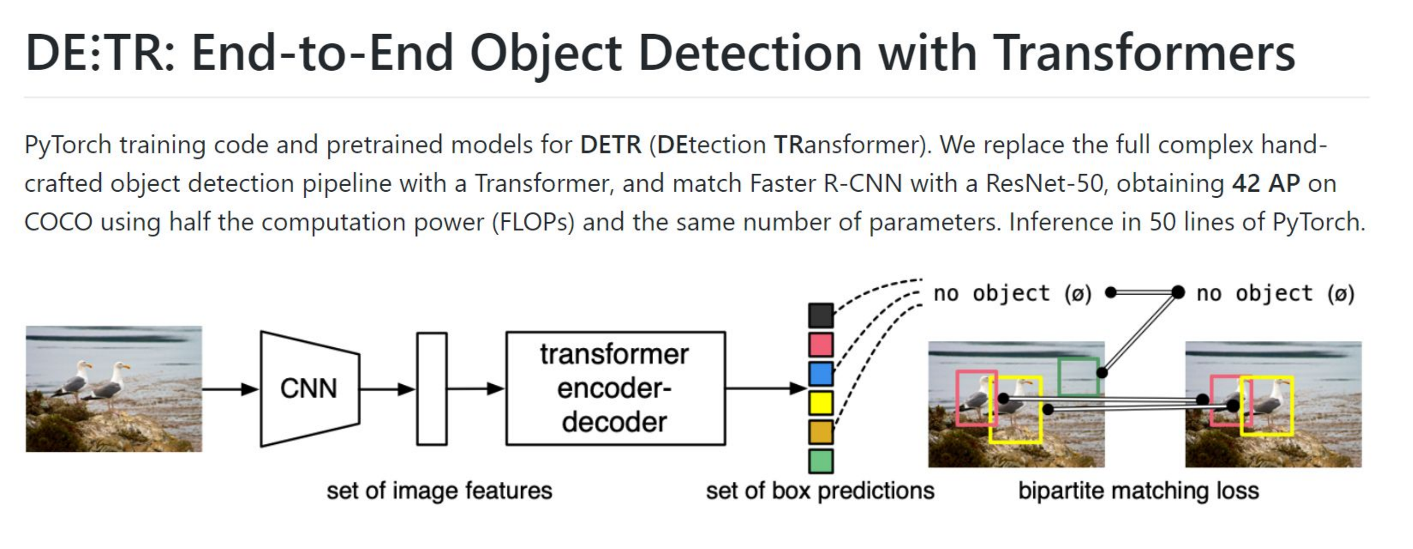
# DETR: End-to-End Object Detection with Transformers

By Sirak Ghebremusse
w251 Summer 2020

# Topic

- Breakdown of research paper
- Training of DETR model with mixed precision
- Provide Docker files for cloud training and Jetson inference

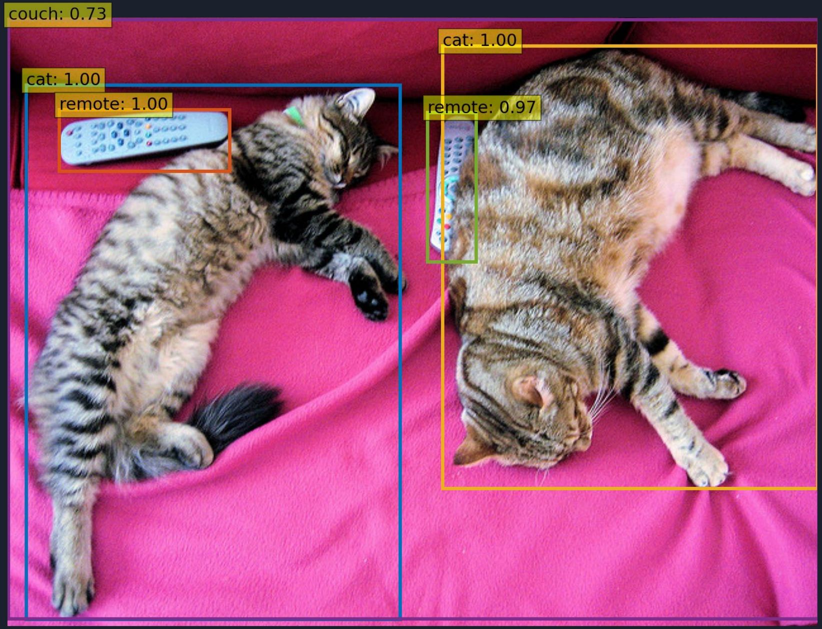


## DE:TR: End-to-End Object Detection with Transformers

PyTorch training code and pretrained models for **DETR** (**DE**tection **TR**ansformer). We replace the full complex hand-crafted object detection pipeline with a Transformer, and match Faster R-CNN with a ResNet-50, obtaining **42 AP** on COCO using half the computation power (FLOPs) and the same number of parameters. Inference in 50 lines of PyTorch.

# Dataset

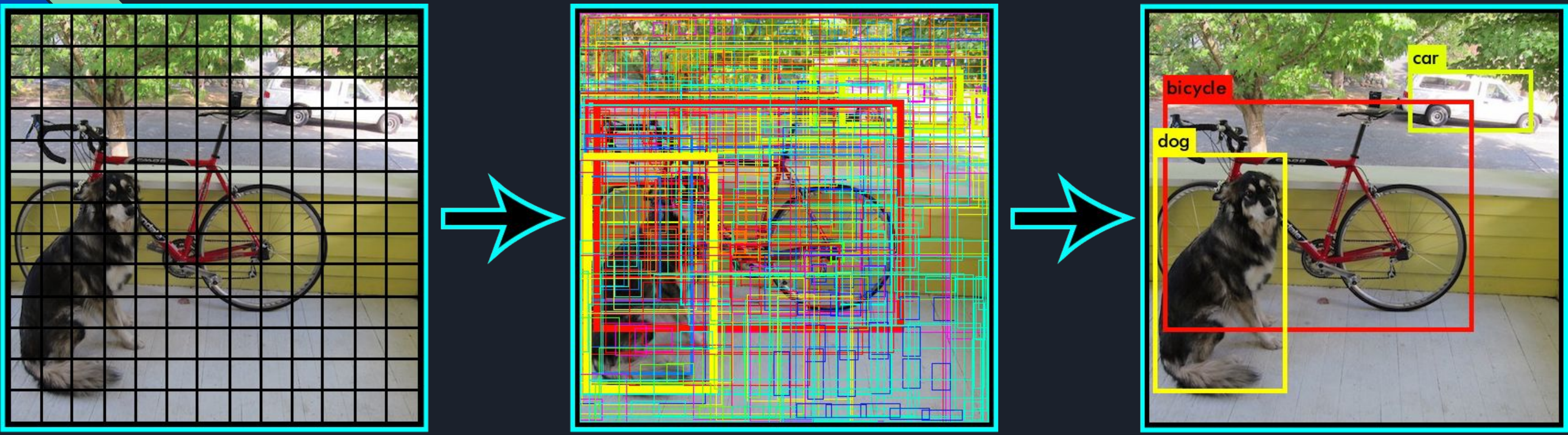
- Train and evaluation using the COCO dataset

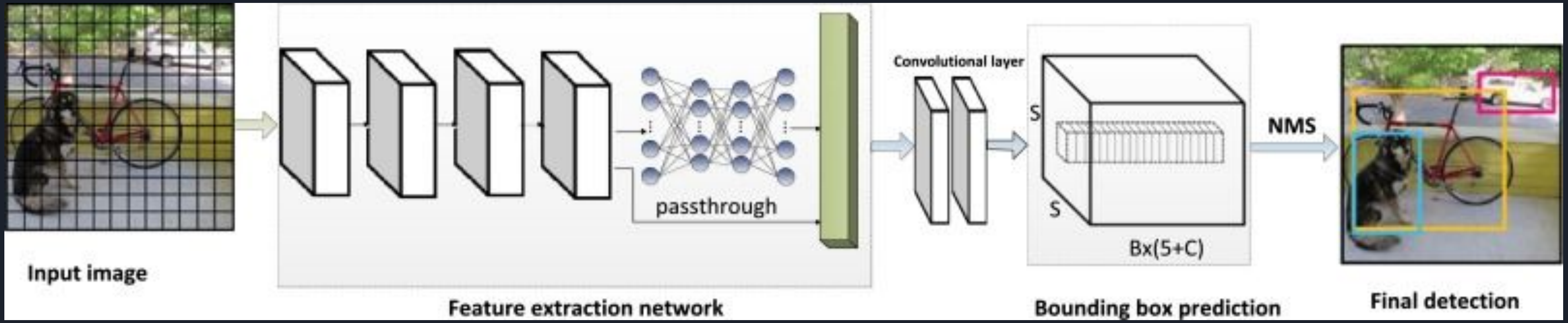


# Tools

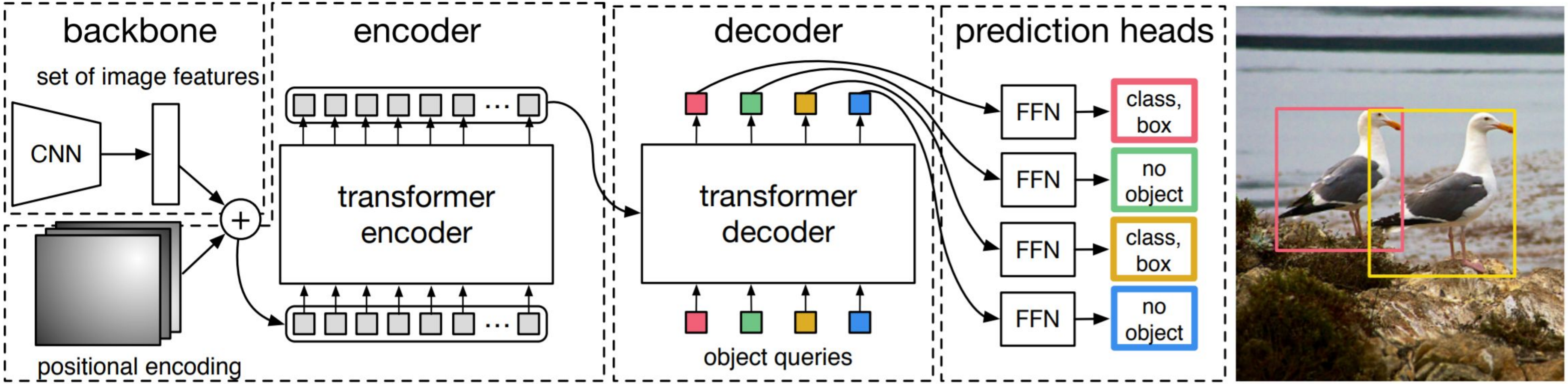
- Pytorch
- TensorRT
- OpenCV
- Docker
- Nvidia Apex
- Jetson Xavier NX
- IBM and AWS Cloud VMs

# DETR vs Yolo


car
bicycle
dog


passthrough
Convolutional layer
S
S
Bx(5+C)
NMS
Input image
Feature extraction network
Bounding box prediction
Final detection

# DETR vs Yolo


backbone
set of image features
CNN
positional encoding
encoder
transformer encoder
decoder
transformer decoder
object queries
prediction heads
FFN
FFN
FFN
FFN
class, box
no object
class, box
no object

# Attention for Detection

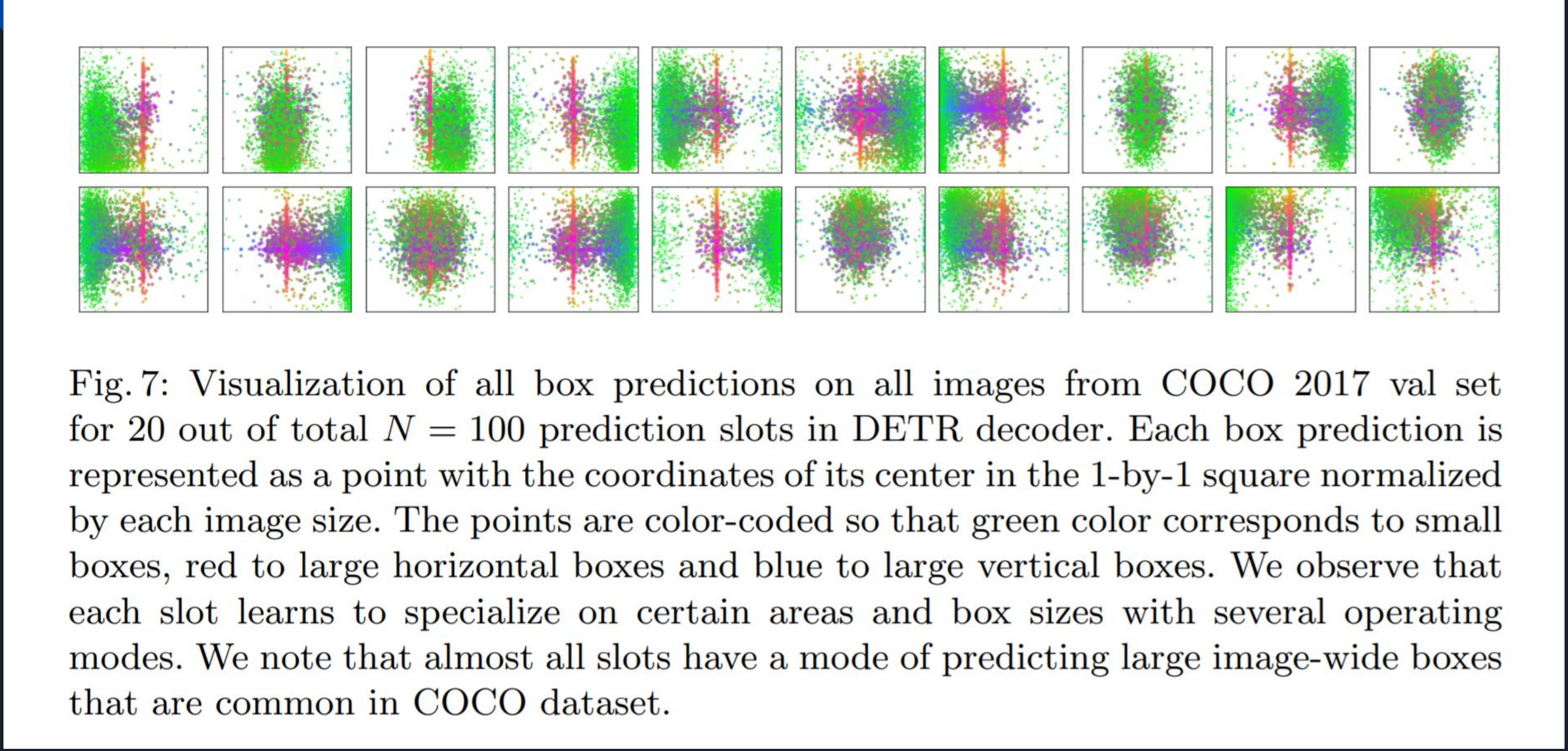
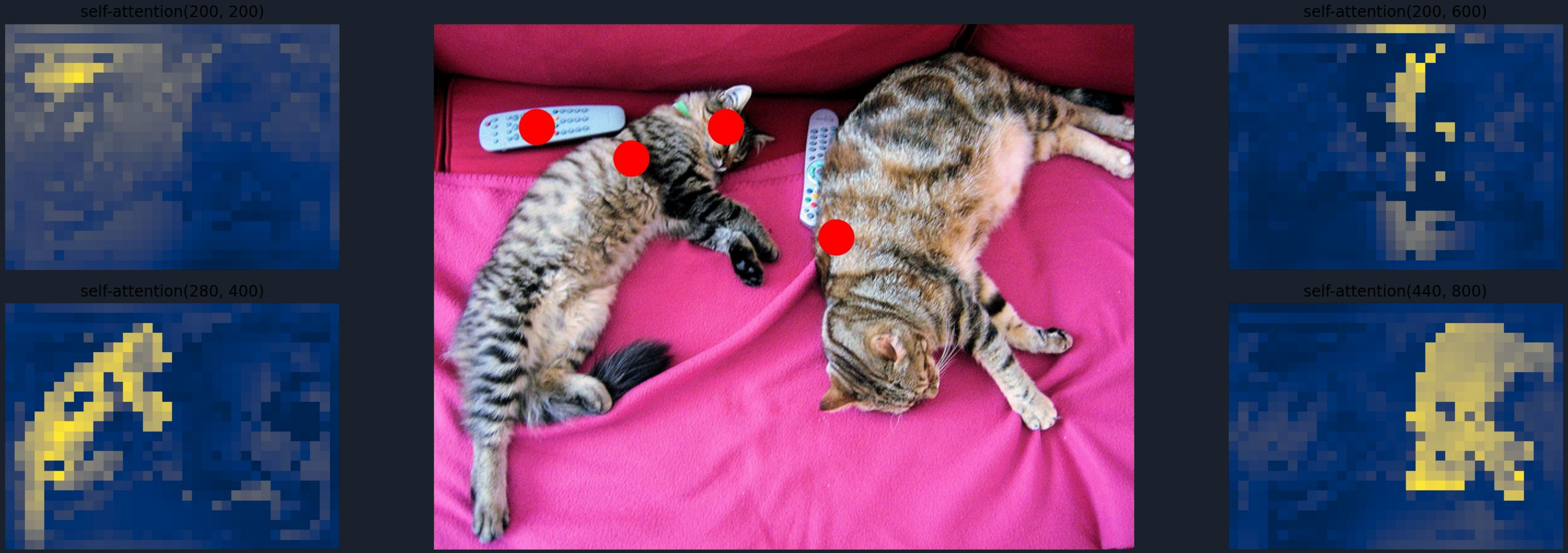


Fig. 7: Visualization of all box predictions on all images from COCO 2017 val set for 20 out of total $N = 100$ prediction slots in DETR decoder. Each box prediction is represented as a point with the coordinates of its center in the 1-by-1 square normalized by each image size. The points are color-coded so that green color corresponds to small boxes, red to large horizontal boxes and blue to large vertical boxes. We observe that each slot learns to specialize on certain areas and box sizes with several operating modes. We note that almost all slots have a mode of predicting large image-wide boxes that are common in COCO dataset.

# Attention for Segmentation



self-attention(200, 200)

self-attention(200, 600)

self-attention(280, 400)

self-attention(440, 800)

# Attention for Segmentation

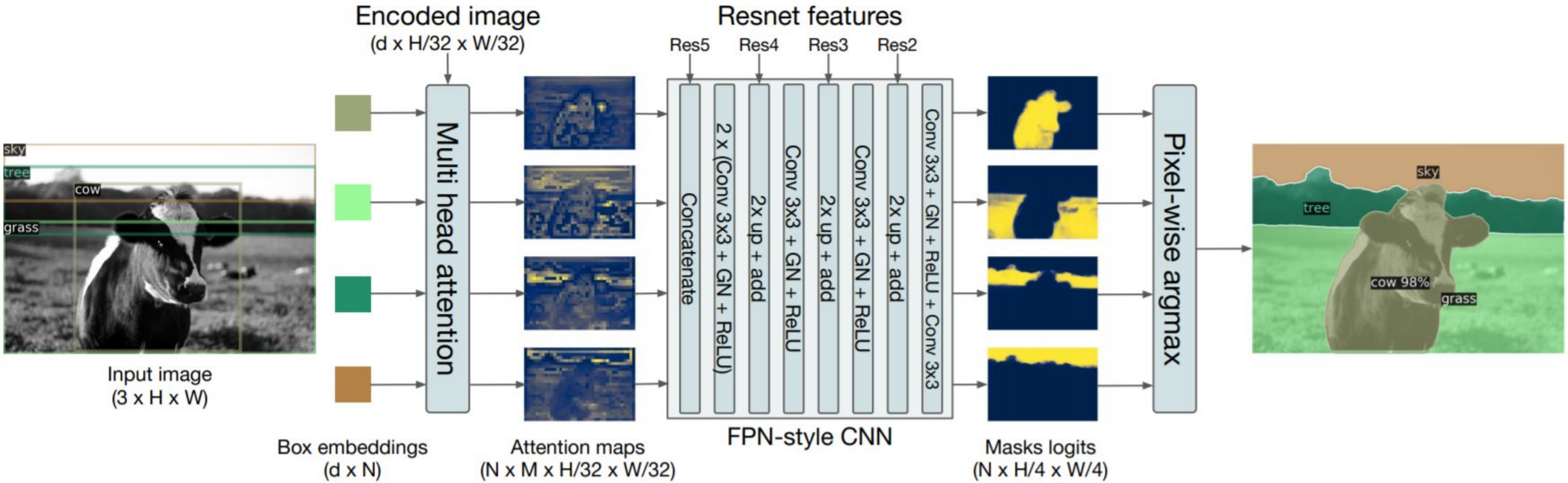
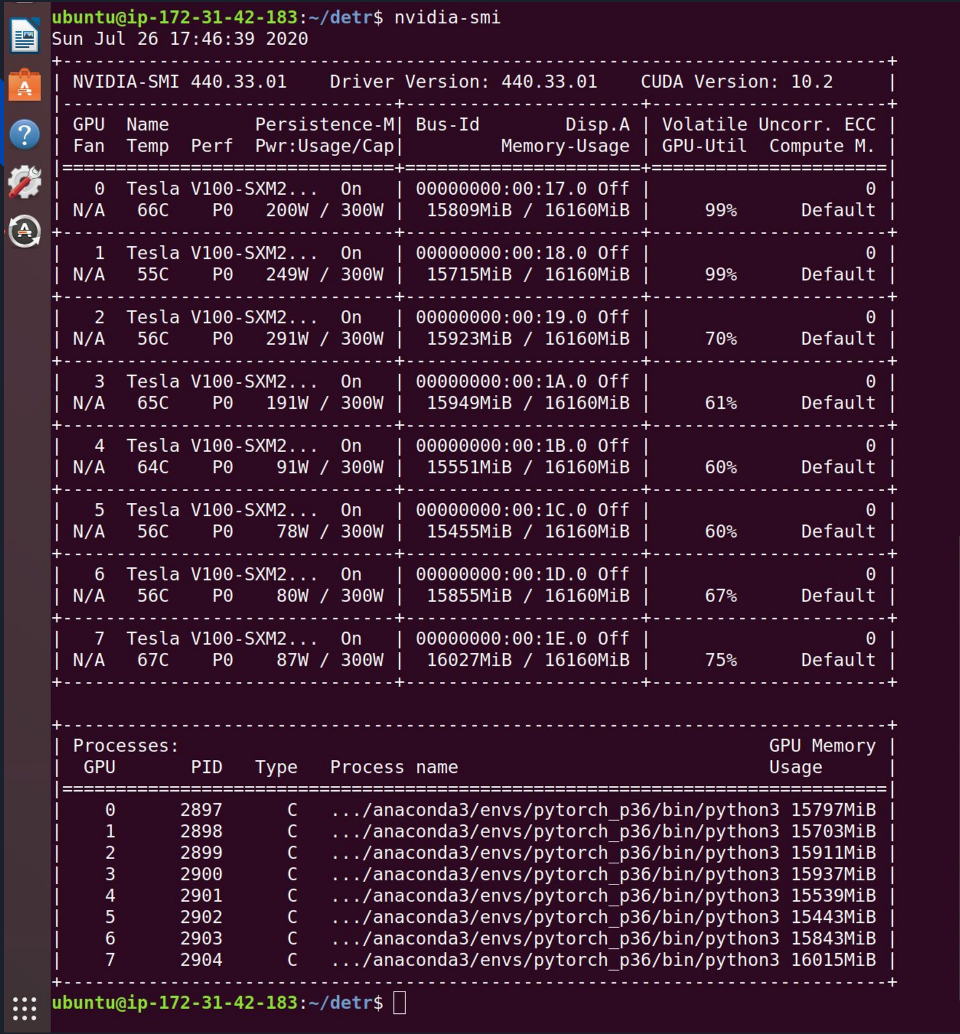


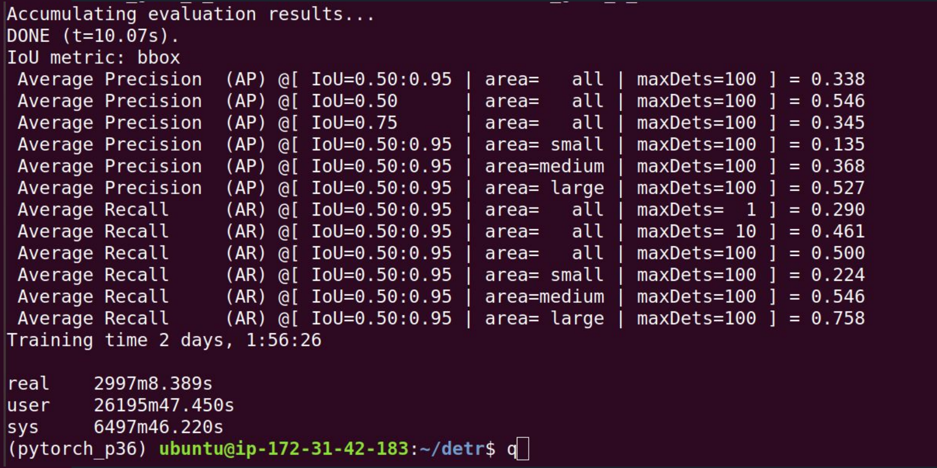
Fig. 8: Illustration of the panoptic head. A binary mask is generated in parallel for each detected object, then the masks are merged using pixel-wise argmax.

# Mixed Precision Training

- 2 Docker Containers :
  - Download Coco (18 GB)
  - Train DETR model (install Apex)
- Training Modifications
  - Include Apex commands in scripts
  - Changed backbone to ResNet-18
  - Optimization level set to 01, for 16 bit weights



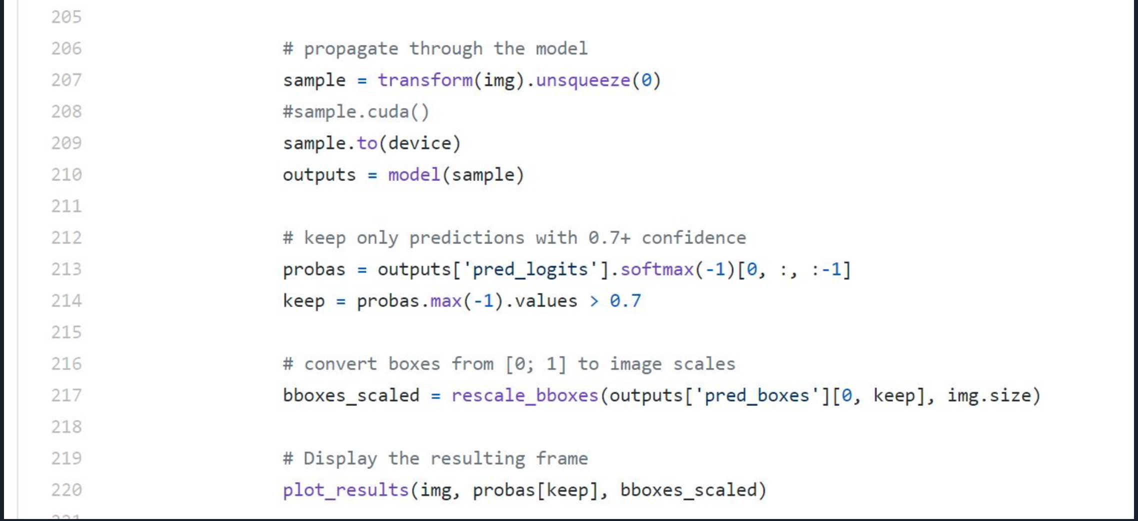
```
ubuntu@ip-172-31-42-183:~/detr$ nvidia-smi
Sun Jul 26 17:46:39 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.33.01    Driver Version: 440.33.01    CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:00:17.0 Off |                    0 |
| N/A   66C    P0   200W / 300W |  15809MiB / 16160MiB |     99%      Default |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000000:00:18.0 Off |                    0 |
| N/A   55C    P0   249W / 300W |  15715MiB / 16160MiB |     99%      Default |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  On   | 00000000:00:19.0 Off |                    0 |
| N/A   56C    P0   291W / 300W |  15923MiB / 16160MiB |     70%      Default |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  On   | 00000000:00:1A.0 Off |                    0 |
| N/A   65C    P0   191W / 300W |  15949MiB / 16160MiB |     61%      Default |
+-------------------------------+----------------------+----------------------+
|   4  Tesla V100-SXM2...  On   | 00000000:00:1B.0 Off |                    0 |
| N/A   64C    P0    91W / 300W |  15551MiB / 16160MiB |     60%      Default |
+-------------------------------+----------------------+----------------------+
|   5  Tesla V100-SXM2...  On   | 00000000:00:1C.0 Off |                    0 |
| N/A   56C    P0    78W / 300W |  15455MiB / 16160MiB |     60%      Default |
+-------------------------------+----------------------+----------------------+
|   6  Tesla V100-SXM2...  On   | 00000000:00:1D.0 Off |                    0 |
| N/A   56C    P0    80W / 300W |  15855MiB / 16160MiB |     67%      Default |
+-------------------------------+----------------------+----------------------+
|   7  Tesla V100-SXM2...  On   | 00000000:00:1E.0 Off |                    0 |
| N/A   67C    P0    87W / 300W |  16027MiB / 16160MiB |     75%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      2897      C   .../anaconda3/envs/pytorch_p36/bin/python3 15797MiB |
|    1      2898      C   .../anaconda3/envs/pytorch_p36/bin/python3 15703MiB |
|    2      2899      C   .../anaconda3/envs/pytorch_p36/bin/python3 15911MiB |
|    3      2900      C   .../anaconda3/envs/pytorch_p36/bin/python3 15937MiB |
|    4      2901      C   .../anaconda3/envs/pytorch_p36/bin/python3 15539MiB |
|    5      2902      C   .../anaconda3/envs/pytorch_p36/bin/python3 15443MiB |
|    6      2903      C   .../anaconda3/envs/pytorch_p36/bin/python3 15843MiB |
|    7      2904      C   .../anaconda3/envs/pytorch_p36/bin/python3 16015MiB |
+-----------------------------------------------------------------------------+
ubuntu@ip-172-31-42-183:~/detr$
```



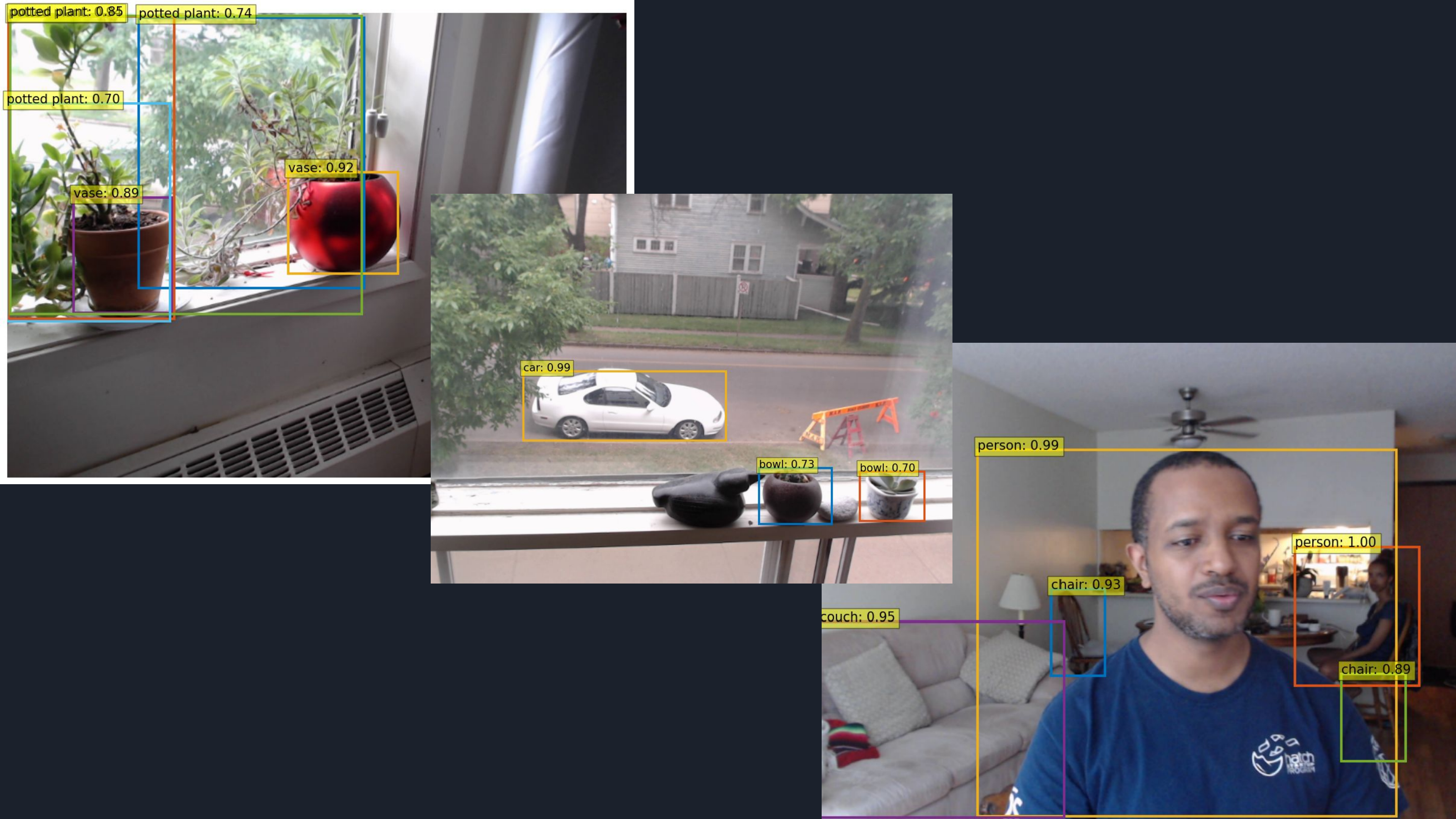
```
Accumulating evaluation results...
DONE (t=10.07s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.338
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.546
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.345
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.135
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.368
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.527
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.290
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.461
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.500
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.224
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.546
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.758
Training time 2 days, 1:56:26

real    2997m8.389s
user    26195m47.450s
sys     6497m46.220s
(pytorch_p36) ubuntu@ip-172-31-42-183:~/detr$ q
```

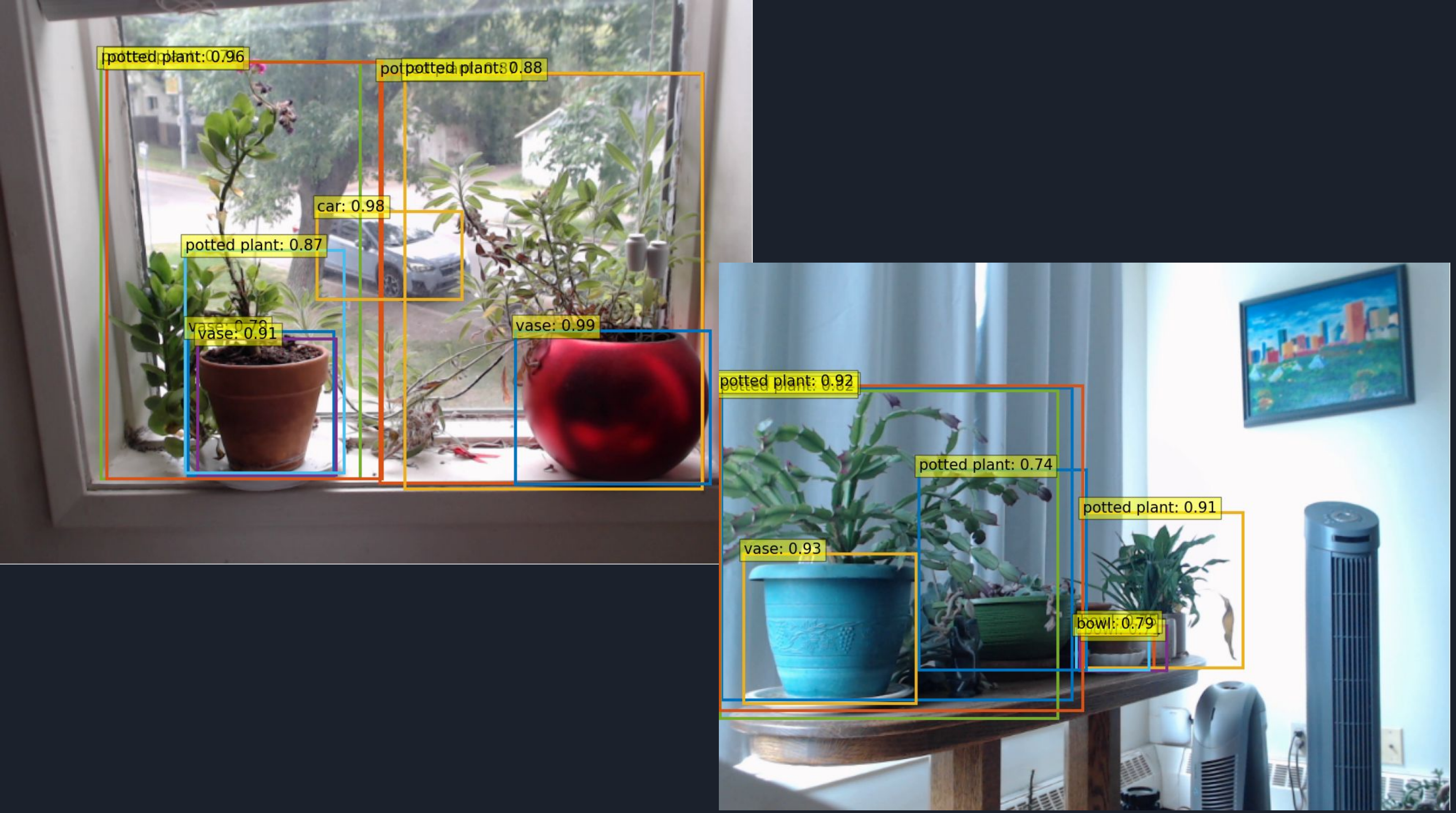
# Mixed Precision Inference

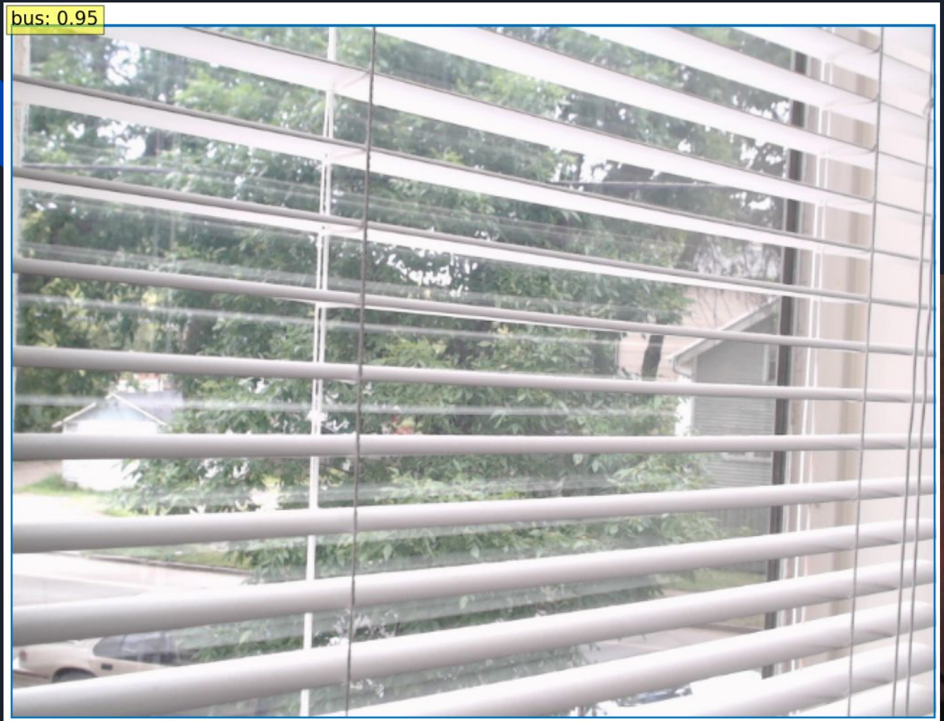
- Torch2TensorRT container
- Inference container for TensorRT or Pytorch model



```

                # propagate through the model
                sample = transform(img).unsqueeze(0)
                #sample.cuda()
                sample.to(device)
                outputs = model(sample)

                # keep only predictions with 0.7+ confidence
                probas = outputs['pred_logits'].softmax(-1)[0, :, :-1]
                keep = probas.max(-1).values > 0.7

                # convert boxes from [0; 1] to image scales
                bboxes_scaled = rescale_bboxes(outputs['pred_boxes'][0, keep], img.size)

                # Display the resulting frame
                plot_results(img, probas[keep], bboxes_scaled)
```


potted plant: 0.85
potted plant: 0.74
potted plant: 0.70
vase: 0.92
vase: 0.89
car: 0.99
bowl: 0.73
bowl: 0.70
person: 0.99
person: 1.00
chair: 0.93
couch: 0.95
chair: 0.89


potted plant: 0.96
potted plant: 0.88
car: 0.98
potted plant: 0.87
vase: 0.70
vase: 0.91
vase: 0.99
potted plant: 0.92
potted plant: 0.82
potted plant: 0.74
potted plant: 0.91
vase: 0.93
bowl: 0.79


bus: 0.95

TITAN

# Challenges

- Torch version and GPU driver mismatch on IBM
- APEX installation required NGC containers
- Unable to fully utilize AWS GPUs (~70%)
- Torch2TensorRT not very intuitive
-

# Future Goals

- Train for Image segmentation
- Multi-server training
- Cast layers to below 16-bit precision
- Ensure it runs on Tensor cores on Jetson NX
- Smaller vision backbone!

# Thank you!

# Links

https://ai.facebook.com/blog/end-to-end-object-detection-with-transformers/

https://ai.facebook.com/research/publications/end-to-end-object-detection-with-transformers/

https://www.youtube.com/watch?v=T35ba_VXkMY&t=1464s

https://colab.research.google.com/github/facebookresearch/detr/blob/colab/notebooks/detr_demo.ipynb

https://colab.research.google.com/github/facebookresearch/detr/blob/colab/notebooks/detr_demo.ipynb

https://colab.research.google.com/github/facebookresearch/detr/blob/colab/notebooks/DETR_panoptic.ipynb

Github Repo: https://github.com/sirakzg/detr