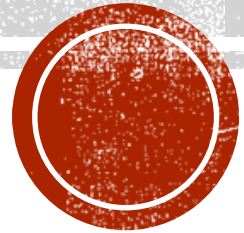
Google Cloud

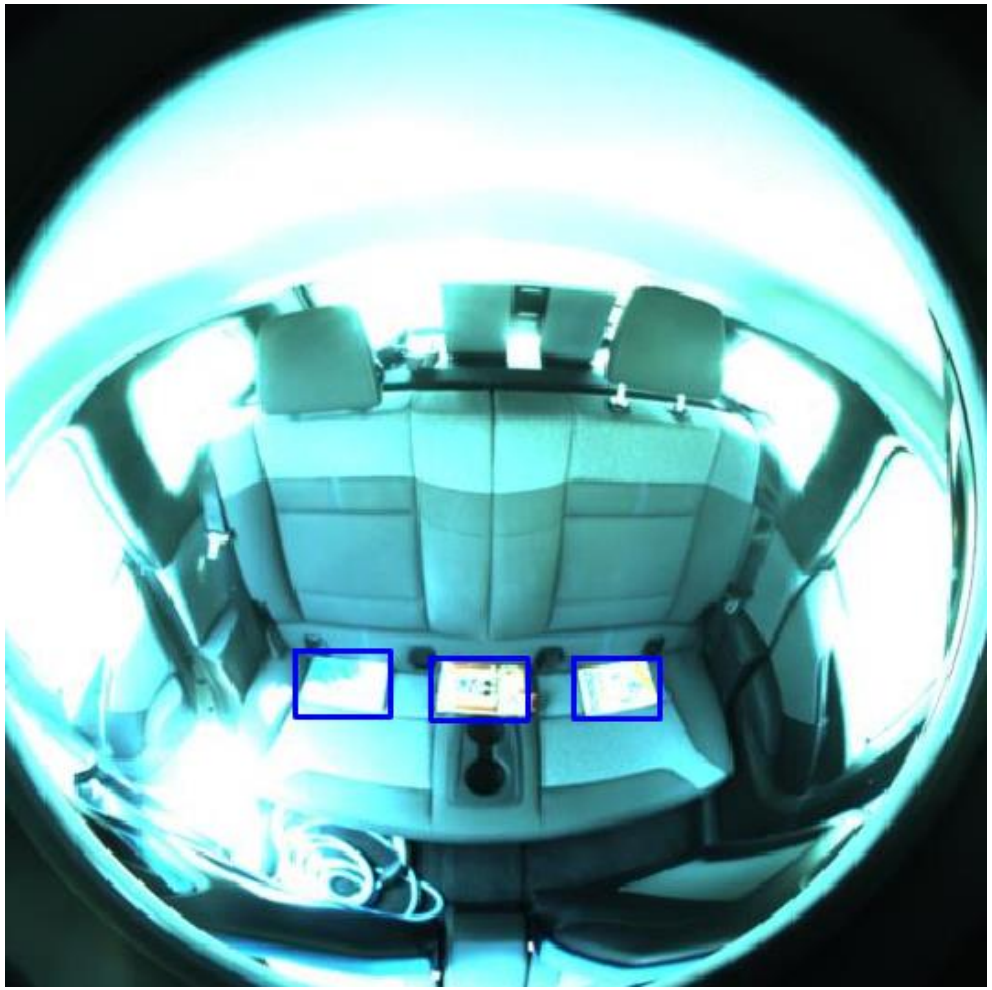
VISUM PROJECT



INESCTEC

BOSCH

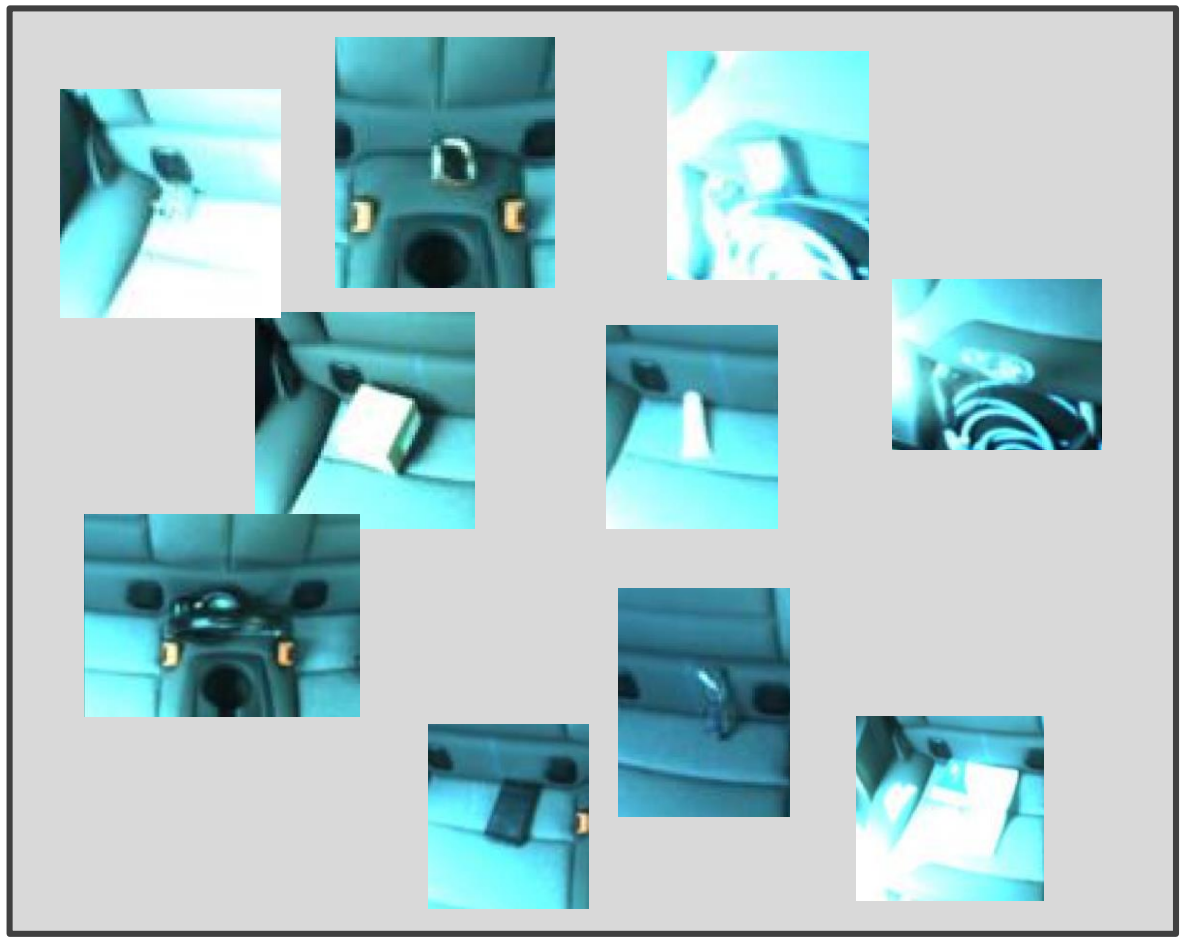
OBJECT DETECTION INSIDE A CAR



In collaboration with:

BOSCH

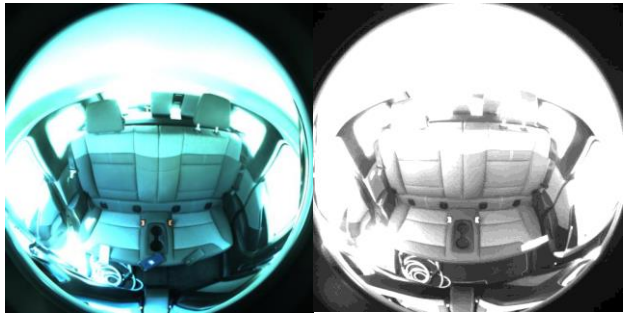
OBJECT DETECTION



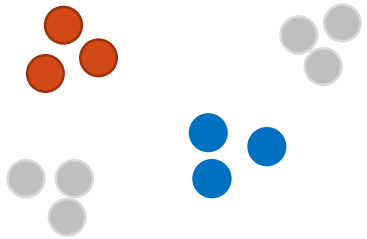
Classes of objects

- Book
- Bottle
- Box
- Cell phone
- Cosmetics
- Glasses
- Headphones
- Keys
- Wallet
- Watch

TASKS

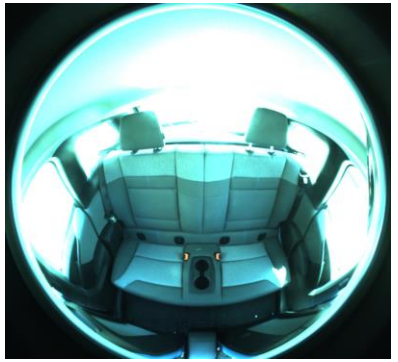


Additional modality for training



Open Set Classification

Object Detection



Empty Cars in Test

Each team will have access to a Google Cloud machine

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

THE DATASET

TRAIN:

- 911 RGB/NIR pairs;
- Annotations for the RGB images;
- All images contain at least one object.

TEST (NO ACCESS):

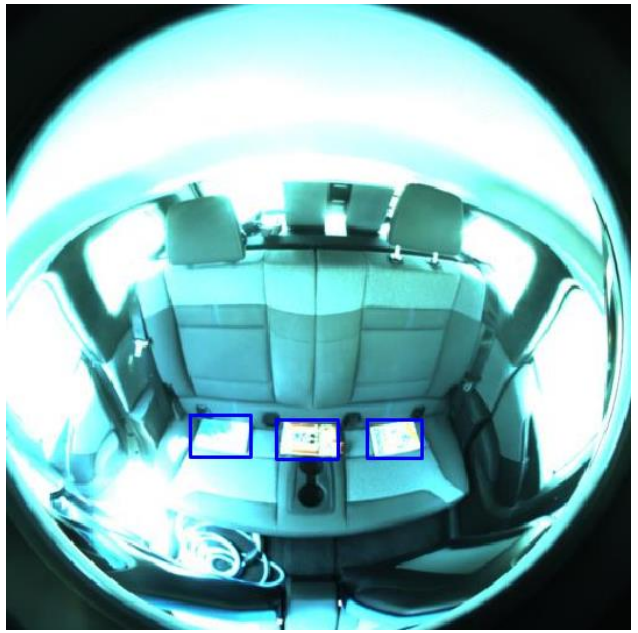
- 441 RGB images;
- Daily Test is a subset with 88 images;
- Some images do not contain any object;
- Some objects do not belong to any of the training classes.

ANNOTATIONS – CSV FILE

```
1. img_name, x_min, y_min, x_max, y_max, class
2. img_name, x_min, y_min, x_max, y_max, class
3. img_name, x_min, y_min, x_max, y_max, class
```

Can be found in:

/home/master/train/annotation.csv

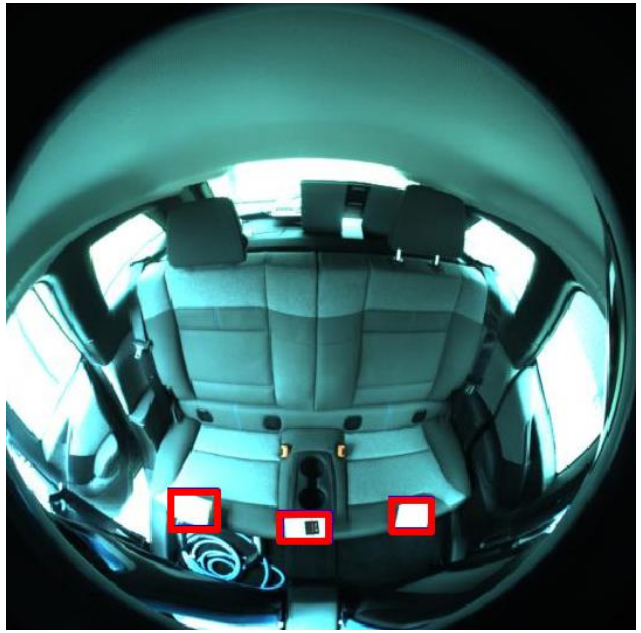


PREDICTIONS – CSV FILE

```
1. img_name, x_min, y_min, x_max, y_max, class, confidence
2. img_name, x_min, y_min, x_max, y_max, class, confidence
3. img_name, x_min, y_min, x_max, y_max, class, confidence
```

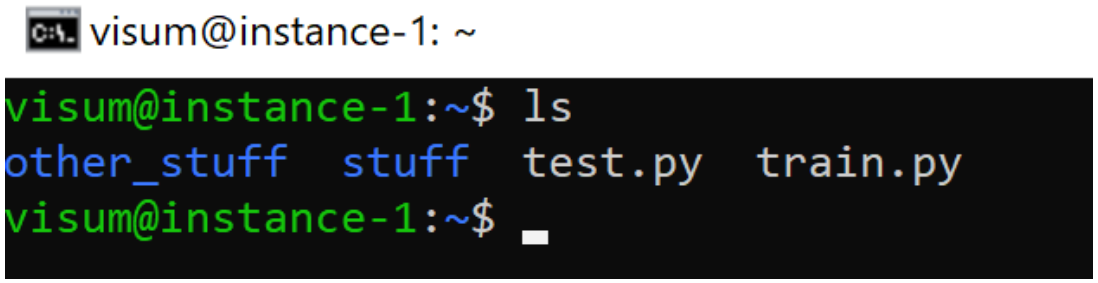
Must be saved in:

/home/visum/predictions.csv



INFERENCE

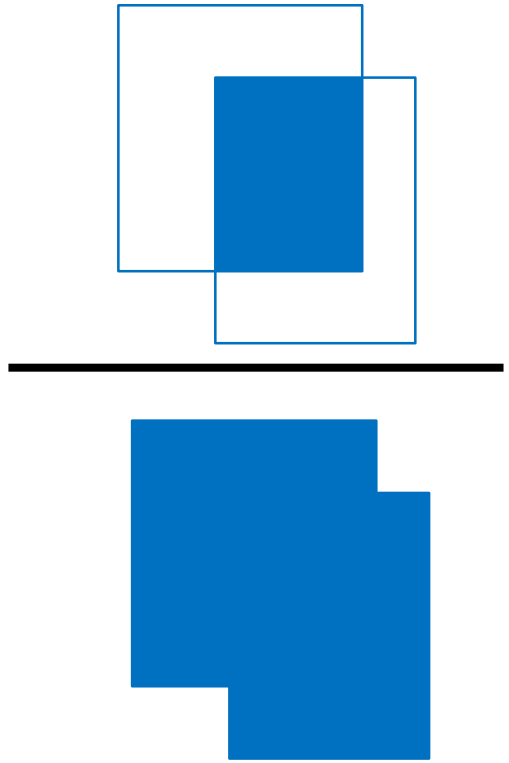
- Test file path: /home/visum/test.py
- Should infer: /home/master/dataset/test/
- Should generate a **valid** predictions.csv at `/home/visum/predictions.csv`



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

EVALUATION – IOU

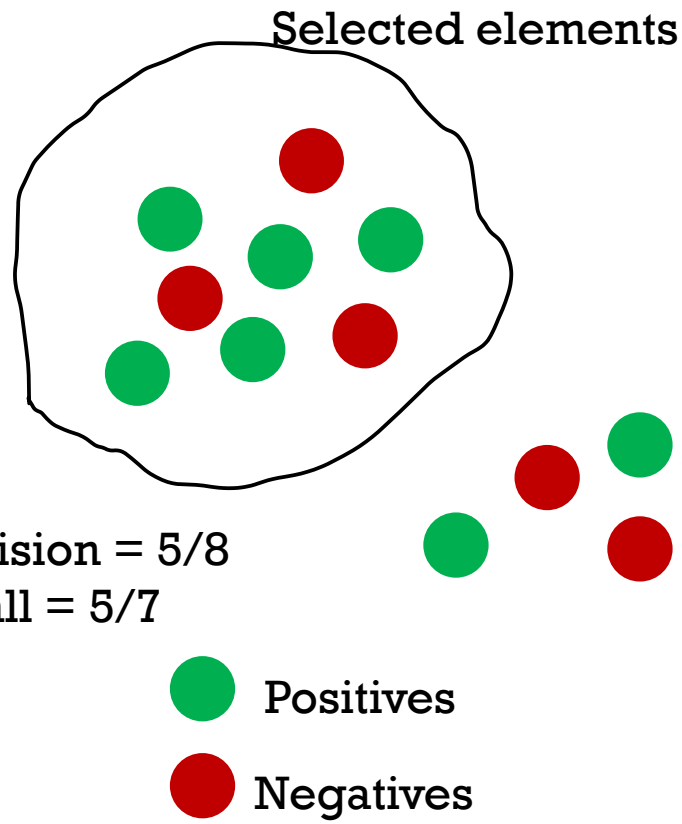
- $IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$



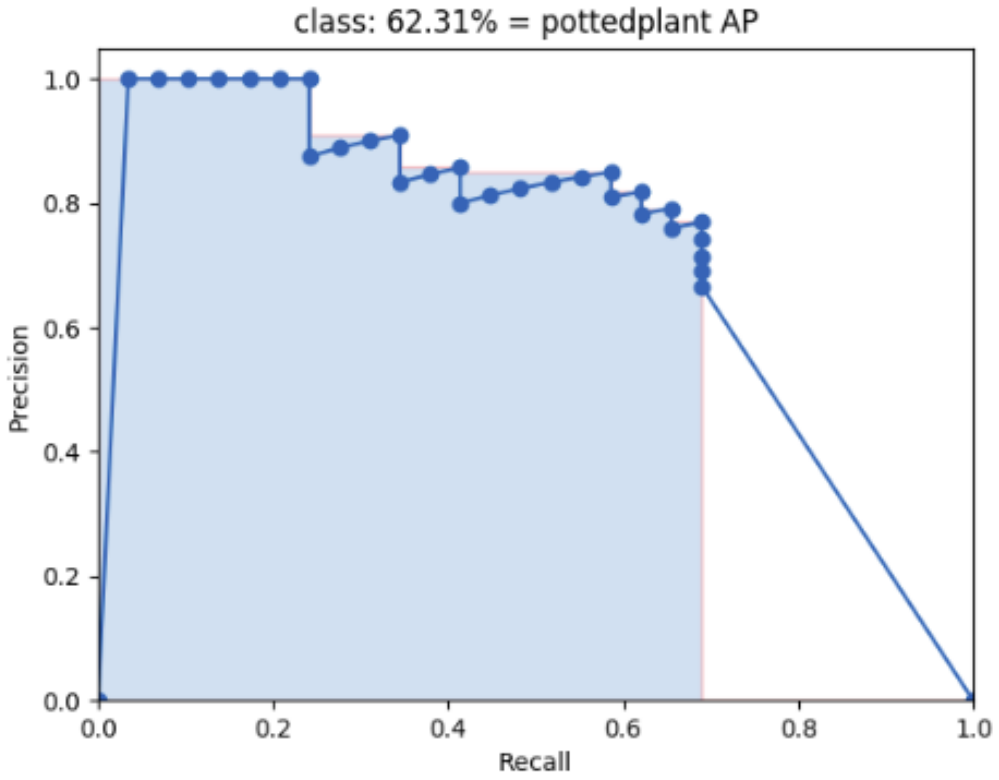
EVALUATION – PRECISION AND RECALL

- $Precision = \frac{TPs}{Total\ Detections}$

- $Recall = \frac{TPs}{Total\ Objects}$



EVALUATION – AVERAGE PRECISION



https://github.com/cartucho/map

mAP@[0.5:0.95]

- One class vs all;
- Averaged over all classes;
- Averaged over different IoU thresholds.

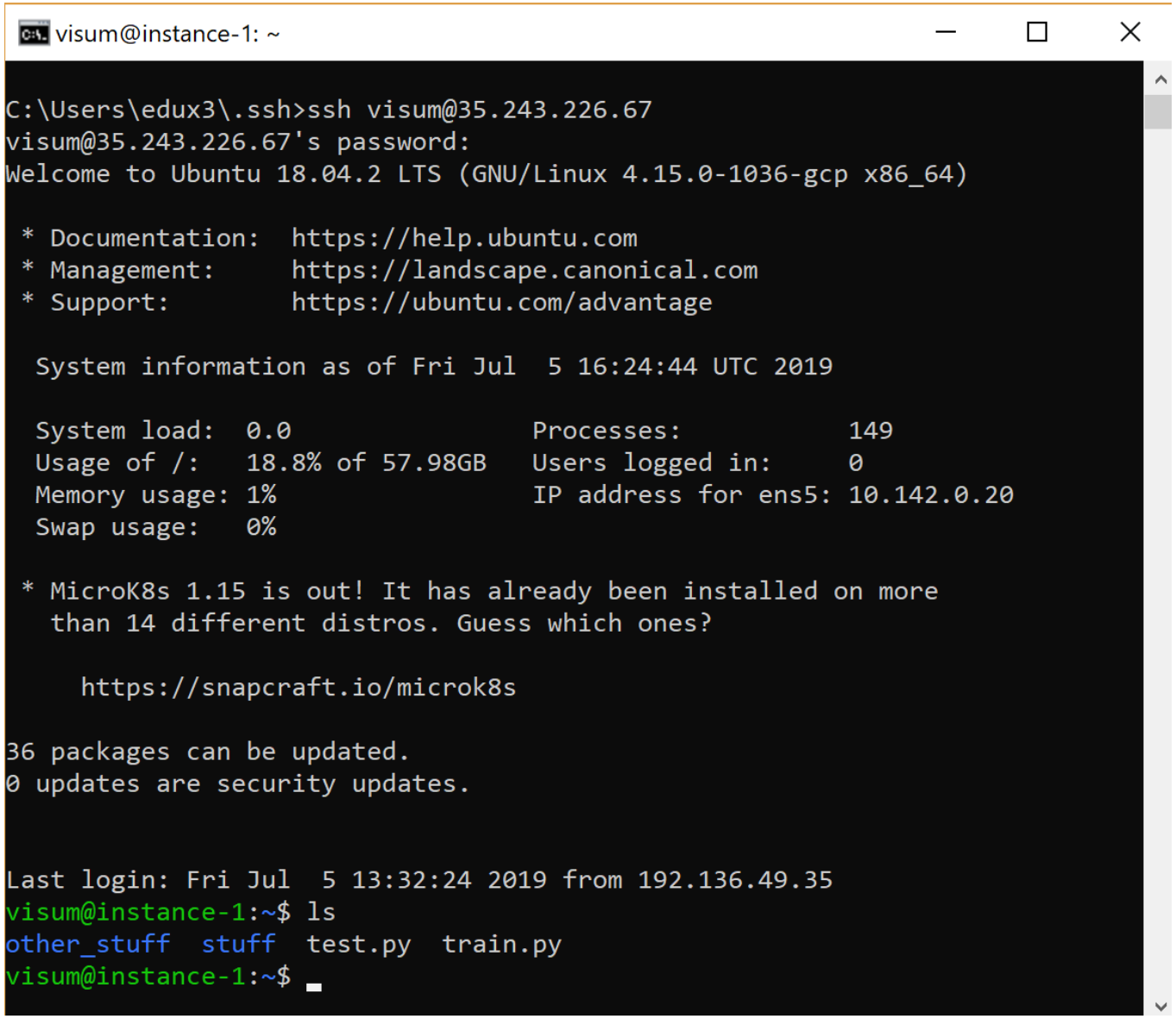
AP@[0.5:0.95] unknown class

- Unknown class vs all;
- Averaged over different IoU thresholds.

AP empty car

- Confidence that the car is empty is equal to:

$$-\max(detections)$$



USING YOUR MACHINE - CONNECTING

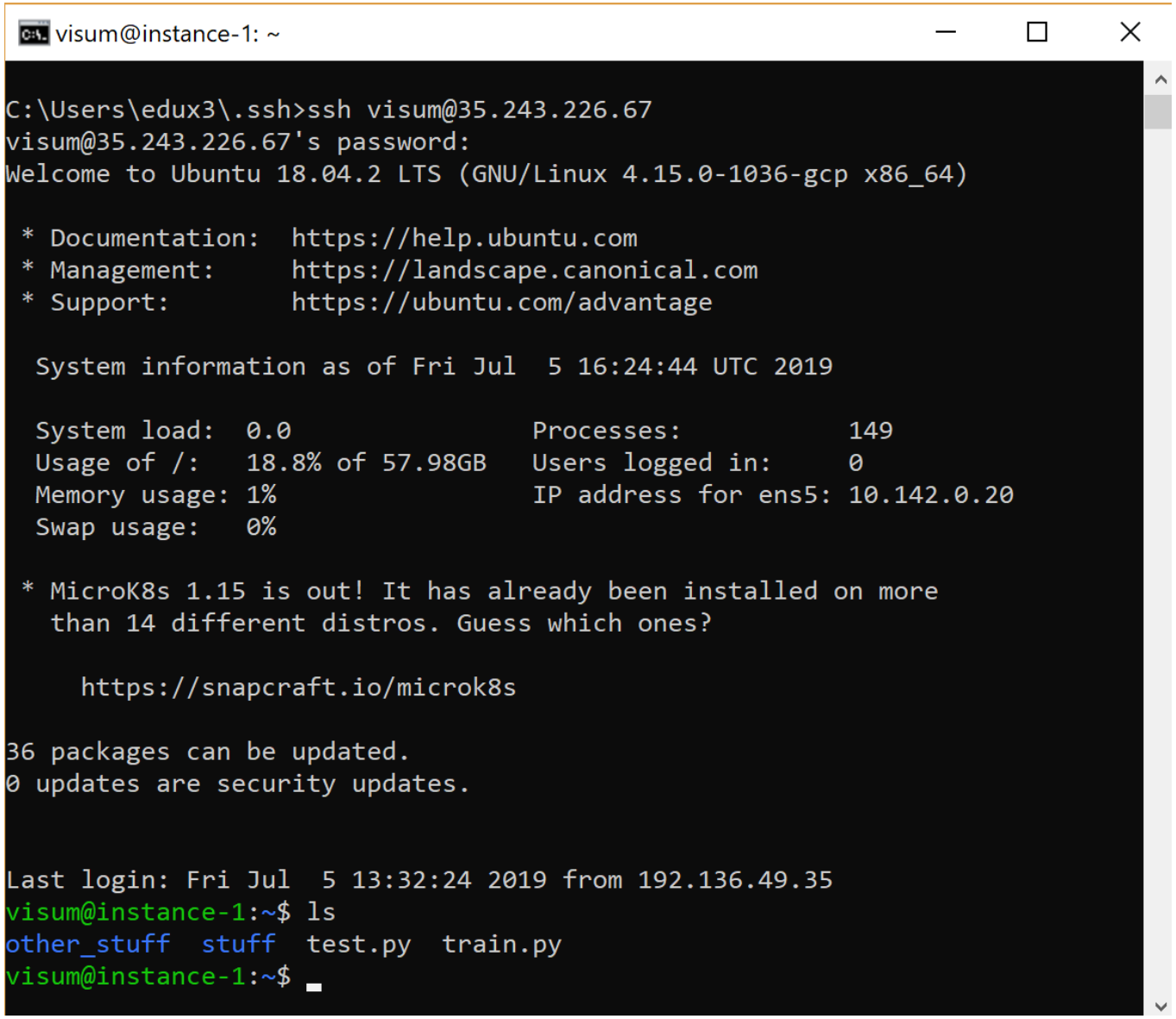
ssh visum@<ip_address>

User: visum

Pass: ********

All users share the same account

There is no GUI



USING YOUR MACHINE – EDITING FILES

You may map your machine's /home to a folder on your computer.

This will allow you to use fancy text editors, view images and plot results.

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

DAILY AND FINAL LEADERBOARD

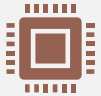
	Team	mAP@[0.5:0.95]
1	BASELINE	0.2420
2	team-1	nan
3	team-2	nan

Should run every day at around 12pm!

Runs on a subset of the test set

	Team	AP@[0.5:0.95] for unknown objects
1	BASELINE	0.0283
2	team-1	nan
3	team-2	nan

	Team	AP for empty car detection
1	BASELINE	0.6414
2	team-1	nan
3	team-2	nan



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 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)

DEVELOPMENT

DURATION

	July, 04	July, 05	July, 06	July, 07	July, 08	July, 09	July, 10	July, 11
9:00	Registration 08:45 - 09:45	Theoretical I 09:00 - 10:30	Exhibition 09:00 - 10:30		Theoretical I 09:00 - 10:30	Theoretical I 09:00 - 10:30	Theoretical I 09:00 - 10:30	Theoretical I 09:00 - 10:30
0:00	Welcome Session 09:45 - 10:30							
	Coffee Break	Coffee Break	Coffee Break		Coffee Break	Coffee Break	Coffee Break	Coffee Break
1:00	Theoretical I 11:00 - 12:30	Theoretical II 11:00 - 12:30	Exhibition 11:00 - 12:30		Theoretical II 11:00 - 12:30	Theoretical II 11:00 - 12:30	Theoretical II 11:00 - 12:30	Theoretical II 11:00 - 12:30
2:00								
3:00	Lunch 12:30 - 14:00	Lunch 12:30 - 14:00	Lunch 12:30 - 14:00		Lunch 12:30 - 14:00	Lunch 12:30 - 14:00	Lunch 12:30 - 14:00	Lunch 12:30 - 14:00
4:00	Theoretical II 14:00 - 15:30	"Hands-On" I 14:00 - 15:30	Debate 14:00 - 15:30	Social Programme 14:00 - 20:00	"Hands-On" 14:00 - 15:30	"Hands-On" I 14:00 - 15:30	"Hands-On" I 14:00 - 15:30	"Hands-On" I 14:00 - 15:30
5:00								
	Coffee Break	Coffee Break	Coffee Break		Coffee Break	Coffee Break	Coffee Break	Coffee Break
6:00	"Hands-On" 16:00 - 18:00	"Hands-On" II 16:00 - 17:30	Project launch 16:00 - 18:00		Project 16:00 - 17:30	"Hands-On" II 16:00 - 17:30	"Hands-On" II 16:00 - 17:30	"Hands-On" II 16:00 - 17:30
7:00								
8:00						Project 17:30 - 20:00	Project Hackathon (with dinner) 17:30 - 24:00	Project Hackathon (with dinner) 17:30 - 24:00
9:00					DSPT meetup 18:30 - 21:00			
0:00			Pint of Science 19:30 - 21:30	Dinner 20:00 - 22:00				
	Machine Learning & Computer Vision	Computer Vision with Deep Learning	Industry Day	Social Day	Deep Generative Models	Optimization & Constraint Programming	Visual Approaches for Robotic Control	Interpretability

WINNING THE PROJECT COMPETITION

- Teams are eligible to win the project competition if:
  - At the end of the competition they have a valid submission;
  - They have been chosen for an oral presentation at the last day of the summer school.
- The winner will be chosen by a panel of evaluators. They will look at:
  - The performance of your algorithm;
  - The ideas behind the model you designed;
  - How well you communicate your ideas during the final presentation.

FINDING HELP

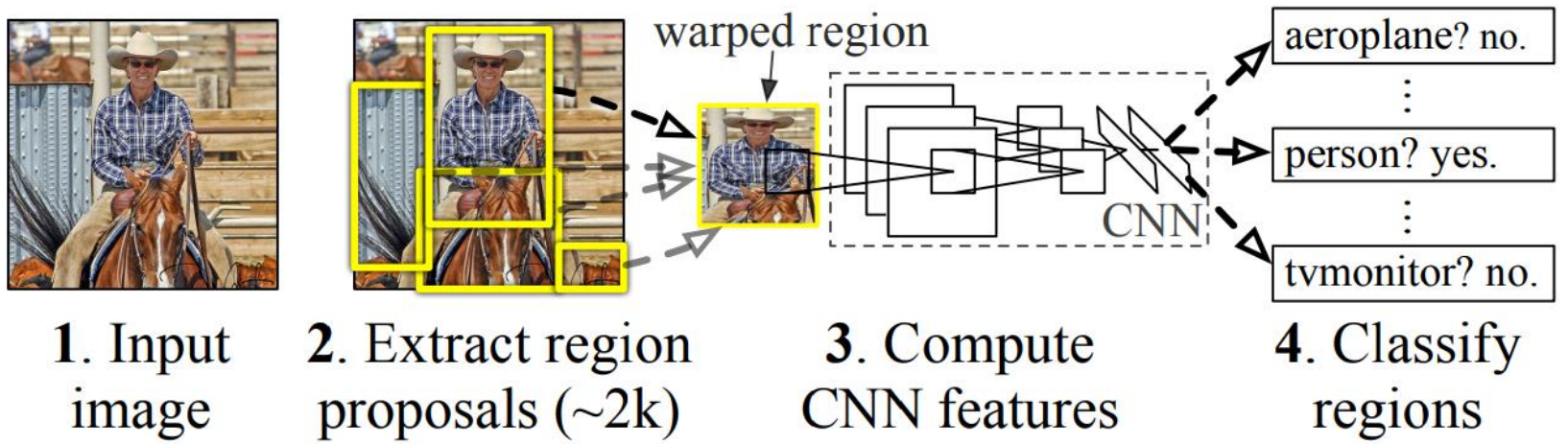
- If you have any questions regarding the project, relating to the rules, difficulties in understanding code or in setting up the google cloud machines please find one member of the project staff:

- Diogo – dpc@inesctec.pt
- Eduardo – eduardo.m.castro@inesctec.pt
- João – joao.t.pinto@inesctec.pt
- Ricardo – ricardo.j.araujo@inesctec.pt
- Wilson - wilson.j.silva@inesctec.pt

DEVELOPING A BASELINE

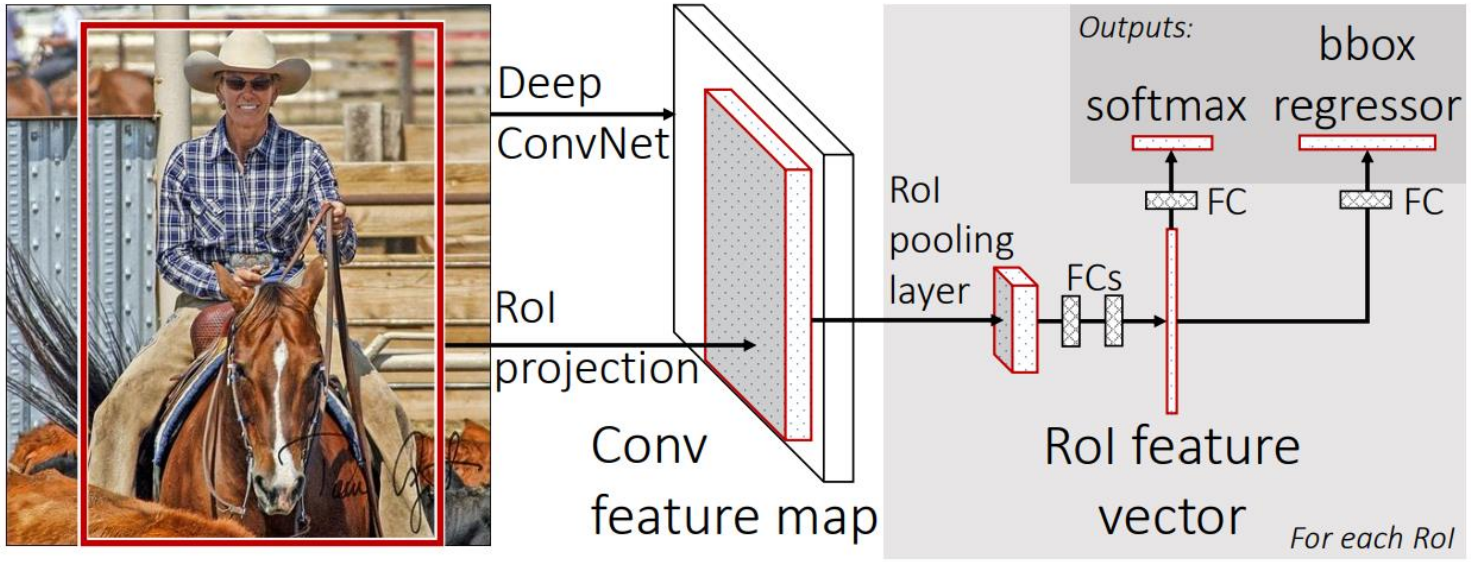
- Network goal:
  - Object detection: **localization** + **classification**

- State-of-the-art Architectures:
  - YOLO (https://arxiv.org/pdf/1506.02640.pdf)
  - RetinaNet (https://arxiv.org/abs/1708.02002)
  - **Faster R-CNN** (https://arxiv.org/pdf/1506.01497.pdf)



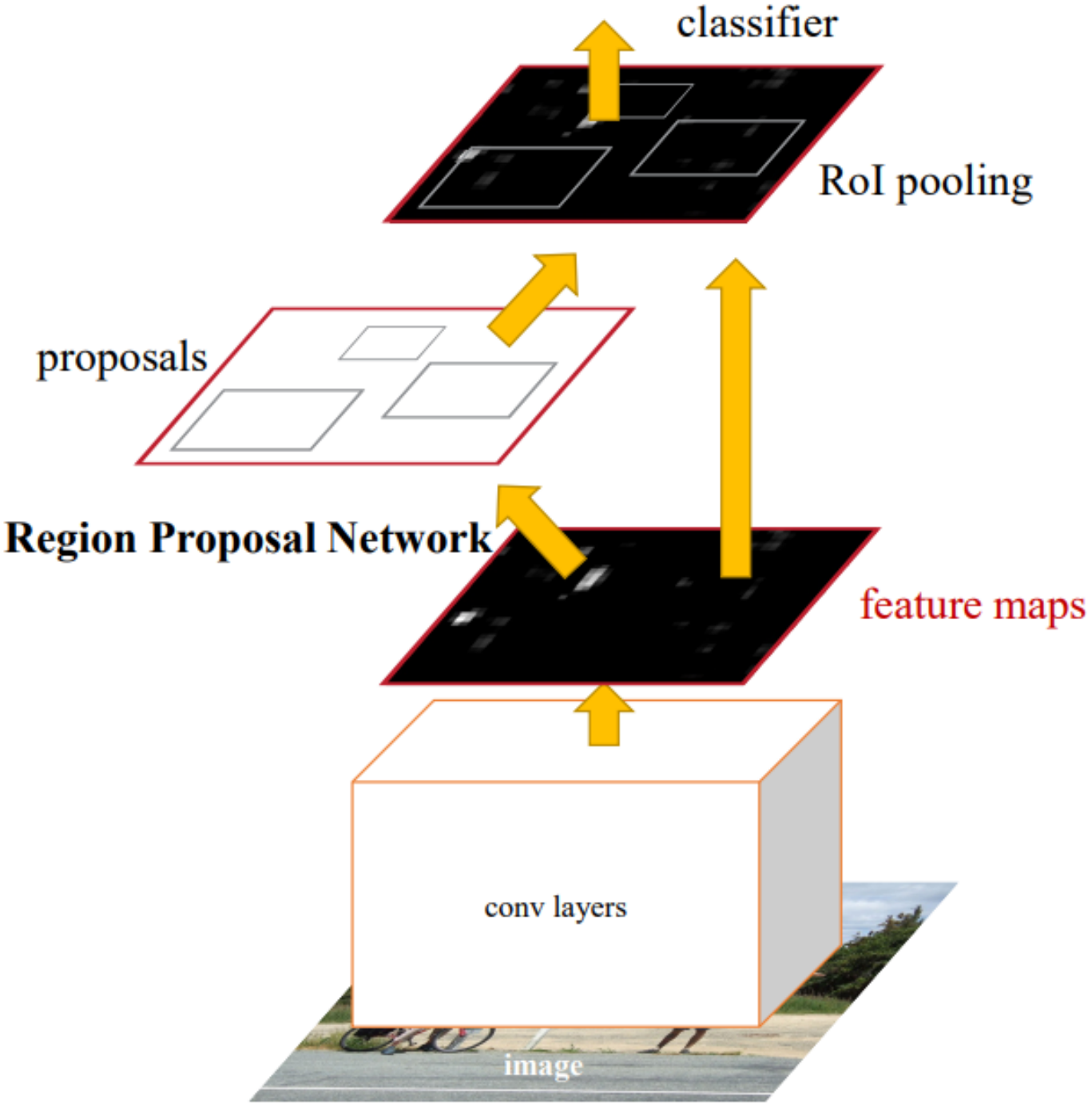
R-CNN

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



FAST R-CNN

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



THE BASELINE

FASTER R-CNN

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.
  - evaluate.py – Check your scores!

THE BASELINE SOLUTION

- train.py – Model.

```
backbone = torchvision.models.mobilenet_v2(pretrained=True).features
backbone.out_channels = 1280

anchor_generator = AnchorGenerator(sizes=((32, 64, 128, 256, 512),),
                                   aspect_ratios=((0.5, 1.0, 2.0),))

roi_pooler = torchvision.ops.MultiScaleRoIAlign(featmap_names=[0],
                                                output_size=7,
                                                sampling_ratio=2)


# put the pieces together inside a FasterRCNN model
model = FasterRCNN(backbone,
                   num_classes=10,
                   rpn_anchor_generator=anchor_generator,
                   box_roi_pool=roi_pooler)
```

THE BASELINE SOLUTION

- train.py – Split data into training and validation and define data augmentation.

```
# use our dataset and defined transformations
dataset = VisumData(args['data_path'], modality='rgb', transforms=get_transform(train=True))
dataset_val = VisumData(args['data_path'], modality='rgb', transforms=get_transform(train=False))

# split the dataset in train and test set
torch.manual_seed(1)
indices = torch.randperm(len(dataset)).tolist()
dataset = torch.utils.data.Subset(dataset, indices[:-100])
dataset_val = torch.utils.data.Subset(dataset_val, indices[-100:])

# define training and validation data loaders
data_loader = torch.utils.data.DataLoader(
    dataset, batch_size=2, shuffle=True, num_workers=0,
    collate_fn=utils.collate_fn)

data_loader_val = torch.utils.data.DataLoader(
    dataset_val, batch_size=2, shuffle=False, num_workers=0,
    collate_fn=utils.collate_fn)
```

```
# Data augmentation
def get_transform(train):
    transforms = []
    # converts the image, a PIL image, into a PyTorch Tensor
    transforms.append(T.ToTensor())
    if train:
        # during training, randomly flip the training images
        # and ground-truth for data augmentation
        transforms.append(T.RandomHorizontalFlip(0.5))
    return T.Compose(transforms)
```

THE BASELINE SOLUTION

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

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

model.to(device)

params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(params, lr=args['lr'],
                            momentum=0.9, weight_decay=args['l2'])

lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
                                               step_size=10,
                                               gamma=0.5)

for epoch in range(args['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 test dataset
    evaluator = evaluate(model, data_loader_val, device=device)

torch.save(model, args['model_path'])
```

THE BASELINE SOLUTION

- test.py – Load data and your model!

```
# Load datasets
test_data = VisumData(args['data_path'], 'rgb', mode='test', transforms=get_transform(False))

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

model = torch.load(args['model_path'])

test_loader = torch.utils.data.DataLoader(
    test_data, batch_size=1, shuffle=False, num_workers=4,
    collate_fn=utils.collate_fn)
```

THE BASELINE SOLUTION

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

```
predictions = list()
for i, (imgs, _, file_names) in enumerate(test_loader):
    # set the model to evaluation mode
    model.eval()
    with torch.no_grad():
        prediction = model(list(img.to(device) for img in imgs))

    boxes = np.array(prediction[0]['boxes'].cpu())
    labels = list(prediction[0]['labels'].cpu())
    scores = list(prediction[0]['scores'].cpu())

    nms_boxes, nms_labels = nms(boxes, labels, NMS_THR)

    for bb in range(len(nms_labels)):
        pred = np.concatenate((list(file_names), list(nms_boxes[bb, :])))  # bounding box
        if scores[bb] >= REJECT_THR:
            pred = np.concatenate((pred, [nms_labels[bb]]))  # object label
        else:
            pred = np.concatenate((pred, [-1]))  # Rejects to classify
        pred = np.concatenate((pred, [scores[bb]]))  # BEST CLASS SCORE
        pred = list(pred)
        predictions.append(pred)

with open(args['output'], 'w') as f:
    for pred in predictions:
        f.write("{},{},{},{},{},{},{}\n".format(pred[0], float(pred[1]), float(pred[2]), float(pred[3]),
```

THE BASELINE SOLUTION

- evaluate.py – Compute your scores by:
  - Calling metrics()
  - Running python evaluate.py

```
scores = metrics(ground_truth_file, pred_file, datase_dir)
print("Scores for:", pred_file, ":")
print("  mAP@[0.5:0.95] =", scores[0])
print("  AP@[0.5:0.95] unknown class =", scores[1])
print("  AP empty car =", scores[2])
with open("./scores.txt", "w") as file:
    writer = csv.writer(file)
    writer.writerow(scores)
```

```
# Returns:
# - MAP - detection task
# - AP for unknown objects - open set task
# - AP for empty image
def metrics(ground_truth_file, pred_file, datase_dir):
```

NEXT STEPS - RECOMMENDED

Information

- Go to https://github.com/visum-summerschool/visum-competition2019
- Read the visum_project_FAQ.pdf file

Team Registration

- Organize into a group of **three** and register your team

and give it an awesome name!!!

First result

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