Finetuning a pretrained YOLO detection model with Bboxes only:

Imports:

```python
import os
from datetime import datetime
import pytz
#import tensorflow as tf
```

Some config:

```python
setupnumber = '_alle' # ONLY for saving the model! <-- TODO: cp noch pythonisieren? :-)

modeltype = "yolov8n.pt"
# Other models:
# "yolov8m-seg.pt"
# "yolov8s.pt"

# Train on bounding boxes or segmentations?
#labeltype = "boxes"
labeltype = 'segs'
if labeltype == 'segs':
    modeltype = modeltype.replace(".pt", "-seg.pt")

# Bigger batchsizes for smaller models:
batchsize = 10
if modeltype == "yolov8s.pt":
    batchsize = 40
if modeltype == "yolov8n.pt":
    batchsize = 120
if modeltype == "yolov8n-seg.pt":
    batchsize = 100

imagesize = 640 # 1280
if imagesize == 1280:
    batchsize = batchsize // 4

N_epochs = 300

localtime = datetime.now(pytz.timezone('Europe/Berlin'))
starttime = localtime.strftime("%Y-%m-%d_%H%M")
savestring = "Training date = "+starttime
savestring += "\nSetup = "+setupnumber
savestring += "\nModel = "+modeltype
savestring += "\nLabeltype = "+labeltype
savestring += "\nBatchsize = "+ str(batchsize)
savestring += "\nImagesize = "+ str(imagesize)
savestring += "\nEpochs = " + str(N_epochs)
print(savestring)

```

```
Training date = 2024-09-28_2146
Setup = _alle
Model = yolov8n-seg.pt
Labeltype = segs
Batchsize = 120
Imagesize = 640
Epochs = 300
```

Fetch the data:

```python
# Mount Google Drive and copy data:
if 'data_loaded' not in globals():
    from google.colab import drive
    drive.mount('/content/drive')
    !mkdir -v /content/Data

    !cp -r /content/drive/MyDrive/setup_015/* /content/Data/ #| pv
    print("setup015 copied")
    !cp -r /content/drive/MyDrive/setup_002/* /content/Data/
    print("setup002 copied")
    !cp -r /content/drive/MyDrive/setup_003/* /content/Data/
    print("setup003 copied")
    !cp -r /content/drive/MyDrive/setup_006/* /content/Data/
    print("setup006 copied")
    !cp -v /content/drive/MyDrive/setup-002-standalone.yaml /content/Data/

    print("Data download finished at", datetime.now(pytz.timezone('Europe/Berlin')).strftime("%H:%M"))

```

```
Mounted at /content/drive
mkdir: created directory '/content/Data'
setup015 copied
setup002 copied
setup003 copied
setup006 copied
'/content/drive/MyDrive/setup-002-standalone.yaml' -> '/content/Data/setup-002-standalone.yaml'
Data download finished at 22:01
```

Fetch helper scripts/functions and restructure the data:

```
if 'data_loaded' not in globals():
    !cp -v /content/drive/MyDrive/prepare_yolo_data_folders.py /content/
    # Download helper script from Github: -> doesn't work.
    #!wget https://raw.githubusercontent.com/<username>/<reponame>/<branch>/prepare_yolo_data_folders.py
    from prepare_yolo_data_folders import prepare_yolo_data_folders

    # Colab magic to make "Data" the current directory:
    %cd /content/Data

    #Do the train-validation split and move the image and label files to YOLO folder structure:
    data_loaded = prepare_yolo_data_folders(labeltype) # TODO: Quiet! :-)
    print('data_loaded is', data_loaded)
```

```
'/content/drive/MyDrive/prepare_yolo_data_folders.py' -> '/content/prepare_yolo_data_folders.py'
/content/Data
Preparing files and folders for training with segs labels.
Current folder is: /content/Data
Found 354 labels.
Found 354 labelled images.
Created train/images and train/labels.
Created val/images and val/labels.
Created test/images and test/labels.
Moved 247 images and labels to folders.
Moved 71 images and labels to folders.
Moved 36 images and labels to folders.
Data handling finished.
data_loaded is True
```

```
# Count the files:
num_train = len(os.listdir('/content/Data/train/labels'))
num_val = len(os.listdir('/content/Data/val/labels'))
num_test = len(os.listdir('/content/Data/test/labels'))
print(f"Number of training images: {num_train}")
print(f"Number of validation images: {num_val}")
print(f"Number of test images: {num_test}")

savestring += "\nNumber of training images: " + str(num_train)
savestring += "\nNumber of validation images: " + str(num_val)
savestring += "\nNumber of test images: " + str(num_test)
```

```
Number of training images: 247
Number of validation images: 71
Number of test images: 36
```

```
# Create model directory on Drive:
modeldir = '/content/drive/MyDrive/CucumberModels/'+starttime+"_"+labeltype
if not os.path.exists(modeldir):
    os.makedirs(modeldir)

savestring += "\nModel directory: " + modeldir
```

Write settings file:

```
# Output settings file:
settingsfile = modeldir + "_settings.txt"
with open(settingsfile, "w") as file:
    file.write(savestring)
print("Model settings written to", settingsfile)
```

```
Model settings written to /content/drive/MyDrive/CucumberModels/2024-09-28_2146_settings.txt
```

Install ultralytics and import YOLO:

```
%pip -q install ultralytics
from ultralytics import YOLO
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 875.1/875.1 kB 49.2 MB/s eta 0:00:00
Creating new Ultralytics Settings v0.0.6 file ✅
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytics/settings.json'
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir'. For help see https://docs.ult
```

Okay, so the images and labels are there? Next: Import YOLO model for training! Note: Check Bboxes vs. segmentations!

Load tensorboard:

```
%load_ext tensorboard
```

Import YOLO model for training

```
# Load a pretrained model:
model = YOLO(modeltype)
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v8.2.0/yolov8n-seg.pt to 'yolov8n-seg.pt'...
100%|██████████| 6.74M/6.74M [00:00<00:00, 379MB/s]
```

YAML file for folder and class definition: Everything ok?

```
#!cat /content/misc/setup-002-standalone.yaml
yamlfile = "/content/Data/setup-002-standalone.yaml"
```

Do the training!

```
# If GPU-RAM is full before training:

# import torch
# torch.cuda.empty_cache()

```

```
# Train the model
trainresults = model.train(data=yamlfile, epochs=N_epochs, imgsz=imagesize, batch=batchsize)
```

```
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     56/300      20.9G     0.7551      1.026      0.515     0.9179         86        640: 100%|██████████| 3/3 [00:09<
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mA

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     57/300      22.8G     0.7333     0.9395     0.5089     0.8963        114        640: 100%|██████████| 3/3 [00:09<
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mA

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     58/300      22.6G     0.7973      1.003     0.5384     0.9349         84        640: 100%|██████████| 3/3 [00:04<
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mA

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     59/300      20.8G     0.7356     0.9242     0.4935     0.9118         93        640: 100%|██████████| 3/3 [00:04<
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mA

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
     60/300      21.8G      0.743     0.9185      0.509     0.9054        128        640: 100%|██████████| 3/3 [00:05<
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mA
```

Save the model directly to Drive:

```python
model.save(modeldir+"/"+starttime+"_"+labeltype+".pt")
```

wandb?

... and managing the parameter "matrix"

Tensorboard:

[ ] ↳ 1 Zelle ausgeblendet

Predict the cucumbers on test images:

```python
# Examples:
# #predictresults = model(["image1.jpg", "image2.jpg"])  # return a list of Results objects
# # Run inference on 'bus.jpg' with arguments
# model.predict("bus.jpg", save=True, imgsz=320, conf=0.5)


# /content/Data/test/images

# First, I guess we should _dynamically_?! define a list of images to predict on:
#imagelist = []
imagelist = ['test/images/'+file for file in os.listdir("/content/Data/test/images")]
print(f"Using {len(imagelist)} images for prediction: ")
print(imagelist)

# Then, we can predict on the images:
results = model.predict(imagelist,
                        imgsz=imagesize,
                        conf=0.6,
                        #show=True,
                        save=True)

for index, result in enumerate(results):
  result.save(modeldir+"/"+starttime+"_prediction_"+str(index)+".jpg")

# # Process result:
# # for result in results:
# boxes = result.boxes  # Boxes object for bounding box outputs
# # masks = result.masks  # Masks object for segmentation masks outputs
# # keypoints = result.keypoints  # Keypoints object for pose outputs
# probs = result.probs  # Probs object for classification outputs
# obb = result.obb  # Oriented boxes object for OBB outputs
# result.show()  # display to screen
```

```
Using 36 images for prediction:
['test/images/dsc01853.png', 'test/images/dsc01868.png', 'test/images/20220614_131249.png', 'test/images/dsc02013.png',

0: 1280x1280 6 cucumbers, 12.0ms
1: 1280x1280 2 cucumbers, 12.0ms
2: 1280x1280 28 cucumbers, 12.0ms
3: 1280x1280 8 cucumbers, 12.0ms
4: 1280x1280 9 cucumbers, 12.0ms
5: 1280x1280 21 cucumbers, 12.0ms
6: 1280x1280 8 cucumbers, 12.0ms
7: 1280x1280 3 cucumbers, 12.0ms
8: 1280x1280 3 cucumbers, 12.0ms
9: 1280x1280 8 cucumbers, 12.0ms
10: 1280x1280 15 cucumbers, 12.0ms
11: 1280x1280 11 cucumbers, 12.0ms
12: 1280x1280 1 cucumber, 12.0ms
13: 1280x1280 7 cucumbers, 12.0ms
14: 1280x1280 19 cucumbers, 12.0ms
15: 1280x1280 3 cucumbers, 12.0ms
16: 1280x1280 19 cucumbers, 12.0ms
17: 1280x1280 6 cucumbers, 12.0ms
18: 1280x1280 6 cucumbers, 12.0ms
19: 1280x1280 7 cucumbers, 12.0ms
20: 1280x1280 15 cucumbers, 12.0ms
21: 1280x1280 26 cucumbers, 12.0ms
22: 1280x1280 16 cucumbers, 12.0ms
23: 1280x1280 10 cucumbers, 12.0ms
24: 1280x1280 23 cucumbers, 12.0ms
25: 1280x1280 7 cucumbers, 12.0ms
26: 1280x1280 6 cucumbers, 12.0ms
27: 1280x1280 11 cucumbers, 12.0ms
28: 1280x1280 8 cucumbers, 12.0ms
29: 1280x1280 2 cucumbers, 12.0ms
30: 1280x1280 2 cucumbers, 12.0ms
31: 1280x1280 7 cucumbers, 12.0ms
32: 1280x1280 11 cucumbers, 12.0ms
33: 1280x1280 2 cucumbers, 12.0ms
34: 1280x1280 14 cucumbers, 12.0ms
35: 1280x1280 15 cucumbers, 12.0ms
Speed: 166.1ms preprocess, 12.0ms inference, 12.3ms postprocess per image at shape (1, 3, 1280, 1280)
Results saved to runs/segment/train2
Where are the resulting pictures stored? :-)
And how can I rescale them for/after prediction?
```

Show resulting pictures:

Perhaps replace IPython display by PIL stuff?

```
#from IPython.display import Image
#Image(filename='/content/runs/detect/train53/20220614_134218_Gurkenfoto_2.jpg', width=400)

```

```
#Image(filename='/content/runs/detect/train2/20220614_134218_Gurkenfoto_rescaled_640.jpg', width=400)

```

```
#Image(filename='/content/runs/detect/train53/dsc01843.png', width=400)
```