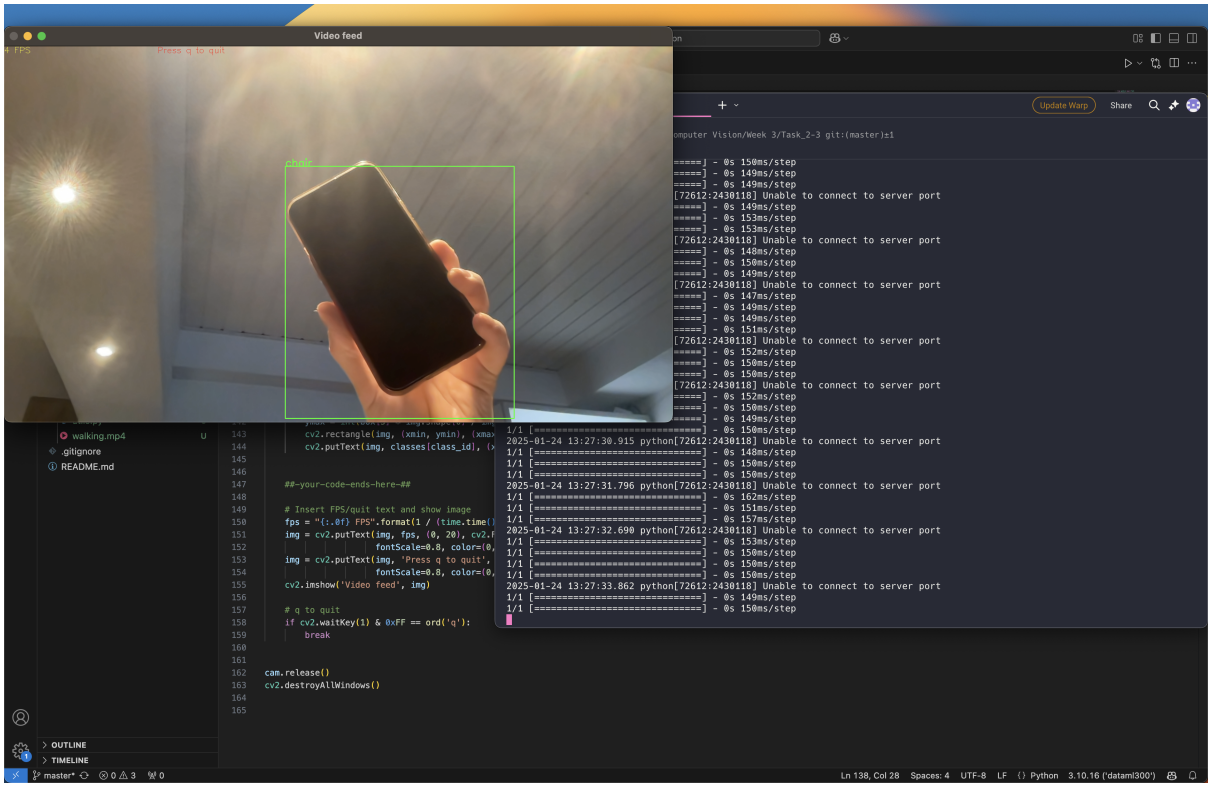

Figure 3: Task 3: Chair

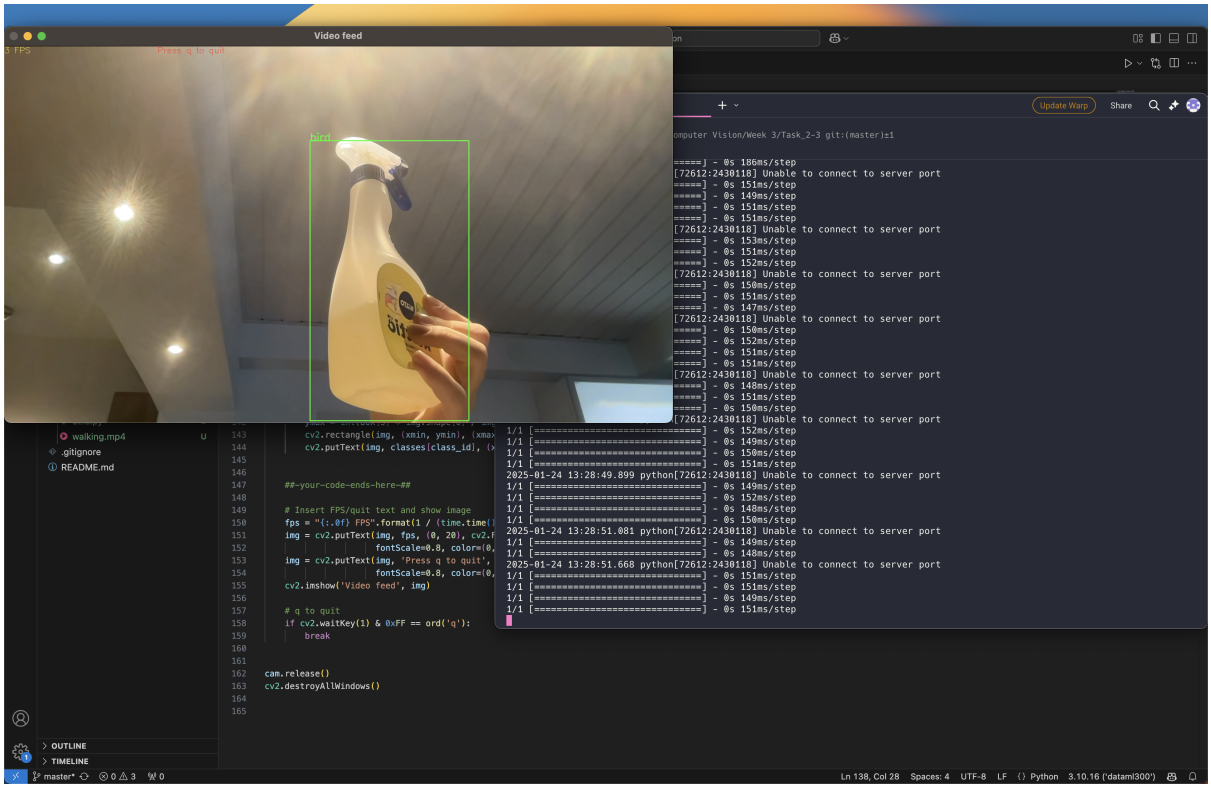

Figure 4: Task 3: Bird

If you are interested, you can also try the previous HOG detector for webcam feed.