**Object detection exercise**

This exercise will introduce you to one of the main tasks in the *computer vision* domain – **object detection**. You will learn the problem formulation, the metrics, and leading approaches, but more apparently you will learn a highly useful skill – taking an existing approach (and git repo) and making it work for you.

Object detection task

Get yourself familiar with the object detection task (through online blogs and such), without focusing on a specific approach for now, see that you know how to answer the following questions:

1. What is the input and output of the model? What do the annotations look like?
2. What is the commonly used evaluation metric? What does it mean and how is it calculated? (be ready to discuss this with your instructor)
3. **Bonus:** Name some leading approaches, appropriate datasets, and similar tasks.

SVHN dataset

We are going to use the Street View House Numbers (SVHN) dataset. Download the data from the following [link](http://ufldl.stanford.edu/housenumbers/). We suggest using the annotations from [this](https://github.com/penny4860/svhn-voc-annotation-format) Github repo (They are in an easier format to use). Perform some exploratory data analysis (EDA), to get some insight into the data, some questions you might be interested in are:

* How many images are there? (train and test)
* What is the class distribution?
* What is the distribution of the number of objects in each image?
* Standard object size and location.
* Many more…

Put these all in an organised notebook for future use and to go over with your instructor.

YOLO

For the task of object detection on the SVHN dataset we will be using YOLO (You Only Look Once) by Joseph Redmon. Before anything else, see his [resume](https://pjreddie.com/static/Redmon%20Resume.pdf). As you can see, he is a bit of a troll (as seen in his papers as well), but his methods are amongst leading approaches for real-time object detection. Yolo had several improvements spanning more than 3 papers. Get to know yolo-V3 (by reading the paper(s), blog posts or both). Look also for answers to the following questions:

* What is the model’s output?
* Which post processing is performed on the output to receive detections? Which Non-Maximum Suppression (NMS) methods are there?
* Describe the model architecture and the training stage

Ultarlytics – YOLO

We will now focus on a specific implementation of YOLO, you are welcome to use any of the following 2: [Yolov3](https://github.com/ultralytics/yolov3) or [Yolov5](https://github.com/ultralytics/yolov5). These are implementations of yolo variants which contain some improvements over what is described in the paper.

1. Set up an environment and clone the git repository to your working directory.
2. Recommended: Look at and run some of the examples given for inference, fine-tuning, transfer learning on the described datasets.
3. Now you are required to train and evaluate the yolo model on the SVHN dataset, this may require formatting the labels, tweaking the model output shape and others. **DO NOT SPEND TIME TWEAKING HYPERPARAMETERS!!!**
4. **Bonus**: Add an additional metric which is relevant for your specific task of house numbers (maybe something that treats all the objects in an image together?)
5. **Bonus 2**: perform some *false analysis* – on which samples is the model performance worse? Do they have something in common?
6. **Bonus 3**: Introduce some improvement to help results from augmentations, optimization tricks, ensembles, test time augmentations etc.

**BONUS**

We treated the task of recognising numbers in images as an object detection task, yet this is generally called OCR (Optical Character Recognition) and can be solved by many approaches. Read a bit about this task (task definition, metrics, leading approaches etc.).

For a project developed by members of our unit regarding COVID-19, they combined several OCR methods to achieve high performance. The following [git](https://github.com/clovaai/deep-text-recognition-benchmark) holds all the required models for their approach. Use the git to perform inference on SVHN (and maybe train). Analyse the performance and compare to YOLO.