Final project report

The following steps describe how to run my project through Colab, YOLOv8 and Roboflow:

Important sidenote: I misunderstood the naming convention on the annotation server, so I named my group/submission 'henrilh' my ntnu alias, instead of 'group 195'.

The following Colab notebook can be used to train a custom YOLOv8 model:

https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolov8-object-detection-on-custom-dataset.ipynb

At step 5 in this tutorial, you need to input the details from my dataset, which can be found here:

https://universe.roboflow.com/henriklatsch-gmail-com/norwayjapan/dataset/1 (link will be deactivated shortly after project is finished)

Generate a download code for yolov5/yolov8 format in this link, and paste into the appropriate location in the original notebook.

In the «custom training» part of the Colab notebook, select image size = 640, epochs = 100, final learning rate (lrf) = 0.001.

Ideally, you should connect the Colab instance to a google drive or similar, to continuously save models during training (not described in the basic tutorial notebook link).

The final output should look something like this:

```
Validating runs/detect/train23/weights/best.pt...
Ultralytics YOLOv8.0.20 🚀 Python-3.10.11 torch-2.0.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 268 layers, 43609692 parameters, 0 gradients, 164.8 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 57/57 [00:39<00:00,  1.43it/s]
                   all       1814       2478      0.478      0.357       0.37      0.183
           D00rotation       1814       1842      0.504      0.463      0.443      0.235
           D10rotation       1814        377      0.456      0.292      0.305      0.117
           D20rotation       1814        160      0.479       0.45       0.47      0.269
           D40rotation       1814         99      0.475      0.222      0.264      0.111
Speed: 0.3ms pre-process, 16.0ms inference, 0.0ms loss, 1.7ms post-process per image
```

Inference is described at a high level (see **inference.ipynb** for implementation):

- Pip install ultralytics, roboflow, PyTorch, Torchvision (and necessary dependencies)
- Crop test images to x_min-x2500, y_1000-y_max
- Stretch test images to square of long_side * long_side
- Run inference on stretched test images
- Scale back detected bounding boxes by a factor of approximately 0.39 (slightly image dependent, solve in code)
- Add 1000 px to each y_location of bounding boxes