VISUM

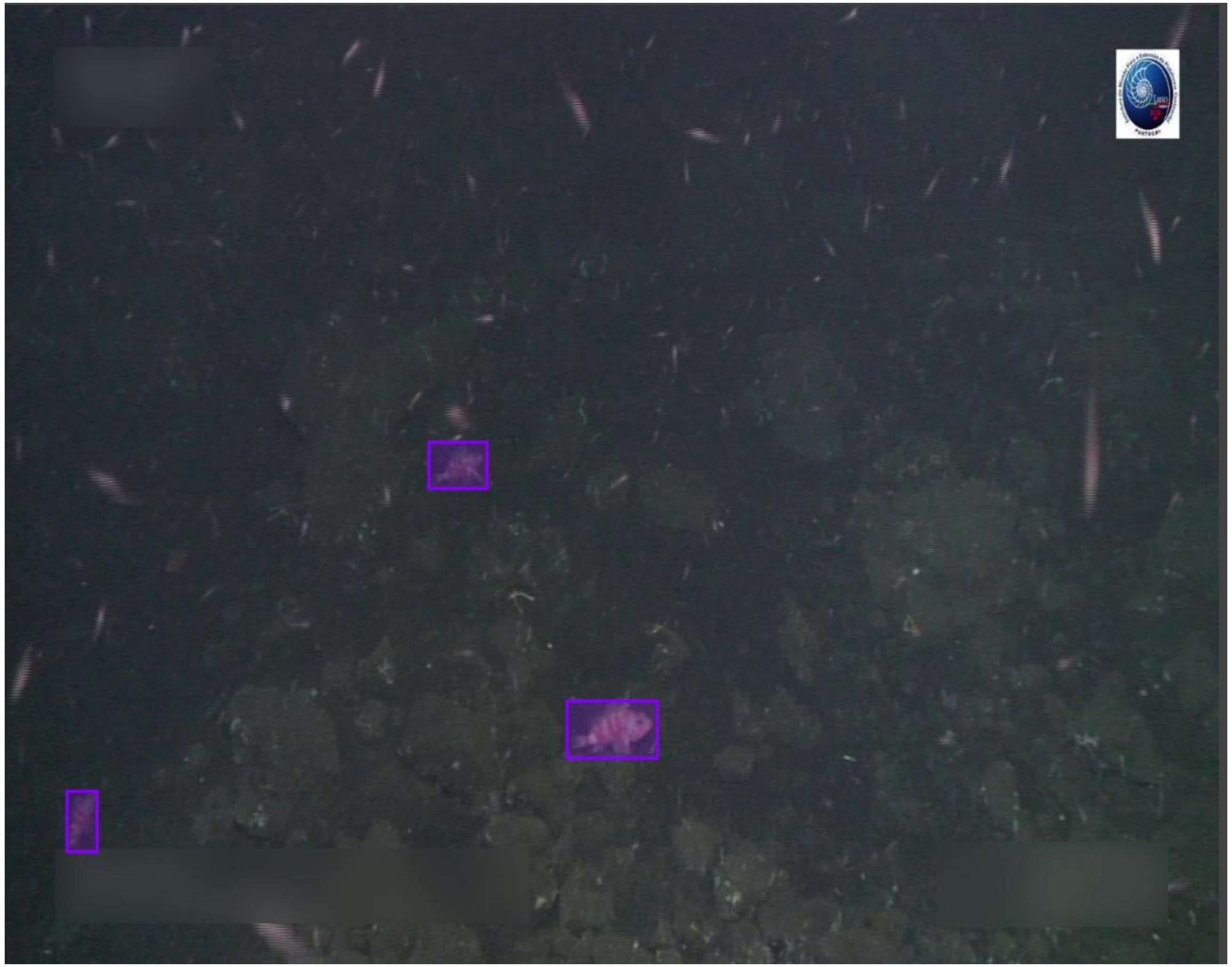
VISUM2020 PROJECT

VCMI VISUAL COMPUTING & MACHINE INTELLIGENCE RESEARCH GROUP

INESCTEC

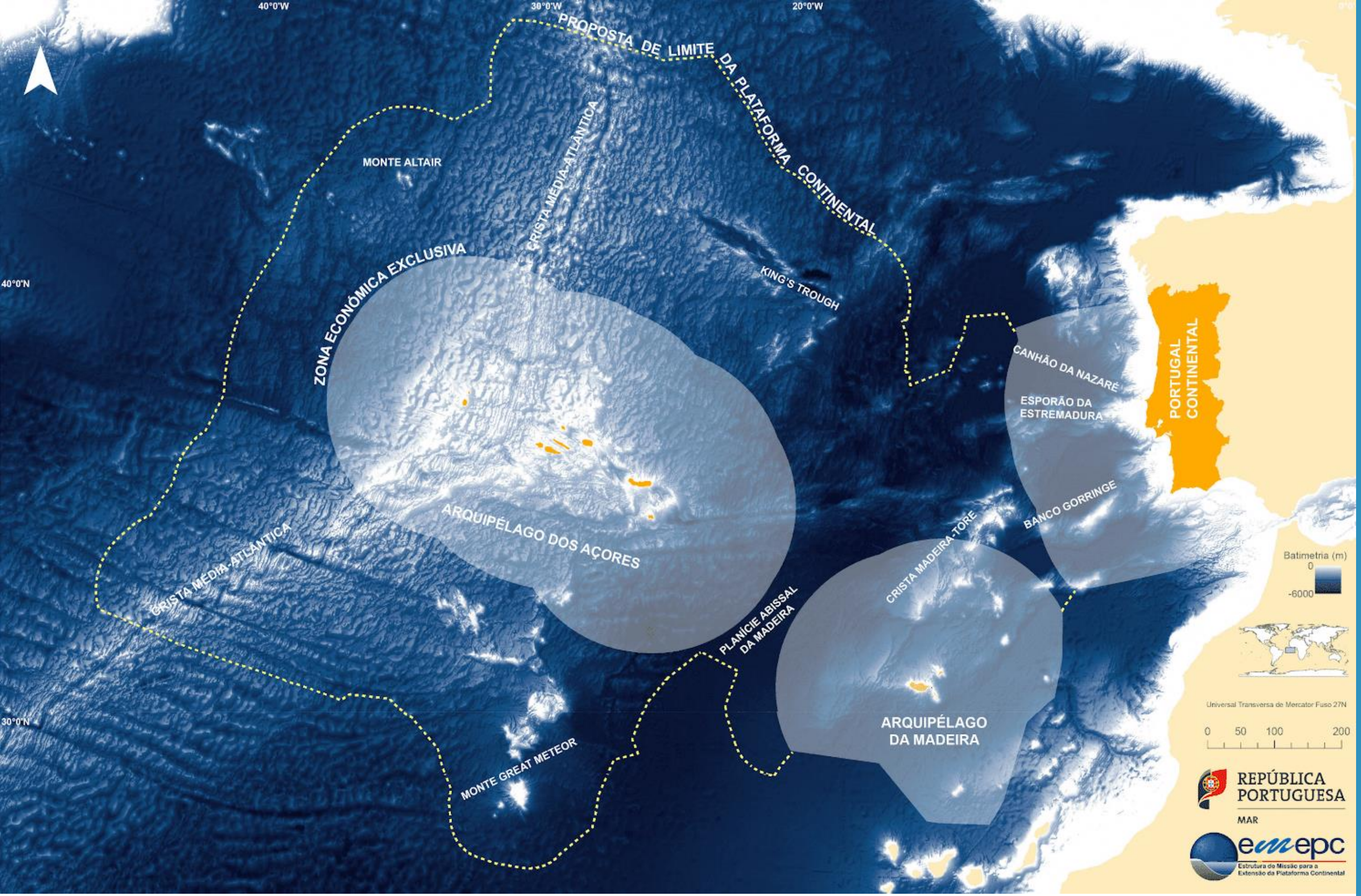
abyssal

Fish Detection in Underwater Images



In collaboration with: abyssal

EMEPC



- Dataset was obtained by EMEPC
- They explore the Portuguese continental shelf
- Scientific work: geology and biology
- Close to 1000 hours of operation

[redacted]
[redacted]
Estrutura de Missão Para a Extensão da Plataforma Continental
Luso
PORTUGAL
emepc
Estrutura de Missão para a Extensão da Plataforma Continental

Why is it important to EMEPC?

Marine Biologists need to see the video to:

- Count fish
- Classify species

Less than 1% of the video frames shows fish!

Why is it important to Abyssal?

- Offshore aquaculture
- Environmental studies
  - Before building a new offshore field (*i.e.* offshore wind farm)
  - Before decommissioning an old field
- Keep collaboration with EMEPC

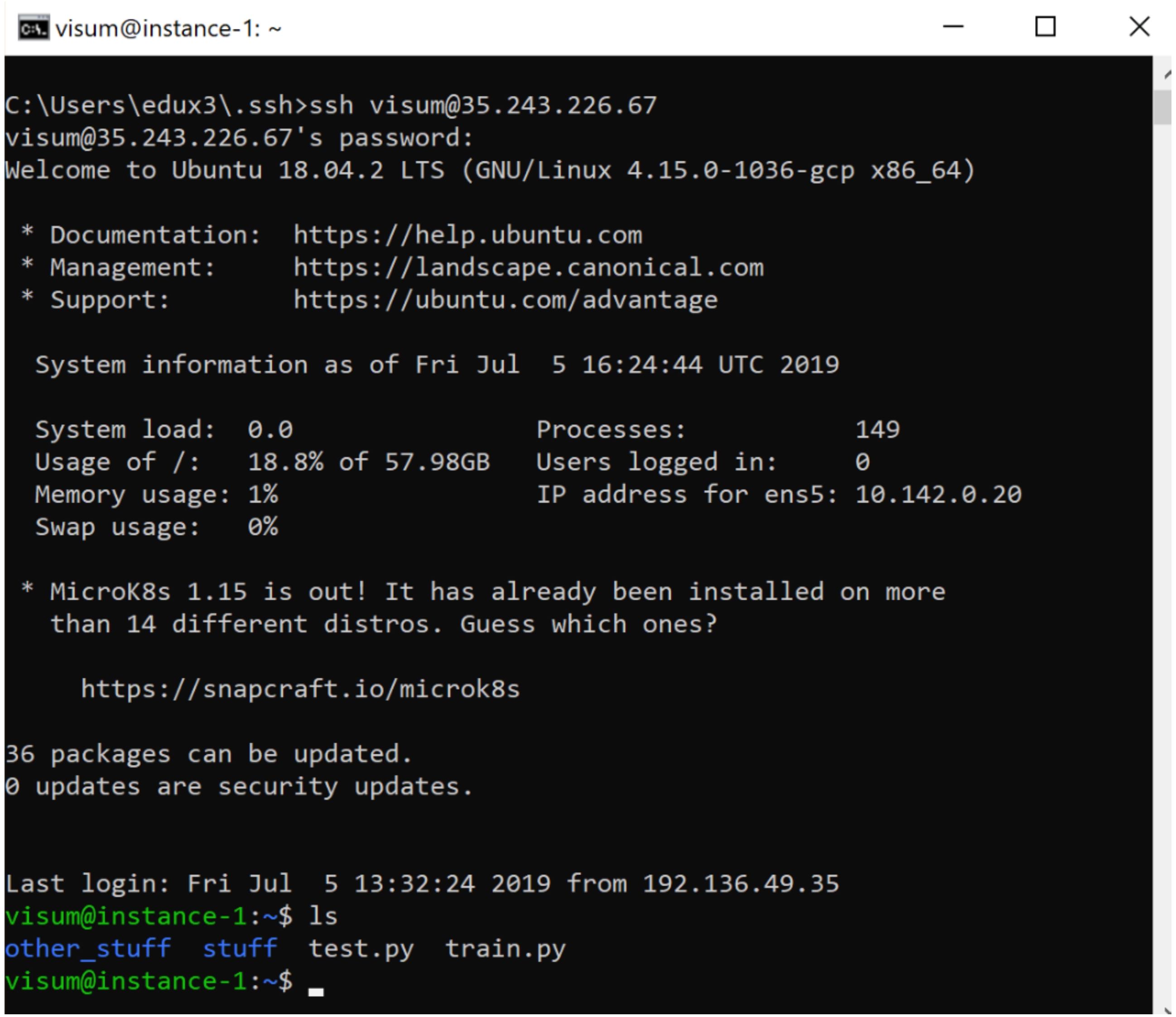
Resources

Each team will have access to a Google Cloud machine:

- NVIDIA Tesla K80
- 8 vCPU
- 30 GB RAM
- 60 GB Hard Disk

Using your machine

Connecting

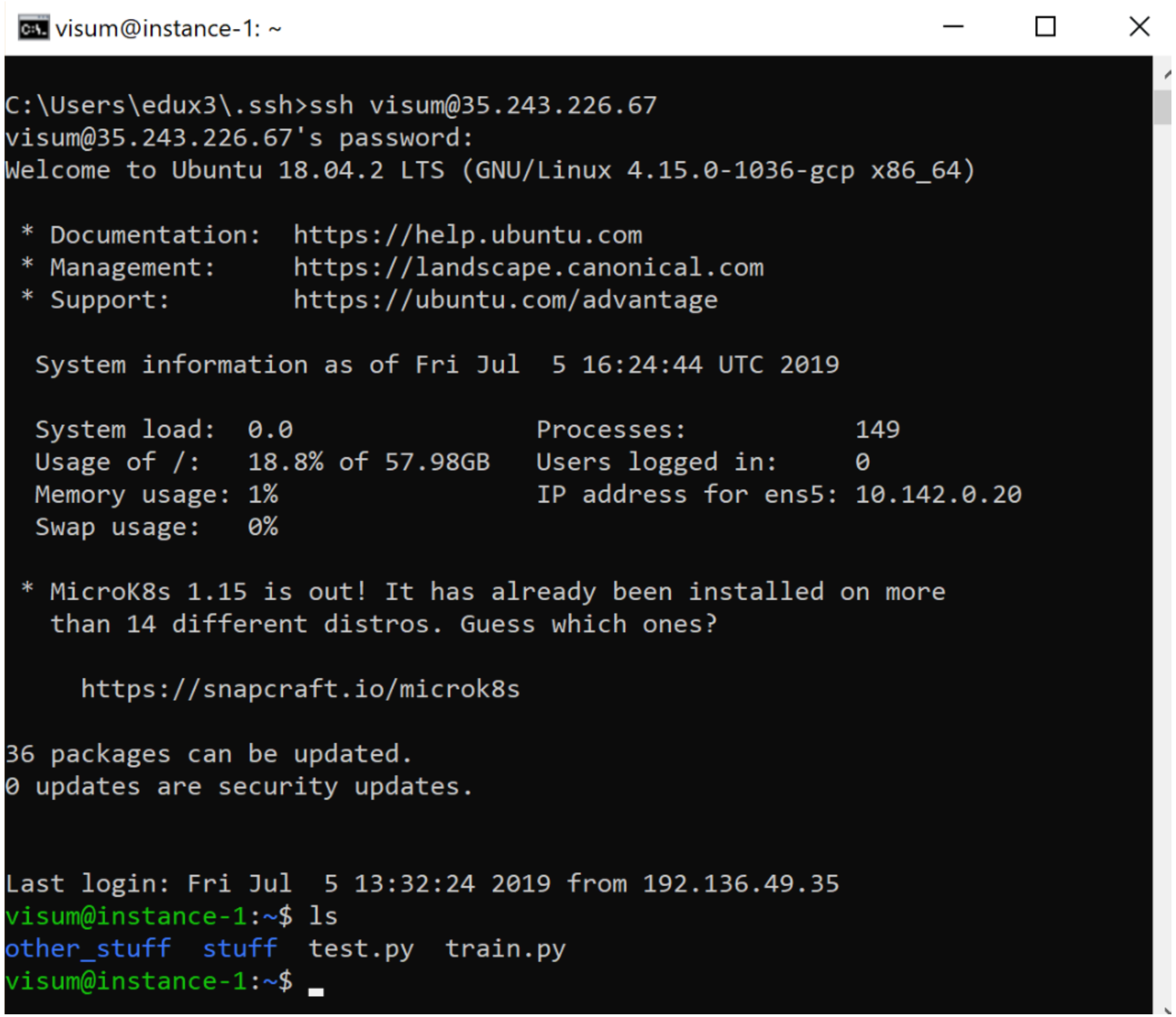


```
C:\Users\edux3\.ssh>ssh visum@35.243.226.67
visum@35.243.226.67's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-1036-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jul  5 16:24:44 UTC 2019

  System load:  0.0                Processes:           149
  Usage of /:   18.8% of 57.98GB   Users logged in:     0
  Memory usage: 1%                 IP address for ens5: 10.142.0.20
  Swap usage:   0%

 * MicroK8s 1.15 is out! It has already been installed on more
   than 14 different distros. Guess which ones?

     https://snapcraft.io/microk8s

36 packages can be updated.
0 updates are security updates.


Last login: Fri Jul  5 13:32:24 2019 from 192.136.49.35
visum@instance-1:~$ ls
other_stuff  stuff  test.py  train.py
visum@instance-1:~$ _
```

- ssh visum@<ip_address>
  - User: visum
  - Password: ********
- **All the users share the same account;**
- **There is no GUI.**

Using your machine

Editing files



visum@instance-1: ~

```
C:\Users\edux3\.ssh>ssh visum@35.243.226.67
visum@35.243.226.67's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-1036-gcp x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jul  5 16:24:44 UTC 2019

  System load:  0.0                Processes:           149
  Usage of /:   18.8% of 57.98GB   Users logged in:     0
  Memory usage: 1%                 IP address for ens5: 10.142.0.20
  Swap usage:   0%

 * MicroK8s 1.15 is out! It has already been installed on more
   than 14 different distros. Guess which ones?

     https://snapcraft.io/microk8s

36 packages can be updated.
0 updates are security updates.


Last login: Fri Jul  5 13:32:24 2019 from 192.136.49.35
visum@instance-1:~$ ls
other_stuff  stuff  test.py  train.py
visum@instance-1:~$ 
```

- You may map your machine's /home to a folder in your computer.
  - This will allow you to use fancy text editors, view images and plot results

sshfs visum@:/home/visum ~Desktop/visum

The Dataset

TRAIN

- 84 sequences of 60 frames each;
- Some images contain one or more fish;
- Some images do not contain any fish.

/home/master/dataset/train/seq000/img0.jpg

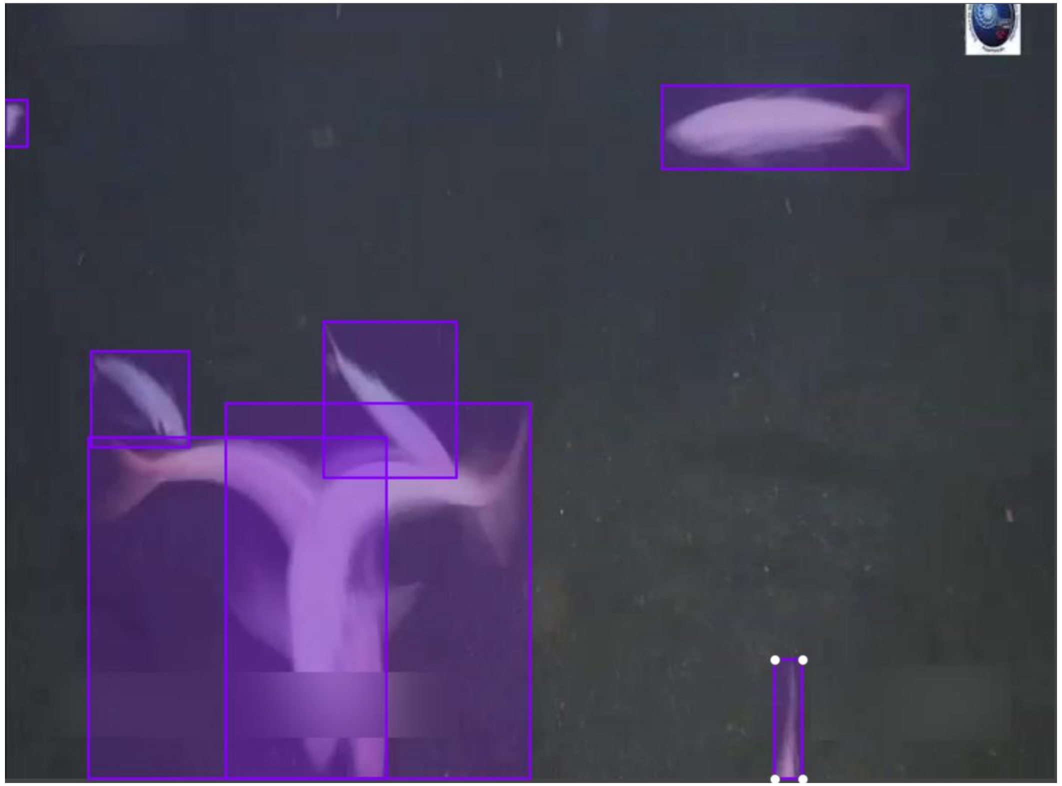
TEST (NO ACCESS)

- 30 sequences of 60 frames each;
- Some images contain one or more fish;
- Some images do not contain any fish;
- Daily test: 10 sequences of 60 frames each.

Annotations

CSV File

```
1.  sequence; frame; [(x_min, y_min, x_max, y_max), (x_min, y_min, x_max, y_max),…]
2.  sequence; frame;
```



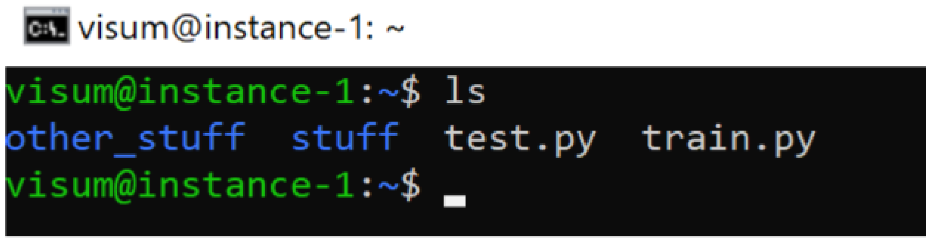
Can be found in:

/home/master/dataset/train/labels.csv

Inference

- Test file path: /home/visum/test.py
- Should infer: /home/master/dataset/test
- Should generate a **valid** predictions.csv at /home/visum/predictions.csv

Your home/visum/ will be copied to our **server** and we will run inference and scoring **there**



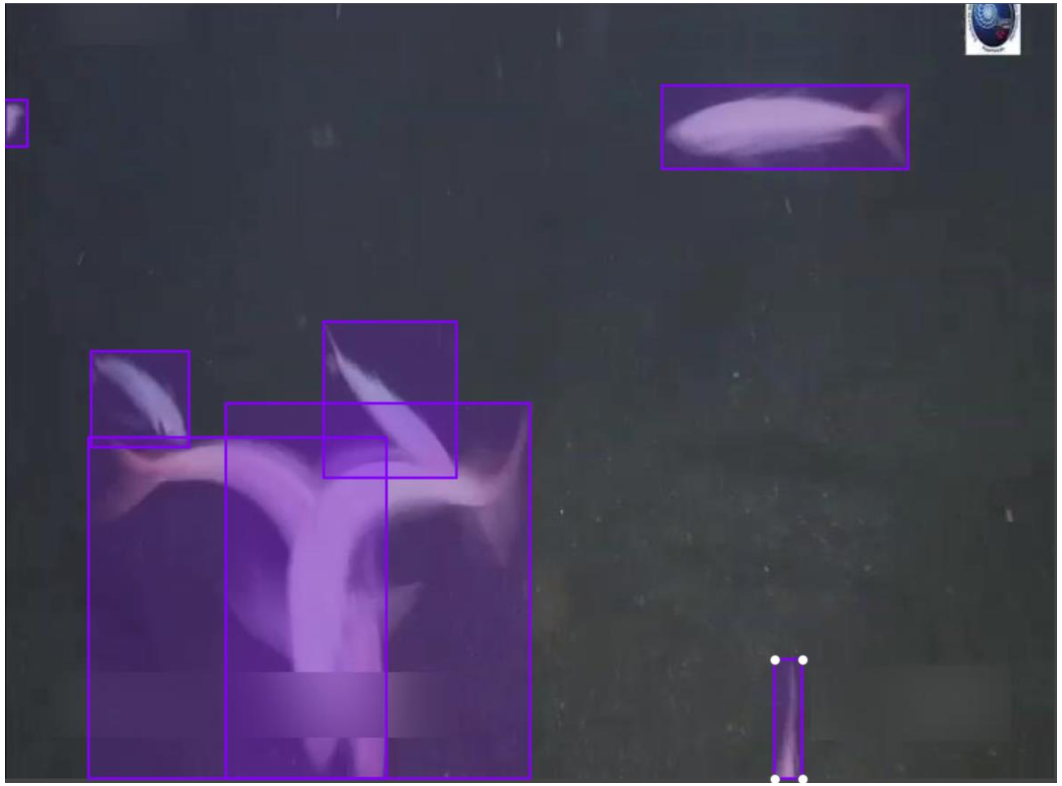
visum@instance-1: ~

```
visum@instance-1:~$ ls
other_stuff  stuff  test.py  train.py
visum@instance-1:~$ _
```

Predictions

CSV File

1. `sequence; frame; [(x_min, y_min, x_max, y_max)]; confidence`



Must be saved in:

/home/visum/predictions.csv

Validating Submissions

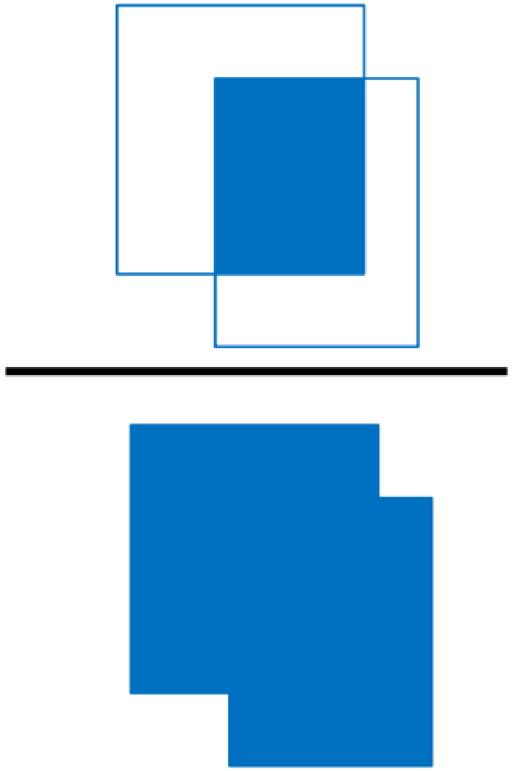
“How do I test if my submission is valid?”

- Run:
  - cd /home/visum/
  - rm predictions.csv
  - python test.py
  - python evaluate.py

Evaluation

IOU

$$IOU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

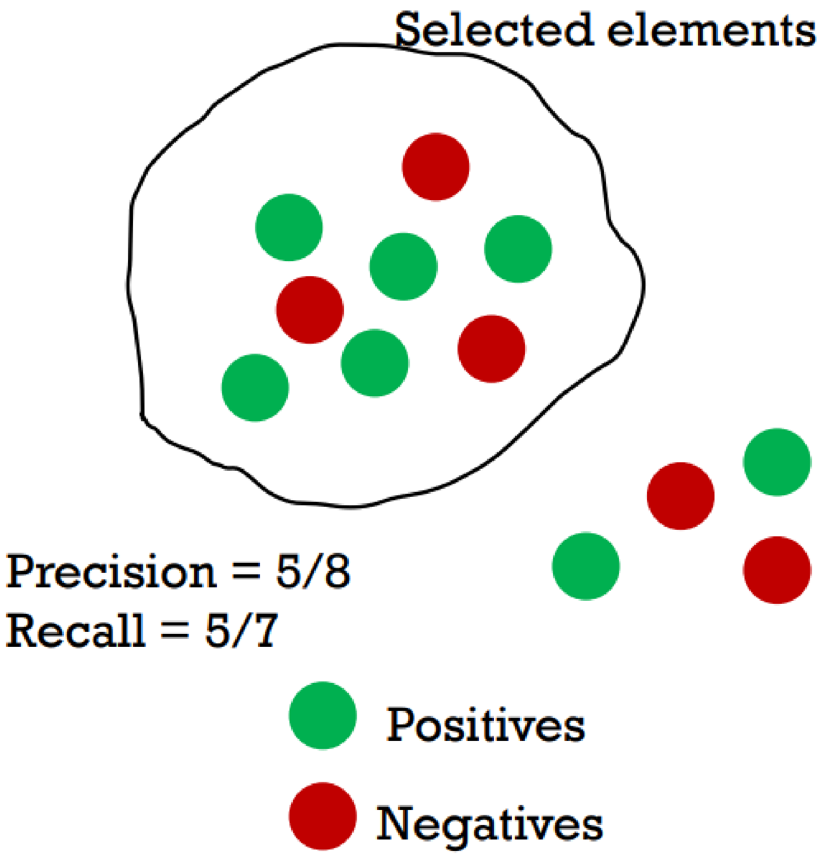


Evaluation

Precision and Recall

$$Precision = \frac{TPs}{Total\ detections}$$

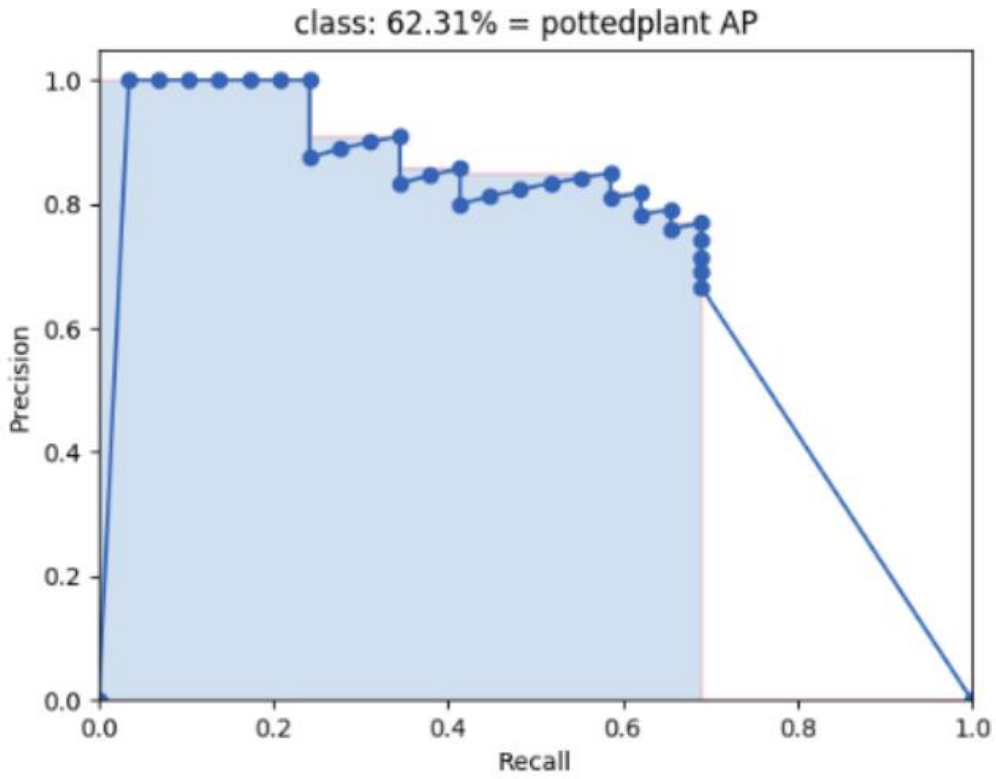
$$Recall = \frac{TPs}{Total\ objects}$$



Precision = 5/8
Recall = 5/7

Evaluation

Average Precision



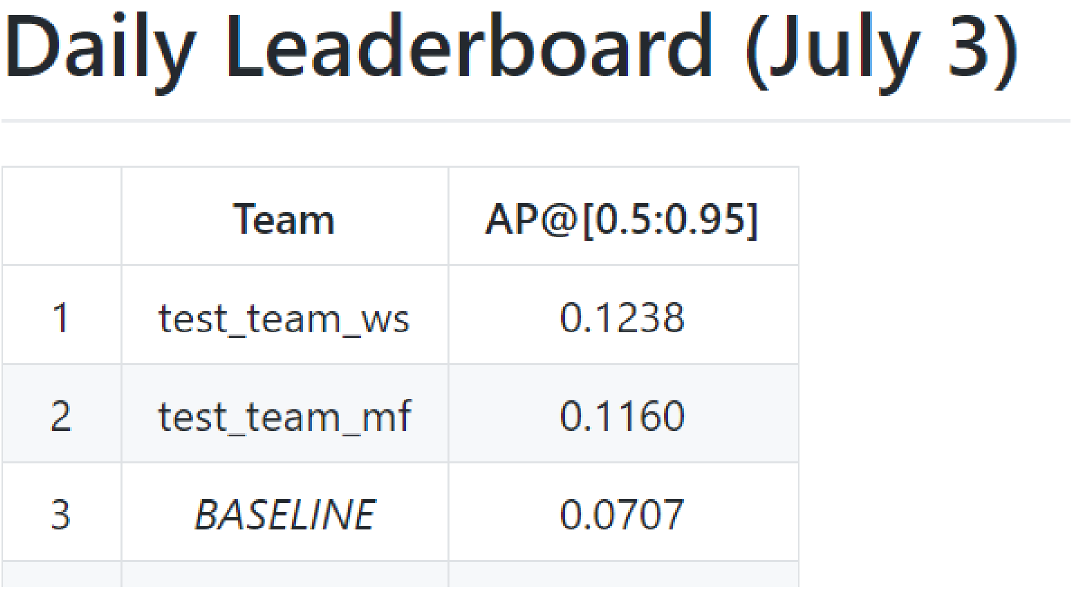
https://github.com/cartucho/map

AP@[0.5:0.95]

- Averaged over different IoU thresholds

Daily, Final and Current Leaderboard

- Should run every day at around 12am
- Additional run whenever the team wants!
- Runs on a subset of the test set

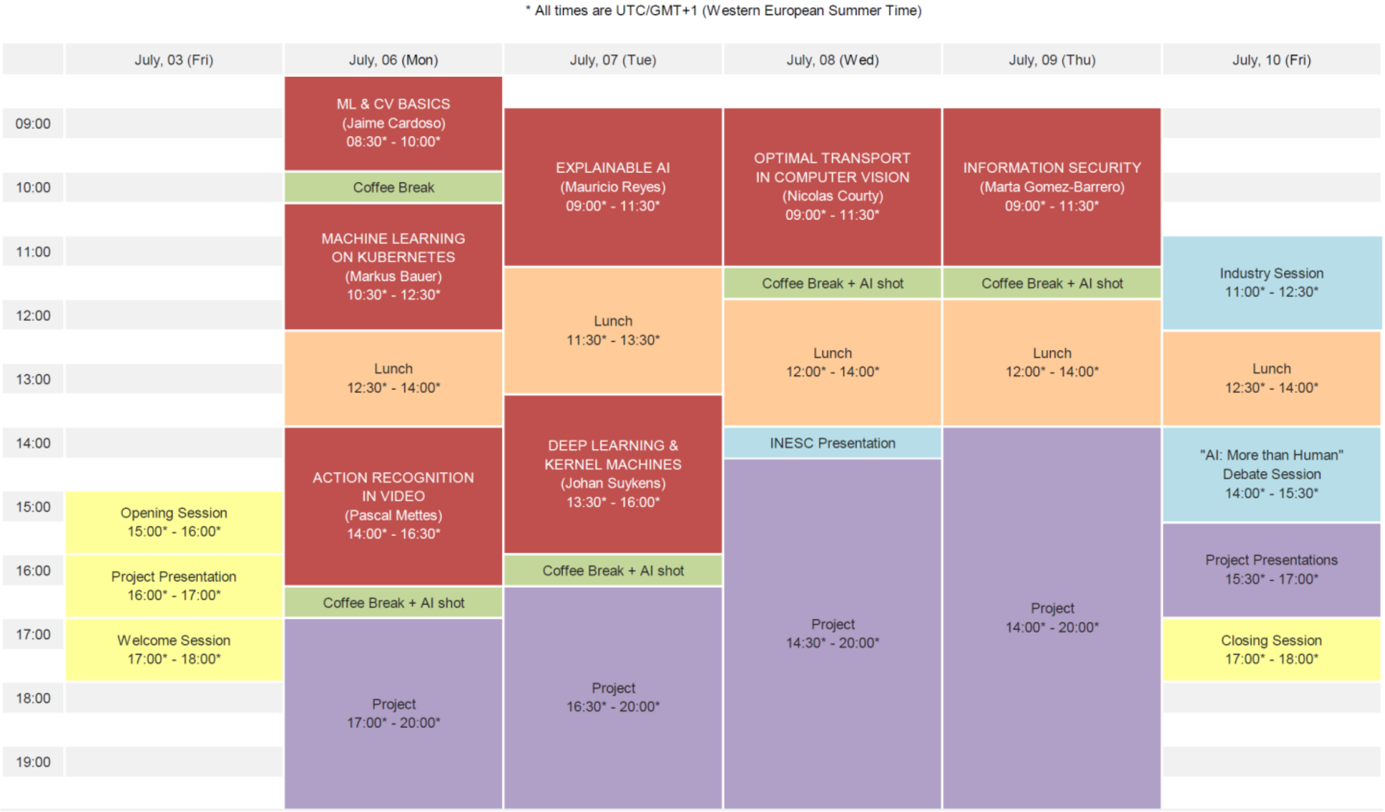


Daily Leaderboard (July 3)

	Team	AP@[0.5:0.95]
1	test_team_ws	0.1238
2	test_team_mf	0.1160
3	*BASELINE*	0.0707

Competition Duration

(Purple blocks)



* All times are UTC/GMT+1 (Western European Summer Time)

	July, 03 (Fri)	July, 06 (Mon)	July, 07 (Tue)	July, 08 (Wed)	July, 09 (Thu)	July, 10 (Fri)
		ML & CV BASICS (Jaime Cardoso) 08:30* - 10:00*				
09:00			EXPLAINABLE AI (Mauricio Reyes) 09:00* - 11:30*	OPTIMAL TRANSPORT IN COMPUTER VISION (Nicolas Courty) 09:00* - 11:30*	INFORMATION SECURITY (Marta Gomez-Barrero) 09:00* - 11:30*	
10:00		Coffee Break				
		MACHINE LEARNING ON KUBERNETES (Markus Bauer) 10:30* - 12:30*				
11:00						Industry Session 11:00* - 12:30*
			Lunch 11:30* - 13:30*	Coffee Break + AI shot	Coffee Break + AI shot	
12:00				Lunch 12:00* - 14:00*	Lunch 12:00* - 14:00*	
		Lunch 12:30* - 14:00*				Lunch 12:30* - 14:00*
13:00						
			DEEP LEARNING & KERNEL MACHINES (Johan Suykens) 13:30* - 16:00*			
14:00		ACTION RECOGNITION IN VIDEO (Pascal Mettes) 14:00* - 16:30*		INESC Presentation	Project 14:00* - 20:00*	"AI: More than Human" Debate Session 14:00* - 15:30*
				Project 14:30* - 20:00*		
15:00	Opening Session 15:00* - 16:00*					
						Project Presentations 15:30* - 17:00*
16:00	Project Presentation 16:00* - 17:00*		Coffee Break + AI shot			
		Coffee Break + AI shot	Project 16:30* - 20:00*			
17:00	Welcome Session 17:00* - 18:00*	Project 17:00* - 20:00*				Closing Session 17:00* - 18:00*
18:00						
19:00						

Development

You must use python3 as a programming language;

Your machine has some software installed: CUDA, cuDNN, Numpy, Open-CV, TensorFlow, PyTorch, Scikit-Learn;

You may not install software packages other than python packages;

Please, do not use virtual environments (e.g. conda);

We recommend you to not use Jupyter notebooks;

Your script (test.py) should run in acceptable time (less than one hour).

Winning the project competition

Teams are eligible to win the project competition if:

1. At the end of the competition they have a valid submission;
2. They have been chosen for an oral presentation at the last the day of the summer school;

The winner will be determined by a jury composed of both VISUM and Abyssal members. The main criterion will be the highest average precision. However, creativity, novelty, and the ability to communicate ideas will also be considered in the final decision.

The Prize

By participating in our competition, you can win:

- A 100€ Amazon Voucher
- Free registration to VISUM 2021

Finding help

If you have any **questions** regarding the project, relating to the rules, difficulties in understanding code or in setting up the google cloud machines, please find one member of the project staff:

- Eduardo - eduardo.m.castro@inesctec.pt
- Isabel - icrto@fe.up.pt
- João - joao.t.pinto@inesctec.pt
- Mafalda – mafalda.falcao@fe.up.pt
- Ricardo - ricardo.j.araujo@inesctec.pt
- Wilson - wilson.j.silva@inesctec.pt

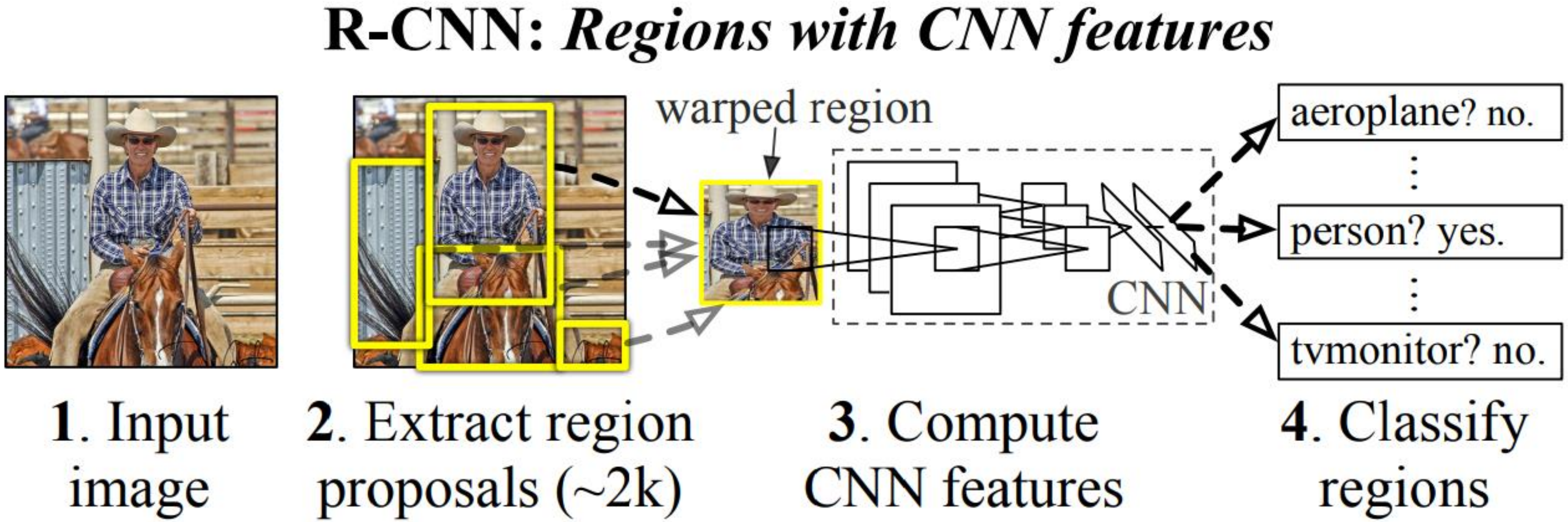
Developing a Baseline

Network Goal: Object Detection

State-of-the-art Architectures:

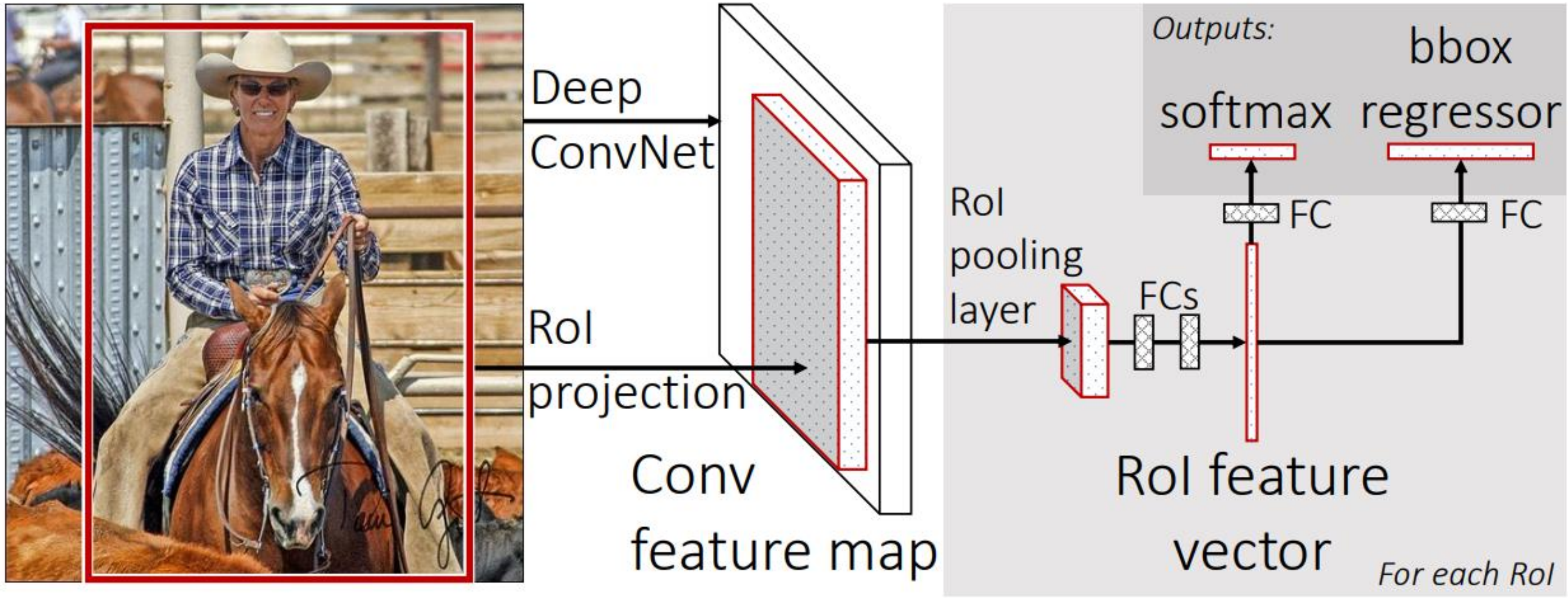
- YOLO (https://arxiv.org/pdf/1506.02640.pdf);
- RetinaNet (https://arxiv.org/abs/1708.02002.pdf);
- **Faster R-CNN** (https://arxiv.org/pdf/1506.01497.pdf).

The R-CNN Architecture



https://arxiv.org/pdf/1311.2524.pdf

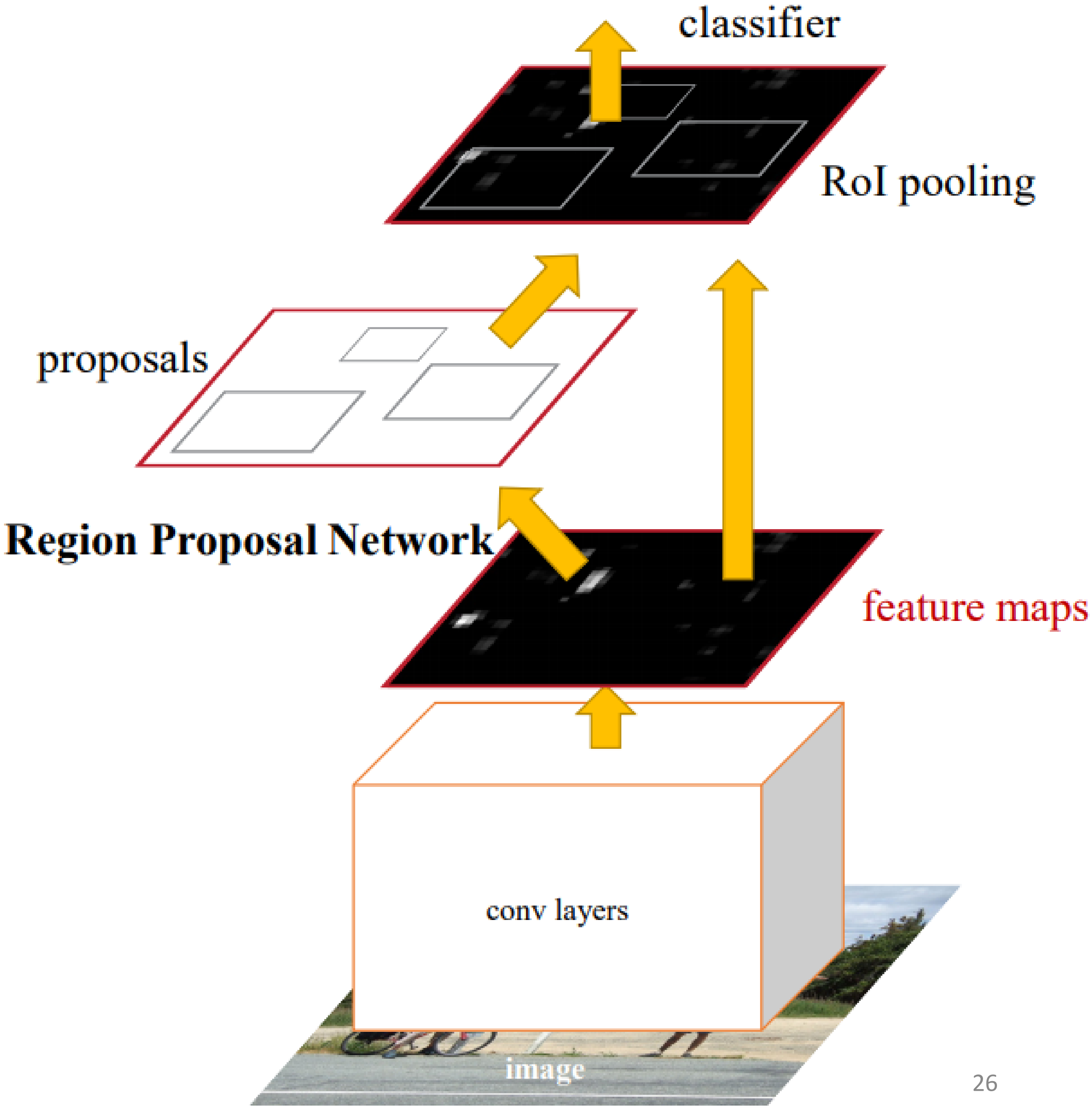
The Fast R-CNN Architecture



https://arxiv.org/pdf/1504.08083.pdf

The Faster R-CNN

The Baseline



The Baseline Solution

Implementation:

- PyTorch (*torchvision*)

  https://pytorch.org/docs/stable/torchvision/models.html#object-detection-instance-segmentation-and-person-keypoint-detection

- Adapted from Mask R-CNN tutorial

  https://colab.research.google.com/github/pytorch/vision/blob/temp-tutorial/tutorials/torchvision_finetuning_ instance_segmentation.ipynb

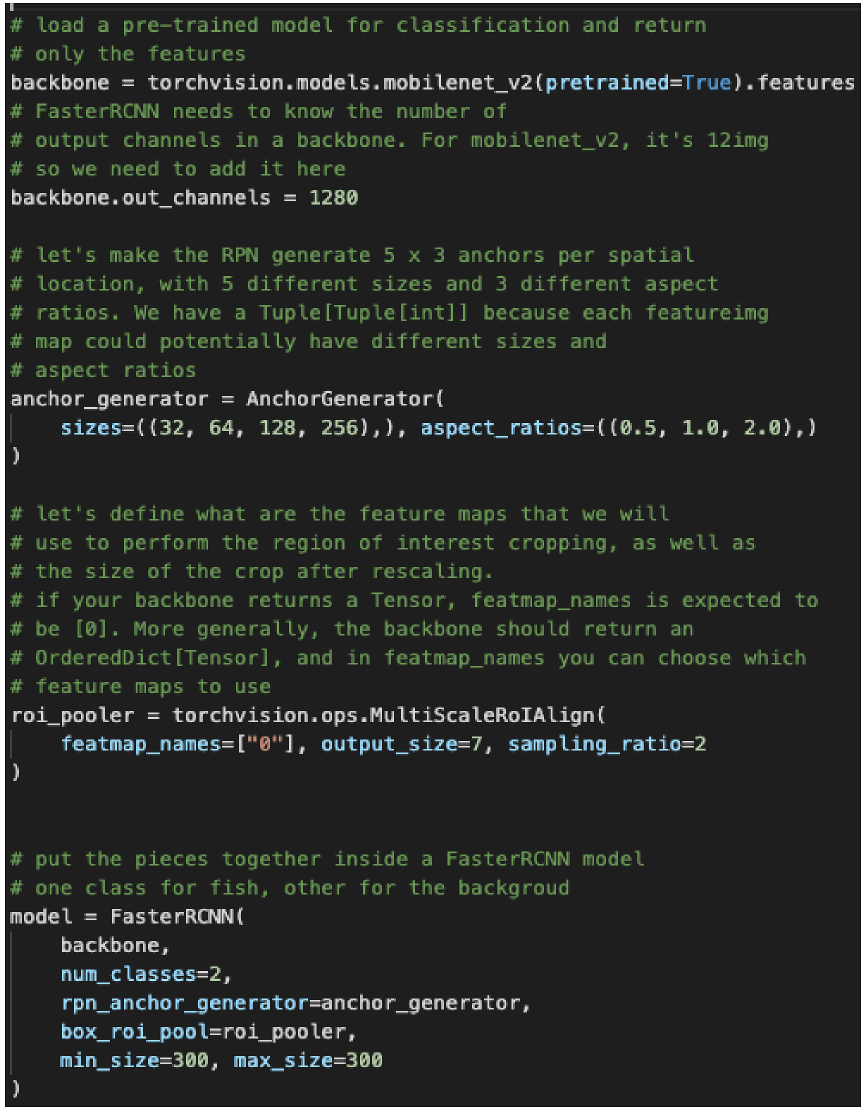
The Baseline Solution

Files:

- train.py – Train and save your model;
- test.py – Load your trained model, post-process it and make predictions;
- plot_results.py - Visualize your predictions;
- evaluate.py – Check your scores!

The Baseline Solution

train.py

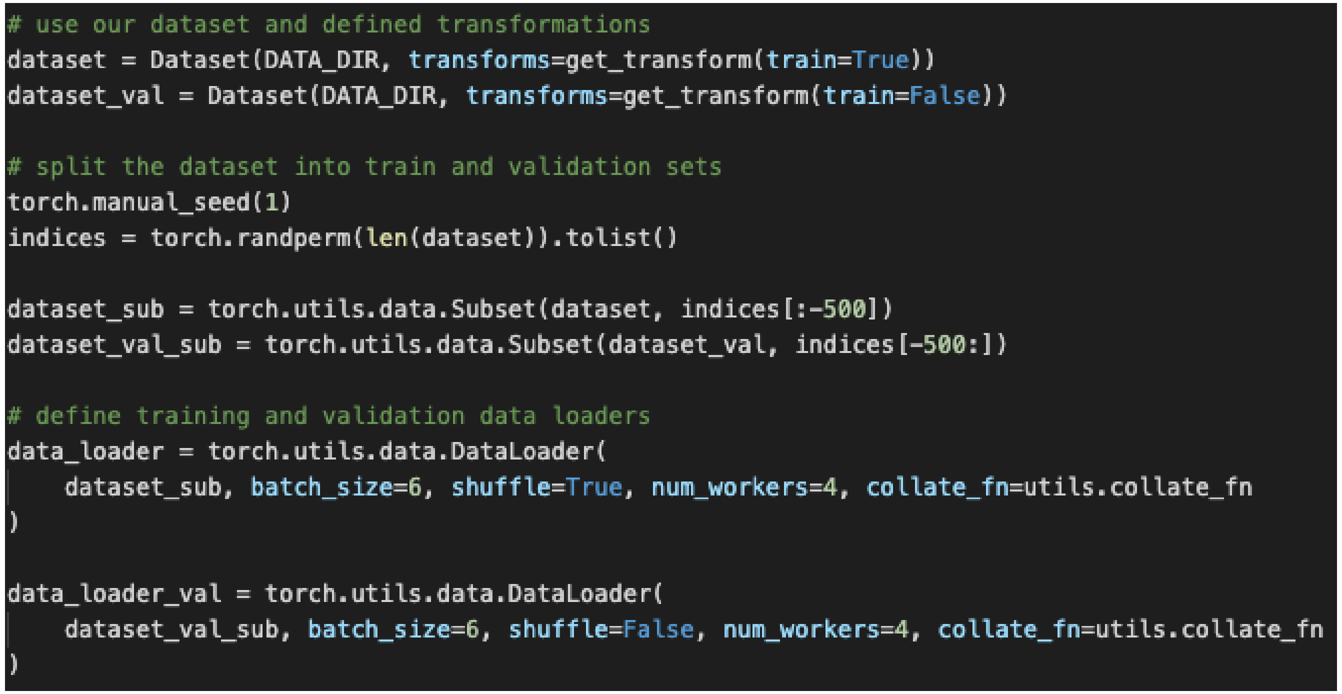


```
# load a pre-trained model for classification and return
# only the features
backbone = torchvision.models.mobilenet_v2(pretrained=True).features
# FasterRCNN needs to know the number of
# output channels in a backbone. For mobilenet_v2, it's 12img
# so we need to add it here
backbone.out_channels = 1280

# let's make the RPN generate 5 x 3 anchors per spatial
# location, with 5 different sizes and 3 different aspect
# ratios. We have a Tuple[Tuple[int]] because each featureimg
# map could potentially have different sizes and
# aspect ratios
anchor_generator = AnchorGenerator(
    sizes=((32, 64, 128, 256),), aspect_ratios=((0.5, 1.0, 2.0),)
)

# let's define what are the feature maps that we will
# use to perform the region of interest cropping, as well as
# the size of the crop after rescaling.
# if your backbone returns a Tensor, featmap_names is expected to
# be [0]. More generally, the backbone should return an
# OrderedDict[Tensor], and in featmap_names you can choose which
# feature maps to use
roi_pooler = torchvision.ops.MultiScaleRoIAlign(
    featmap_names=["0"], output_size=7, sampling_ratio=2
)


# put the pieces together inside a FasterRCNN model
# one class for fish, other for the backgroud
model = FasterRCNN(
    backbone,
    num_classes=2,
    rpn_anchor_generator=anchor_generator,
    box_roi_pool=roi_pooler,
    min_size=300, max_size=300
)
```

Model Definition

The Baseline Solution

train.py

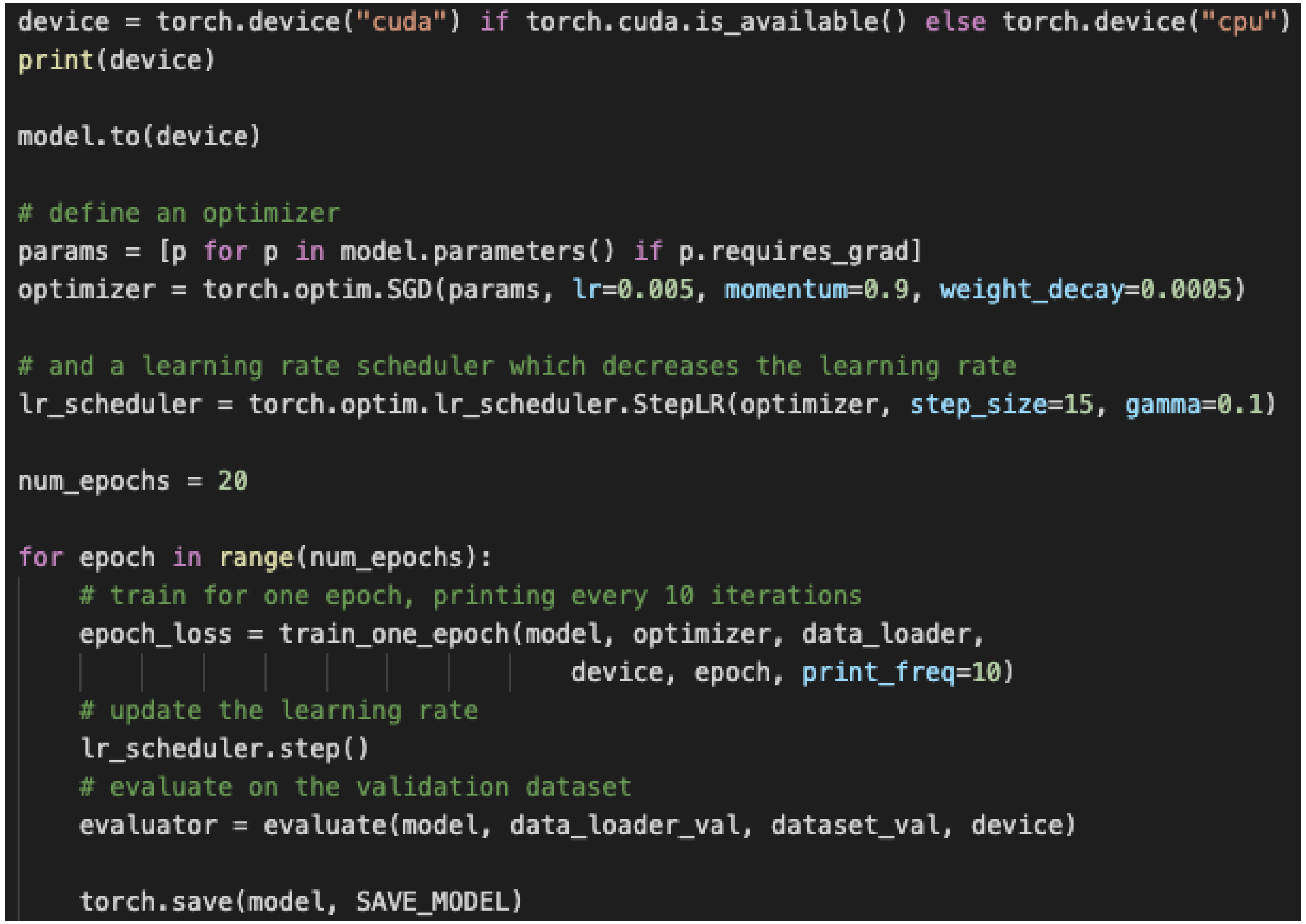


```
# use our dataset and defined transformations
dataset = Dataset(DATA_DIR, transforms=get_transform(train=True))
dataset_val = Dataset(DATA_DIR, transforms=get_transform(train=False))

# split the dataset into train and validation sets
torch.manual_seed(1)
indices = torch.randperm(len(dataset)).tolist()

dataset_sub = torch.utils.data.Subset(dataset, indices[:-500])
dataset_val_sub = torch.utils.data.Subset(dataset_val, indices[-500:])

# define training and validation data loaders
data_loader = torch.utils.data.DataLoader(
    dataset_sub, batch_size=6, shuffle=True, num_workers=4, collate_fn=utils.collate_fn
)

data_loader_val = torch.utils.data.DataLoader(
    dataset_val_sub, batch_size=6, shuffle=False, num_workers=4, collate_fn=utils.collate_fn
)
```

Split data into training and validation.

The Baseline Solution

train.py

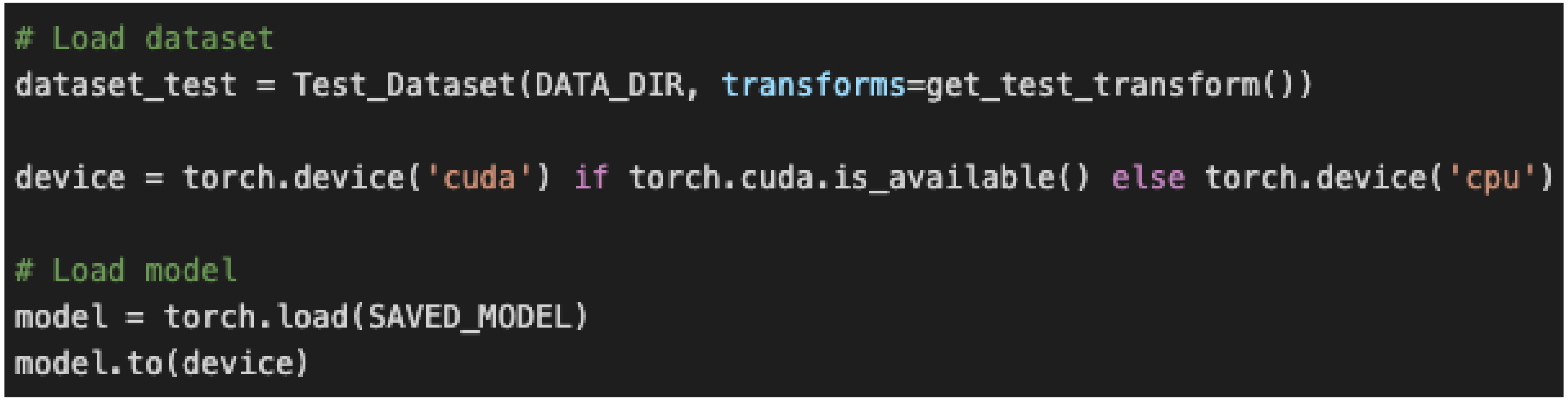


```python
device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
print(device)

model.to(device)

# define an optimizer
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=0.005, momentum=0.9, weight_decay=0.0005)

# and a learning rate scheduler which decreases the learning rate
lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=15, gamma=0.1)

num_epochs = 20

for epoch in range(num_epochs):
    # train for one epoch, printing every 10 iterations
    epoch_loss = train_one_epoch(model, optimizer, data_loader,
                                    device, epoch, print_freq=10)
    # update the learning rate
    lr_scheduler.step()
    # evaluate on the validation dataset
    evaluator = evaluate(model, data_loader_val, dataset_val, device)

    torch.save(model, SAVE_MODEL)
```

Define optimizer and train your model. Finally, save it!

The Baseline Solution

test.py

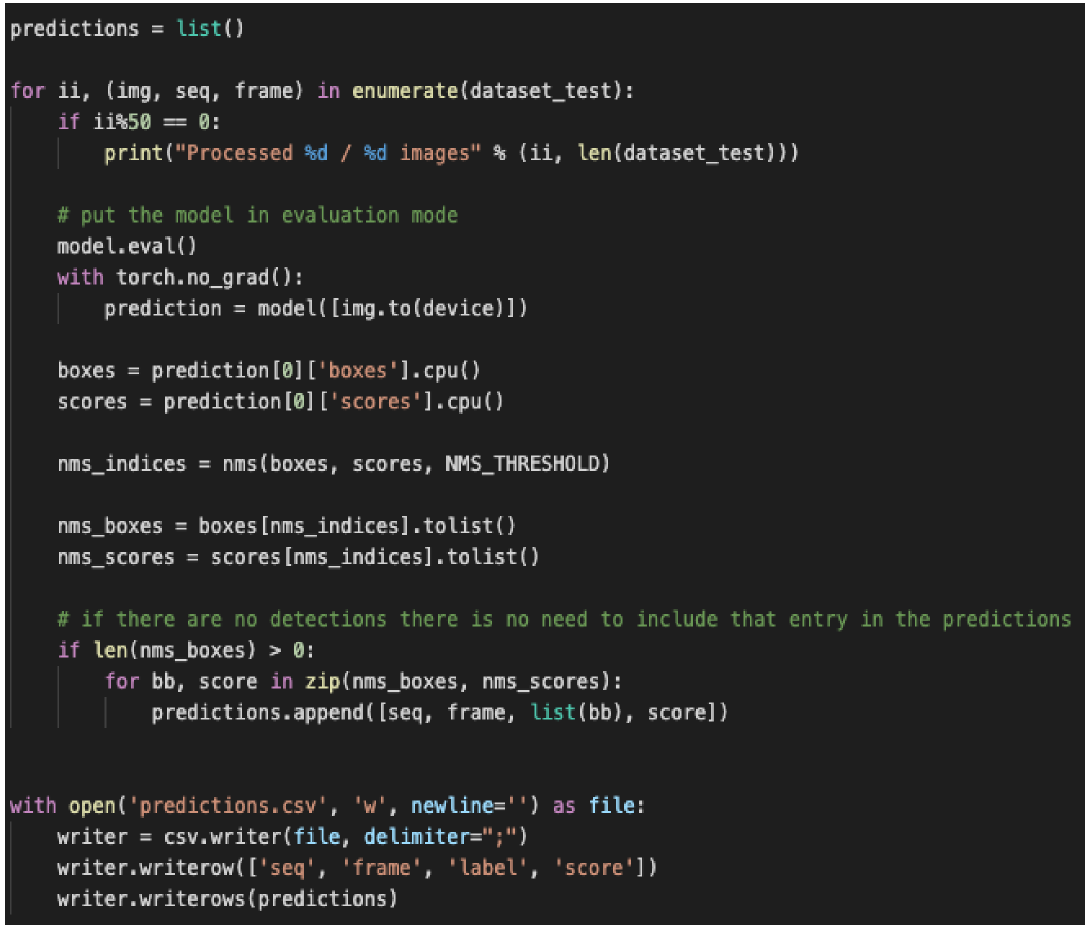


```
# Load dataset
dataset_test = Test_Dataset(DATA_DIR, transforms=get_test_transform())

device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')

# Load model
model = torch.load(SAVED_MODEL)
model.to(device)
```

Load your data and your model.

The Baseline Solution

test.py

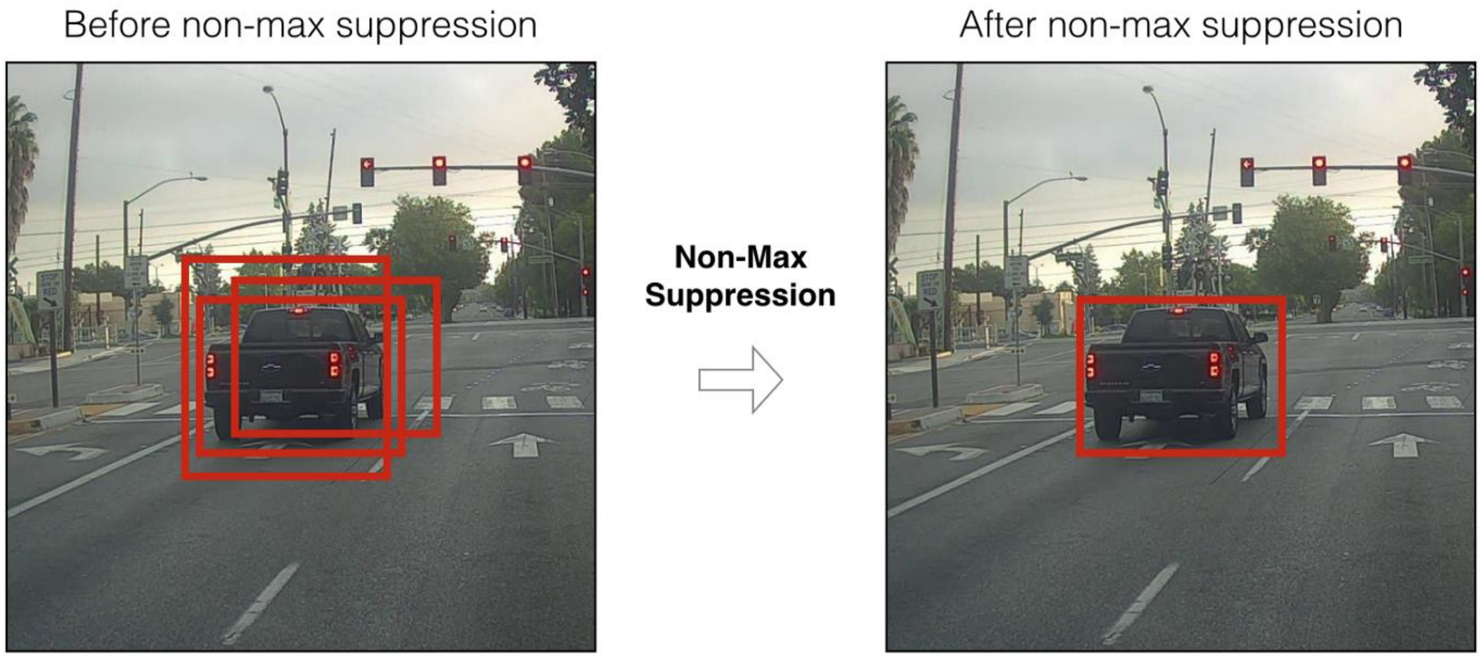


```
predictions = list()

for ii, (img, seq, frame) in enumerate(dataset_test):
    if ii%50 == 0:
        print("Processed %d / %d images" % (ii, len(dataset_test)))

    # put the model in evaluation mode
    model.eval()
    with torch.no_grad():
        prediction = model([img.to(device)])

    boxes = prediction[0]['boxes'].cpu()
    scores = prediction[0]['scores'].cpu()

    nms_indices = nms(boxes, scores, NMS_THRESHOLD)

    nms_boxes = boxes[nms_indices].tolist()
    nms_scores = scores[nms_indices].tolist()

    # if there are no detections there is no need to include that entry in the predictions
    if len(nms_boxes) > 0:
        for bb, score in zip(nms_boxes, nms_scores):
            predictions.append([seq, frame, list(bb), score])


with open('predictions.csv', 'w', newline='') as file:
    writer = csv.writer(file, delimiter=";")
    writer.writerow(['seq', 'frame', 'label', 'score'])
    writer.writerows(predictions)
```

Make predictions, post-process them, and finally, save them in a csv file.

The Baseline Solution

test.py



Before non-max suppression

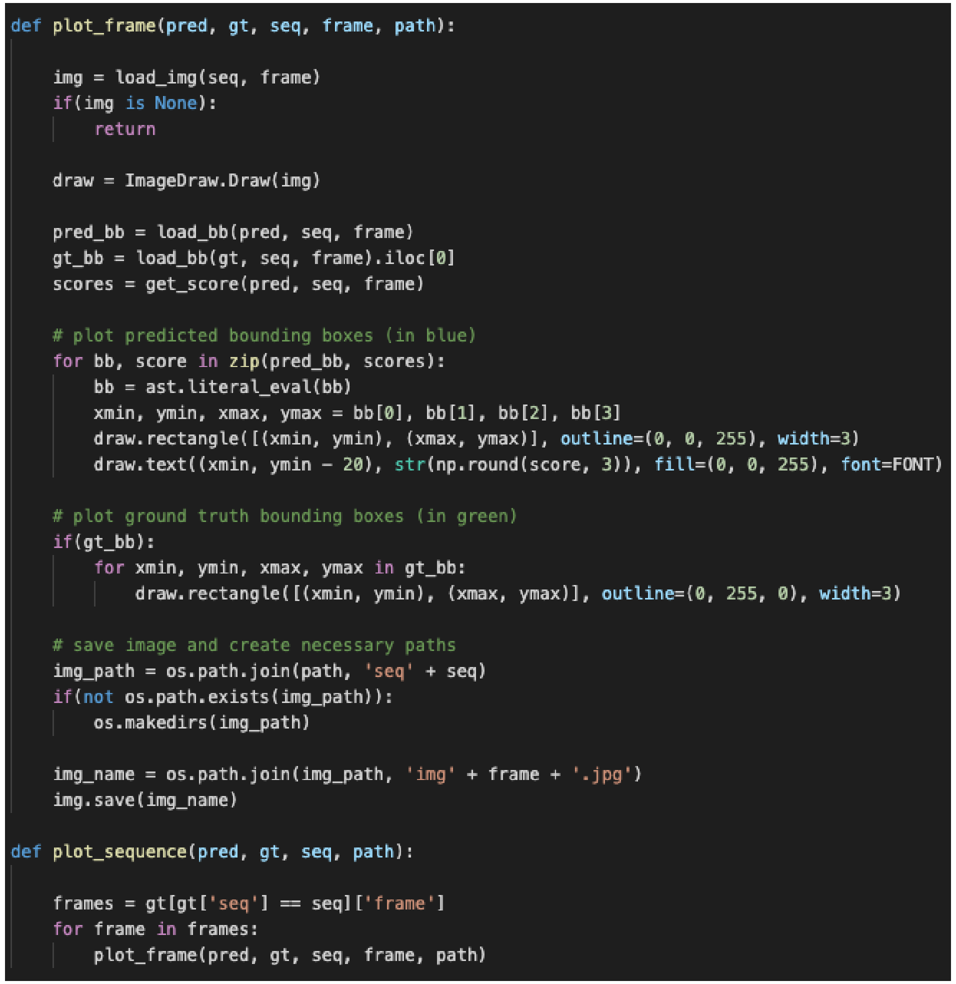
Non-Max Suppression

After non-max suppression

Source: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

The Baseline Solution

plot_results.py

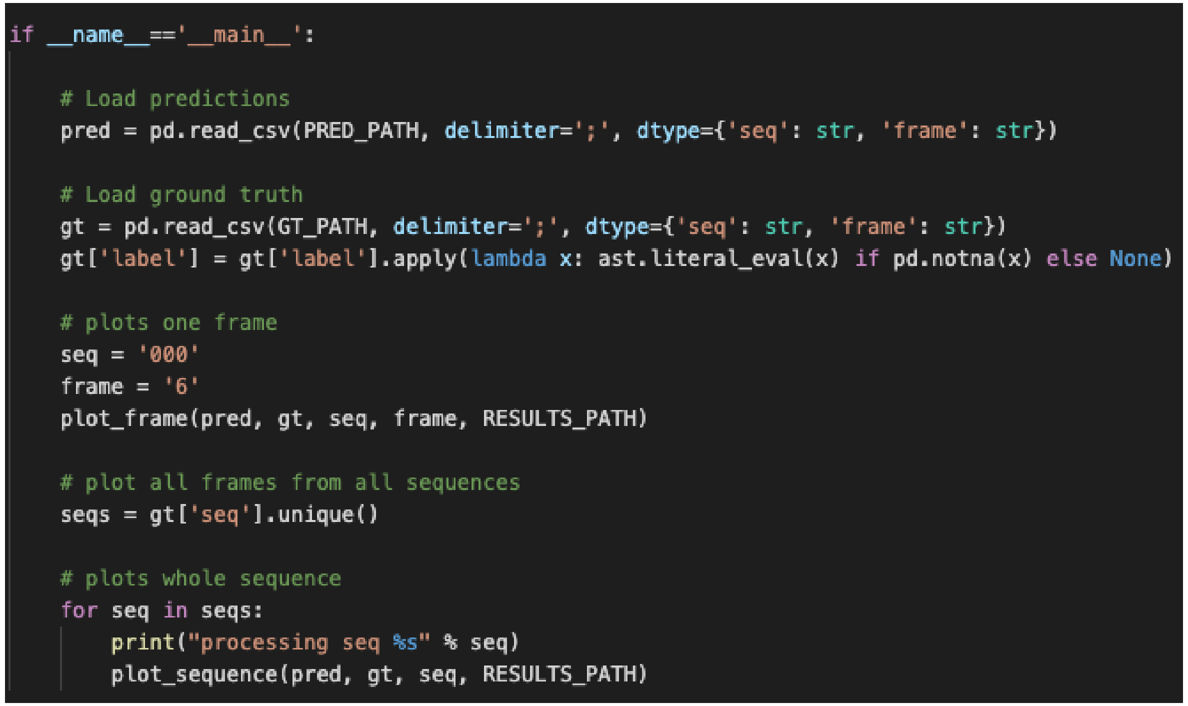


```
def plot_frame(pred, gt, seq, frame, path):

    img = load_img(seq, frame)
    if(img is None):
        return

    draw = ImageDraw.Draw(img)

    pred_bb = load_bb(pred, seq, frame)
    gt_bb = load_bb(gt, seq, frame).iloc[0]
    scores = get_score(pred, seq, frame)

    # plot predicted bounding boxes (in blue)
    for bb, score in zip(pred_bb, scores):
        bb = ast.literal_eval(bb)
        xmin, ymin, xmax, ymax = bb[0], bb[1], bb[2], bb[3]
        draw.rectangle([(xmin, ymin), (xmax, ymax)], outline=(0, 0, 255), width=3)
        draw.text((xmin, ymin - 20), str(np.round(score, 3)), fill=(0, 0, 255), font=FONT)

    # plot ground truth bounding boxes (in green)
    if(gt_bb):
        for xmin, ymin, xmax, ymax in gt_bb:
            draw.rectangle([(xmin, ymin), (xmax, ymax)], outline=(0, 255, 0), width=3)

    # save image and create necessary paths
    img_path = os.path.join(path, 'seq' + seq)
    if(not os.path.exists(img_path)):
        os.makedirs(img_path)

    img_name = os.path.join(img_path, 'img' + frame + '.jpg')
    img.save(img_name)

def plot_sequence(pred, gt, seq, path):

    frames = gt[gt['seq'] == seq]['frame']
    for frame in frames:
        plot_frame(pred, gt, seq, frame, path)
```

Save a frame/sequence with its ground truth and predicted bounding boxes.

The Baseline Solution

plot_results.py

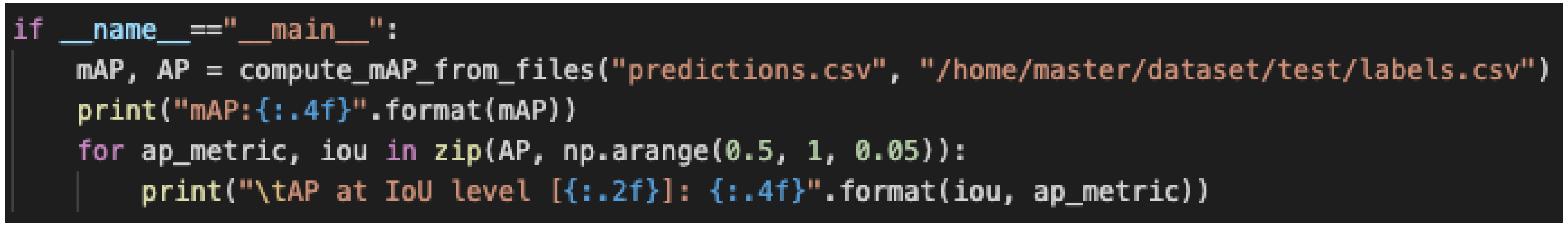


```python
if __name__=='__main__':

    # Load predictions
    pred = pd.read_csv(PRED_PATH, delimiter=';', dtype={'seq': str, 'frame': str})

    # Load ground truth
    gt = pd.read_csv(GT_PATH, delimiter=';', dtype={'seq': str, 'frame': str})
    gt['label'] = gt['label'].apply(lambda x: ast.literal_eval(x) if pd.notna(x) else None)

    # plots one frame
    seq = '000'
    frame = '6'
    plot_frame(pred, gt, seq, frame, RESULTS_PATH)

    # plot all frames from all sequences
    seqs = gt['seq'].unique()

    # plots whole sequence
    for seq in seqs:
        print("processing seq %s" % seq)
        plot_sequence(pred, gt, seq, RESULTS_PATH)
```

Select which sequence or frame you want to inspect.

The Baseline Solution

evaluate.py



```python
if __name__=="__main__":
    mAP, AP = compute_mAP_from_files("predictions.csv", "/home/master/dataset/test/labels.csv")
    print("mAP:{:.4f}".format(mAP))
    for ap_metric, iou in zip(AP, np.arange(0.5, 1, 0.05)):
        print("\tAP at IoU level [{:.2f}]: {:.4f}".format(iou, ap_metric))
```

Compute scores by calling the metrics function.

Ideas / Suggestions:

- Tweak some model (hyper)parameters (anchors, backbones, data augmentation, data splitting);
- Find ways to deal with the big variability of visibility conditions underwater;
- Analyse sequences instead of individual images (use temporal information/context).

Start Exploring

Next Steps (Recommended)

Information

- Go to https://github.com/visum-summerschool/visum-competition2020
- Read the visum_project_FAQ.pdf file

Team Registration

- Organize into a group of **three** and register your team.
- If you do not have a team and want to participate, send us an e-mail and we will find you a team!

First result

- Set up a valid test.py and get your name on the leaderboard! You may use the provided baseline for this.