Damage Detection of Cars and Repair cost Estimation!!

It’s more than just image segmentation

Using the best image segmentation model for instance segmentation to detect damaged areas on cars is not new and has been done before and here I’m also doing the same thing but with an extension of estimating repair costs.

Therefore I intend to visit the instance segmentation part in brief and discuss the repair costs estimation and future work in detail.

The Business Problem

Claiming Insurance for your damaged vehicle is a hassle not only for the insured but for the insurer too. It involves paperwork and inspection that consumes time, yet it is necessary to avoid fraudulent claims. The whole process takes at least a week to finalize the repair cost, after inspection, from the mechanic.

What if we estimate the repair cost of damages from pictures of the damaged areas through machine learning using the past claims data for training that the insurance company already has. This will reduce inspection time and consequently claims can be processed faster.

Though it is absolutely **necessary** to perform thorough inspection for **severe** damages where **internal** parts are affected, **minor/moderate** damages that has only affected the **external** surface of the vehicle can avoid it by estimating the repair cost and save a lot of time both for the insured and the insurer.

Dataset

I got the dataset from Kaggle Datasets here. It contains total 78 images of damaged cars of which 59 are for training, 11 for validation and remaining 8 for testing.

There are two types of annotations per image:

1. All the damaged areas are annotated, with a label **damage**

2. All the parts of car are annotated with labels - **front bumper**, **rear bumper**, **headlamp**, **door** and **hood**.


damage
damage
damage
damage
damage

Annotations of damaged area (left) and car parts (right)

Modifying the damage annotations

Now, to help train my repair cost estimation model I modified the **damage annotations** into three categories: **minor**, **moderate** and **severe**.

It was a manual process where I looked at each car image with damages and edited the annotation json file into 3 categories using the following method:

- Scratches - classified as **minor** damage.
- Dents - classified as **moderate** damage.
- Broken part - classified as **severe** damage.

Now after this modification the updated annotations look like:

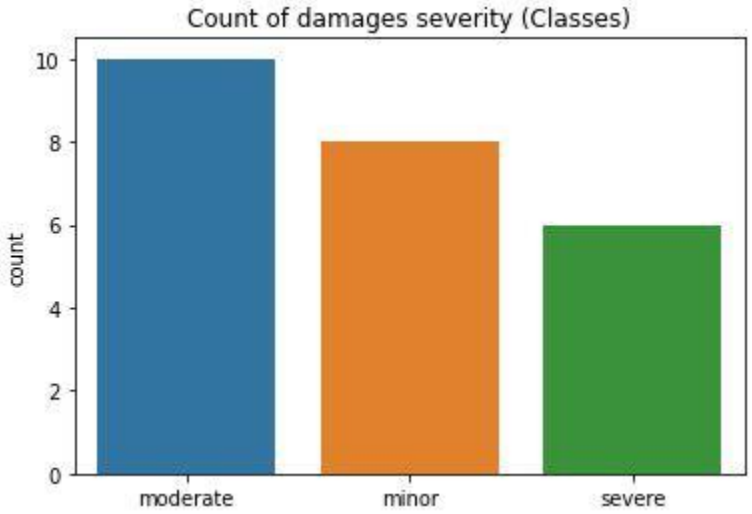
Annotations of damage type (image by author)

EDA

Just to recap, now I have two annotations in total, one for **damage types** (modified above) and another for **parts of car**.

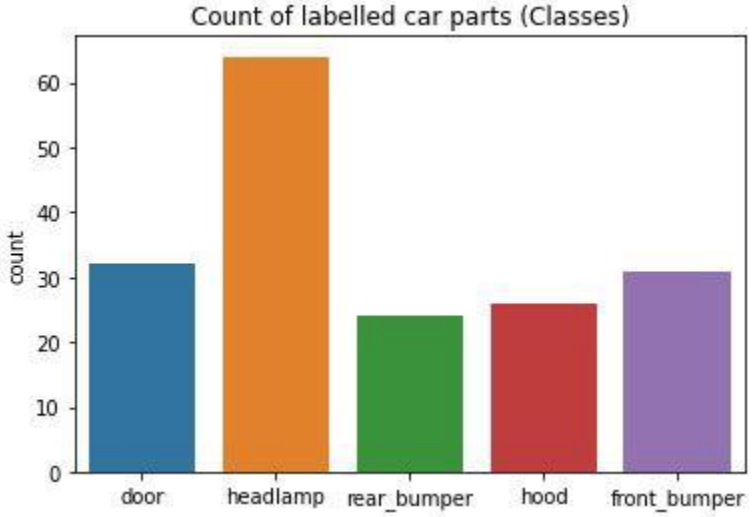
Proper EDA gave me the following results:

- Damage type annotations have **moderate** damages the most followed by minor and severe damages.



Count of damage classes in train data

- Car parts annotations have the **headlamp** annotations the most and **rear bumper** annotations the least.



Count of cart part classes in train data

- All car parts are not annotated, E.g.- **front fender, rear fender, trunk, windshield** etc. Hence the image segmentation model won't be able to detect those parts.
- Training data has only 59 images, which is not enough, so we need a proper pretrained backbone architecture for image segmentation model (Mask R-CNN) to facilitate transfer learning.

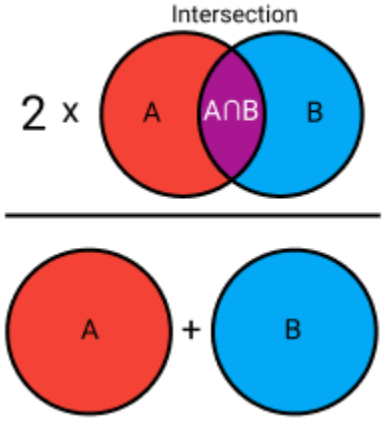
Creating the training data

Just a brief introduction to the one of the most import feature in our training data, **dice coefficient.**

Dice Coefficient

Dice Coefficient is the area of overlap between two masks.

DSC = 2 * (the Area of Overlap) / (the total number of pixels in both masks)



Formula for dice coefficient

Now using the above formula and the annotations we are provided with, we can find ***how much percentage of area is damaged in a particular car part***

Ex: In the above formula, let **A** be segmentation mask of hood of the car, and **B** be segmentation mask of moderate damaged area in the hood, then putting the values in the formula we get:

DSC = 2 * (the Area of Overlap between hood mask and moderate damage mask) / (the total number of pixels in both masks) = 0.2345 (let)

*then we can say that **23%** of the hood is **moderately** damaged and '**hood_dice**' feature gets a value of 0.2345.*

Similarly we can do this iteratively for every damage mask in the car image. The final training dataset looked something like this:

mp_dice	rear_bump_dice	door_dice	hood_dice	front_bump_dice	unknown	minor	moderate	severe	price	
1	0.0	0.000000	0.004600	0.000000	0.000000	0	0	1	0	500
2	0.0	0.000000	0.002410	0.000000	0.000000	0	0	1	0	50
3	0.0	0.041677	0.000000	0.000000	0.000000	0	1	0	0	400
4	0.0	0.026820	0.000000	0.000000	0.000000	0	1	0	0	400
5	0.0	0.000000	0.000000	0.000000	0.000000	1	0	1	0	250
6	0.0	0.000000	0.000000	0.001841	0.070522	0	0	0	1	1700

Modelling Approach

Intuitively there are total 3 models that we need to train:

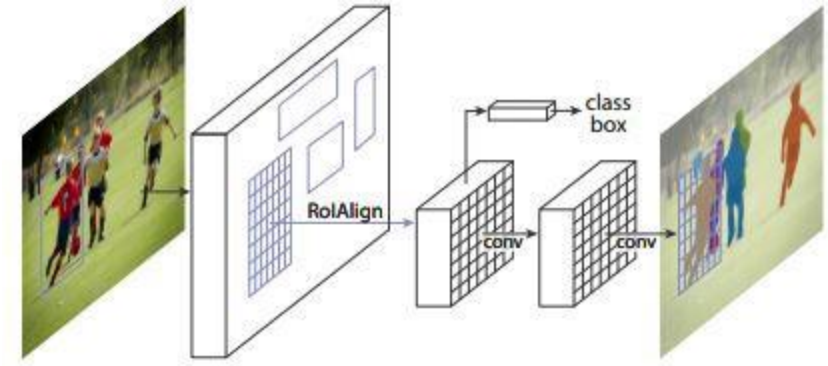
1. First model to train the above dataset for estimating the price, lets call it **model 1**.
2. Second for detecting the **damaged areas** with their classes (**minor**, **moderate** and **severe**), lets call it **model 2**.
3. Third for detecting the **parts of car** with their classes (**front bumper**, **rear bumper**, **headlamp**, **door** and **hood**), lets call it **model 3**.

Model **1** can be trained using a basic logistic regression model, but unfortunately we have very less data for training and hence I would be calculating **mean** price using **group by** from identical rows in test data.

We need an Image segmentation model for **second** and **third** models to help generate the test data.

Mask R-CNN

There are many models available for instance segmentation. Here I have used **Mask R-CNN** which gives decent results considering the size of dataset we have.



Architecture of Mask R-CNN

The architecture contains the following components:

- **Backbone:** This is a standard convolutional neural network (typically, ResNet50 or ResNet101) that serves as a feature extractor.
- **Region Proposal Network (RPN):** It is a lightweight neural network that scans the image in a sliding-window fashion and finds areas that contain objects.
- **ROI Classifier:** This network is deeper and has the capacity to classify ROIs to specific classes (person, car, chair, etc.).
- **Bounding Box Regressor:** Very similar to how it's done in the RPN, and its purpose is to further refine the location and size of the bounding box to encapsulate the object.
- **ROI Pooling:** ROI pooling refers to cropping a part of a feature map and resizing it to a fixed size. The authors suggest a method they named **ROIAlign**, in which they

sample the feature map at different points and apply a bilinear interpolation.

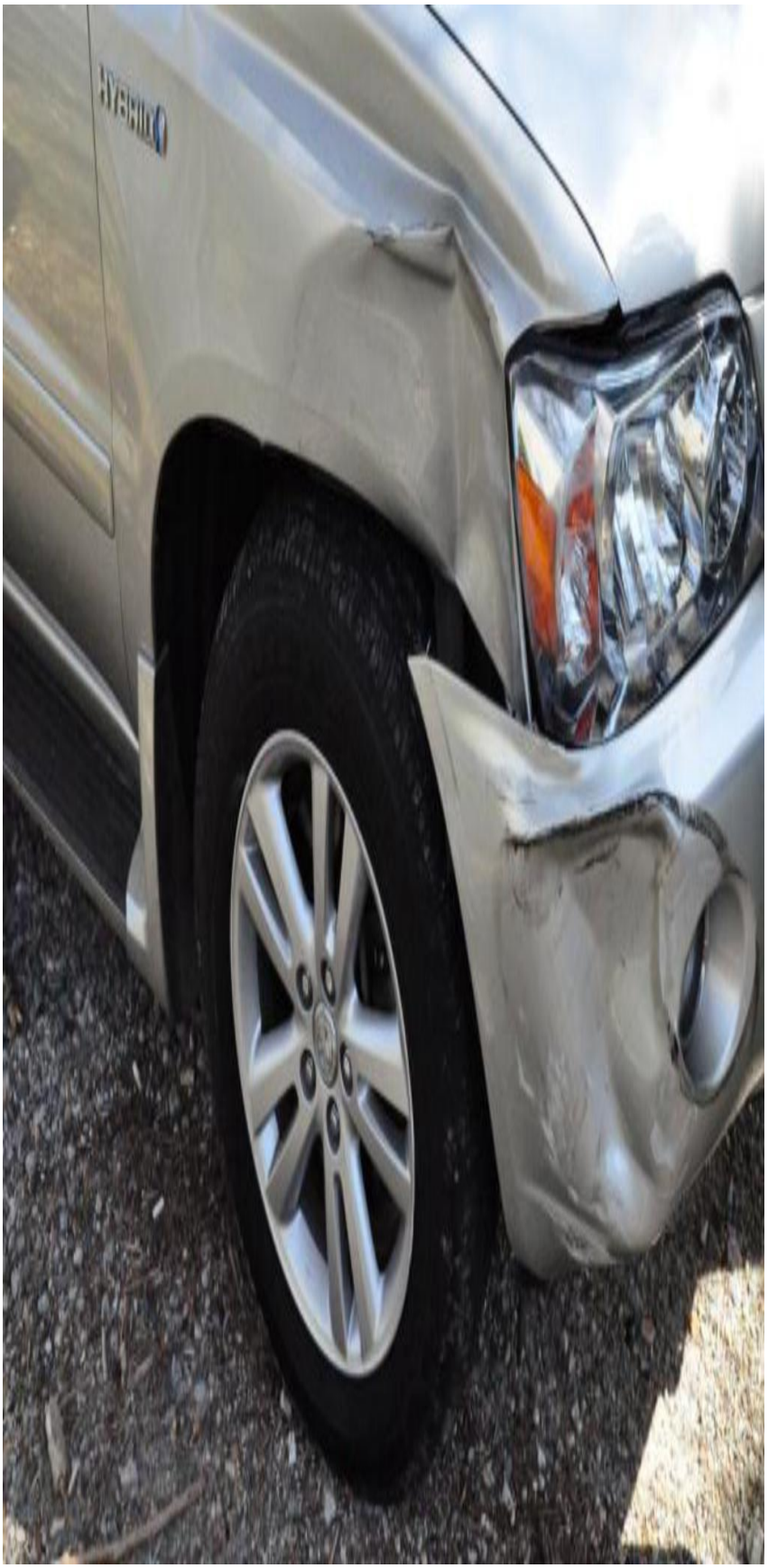
- **Segmentation Masks:** The mask branch is a convolutional network that takes the positive regions selected by the ROI classifier and generates masks for them.

I used a pre-built library called **Detectron2** to train the Mask R-CNN Model.

Model Training and Performance

1. **Damage type detection:** Model **2** gave me an Average Precision (AP) of 8.89 over all classes for segmentation masks. See notebook snippet below:

Training output of Model 2 (damage type detection)



Predictions of damage type detection model

2. **Car part detection:** Model **3** gave me an Average Precision (AP) of 34.18 over all classes for segmentation masks. See notebook snippet below:

Training output of Model 3 (car part detection)

```
register_coco_instances("car_part_train", {},
os.path.join(COCO_DIR,train_dir,"COCO_mul_train_annos.json"),
os.path.join(COCO_DIR,train_dir))
register_coco_instances("car_part_val", {},
os.path.join(COCO_DIR,val_dir,"COCO_mul_val_annos.json"),
os.path.join(COCO_DIR,val_dir))

car_part_dataset_dicts = DatasetCatalog.get("car_part_train")
car_part_metadata_dicts = MetadataCatalog.get("car_part_train")
```

In [ ]:

```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-
InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
#cfg.merge_from_file('configs/COCO-
InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml')
cfg.DATASETS.TRAIN = ("car_part_train",)
cfg.DATASETS.TEST = ("car_part_val",)
cfg.INPUT.CROP.ENABLED = True
cfg.DATALOADER.NUM_WORKERS = 1
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-
InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")  # Let training initialize
from model zoo
cfg.SOLVER.IMS_PER_BATCH = 1
cfg.SOLVER.BASE_LR = 0.001  # pick a good LR
cfg.SOLVER.WARMUP_ITERS = 800
cfg.SOLVER.MAX_ITER = 2500 #adjust up if val mAP is still rising, adjust down
if overfit
cfg.SOLVER.STEPS = (600, 2000,)
cfg.SOLVER.GAMMA = 0.05
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 256   # faster, and good enough
for this  dataset (default: 512)
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 5  # only has one class (damage)
cfg.MODEL.RETINANET.NUM_CLASSES = 5 # only has one class (damage)
cfg.TEST.EVAL_PERIOD = 400
cfg.SOLVER.CHECKPOINT_PERIOD = 400



# Clear any logs from previous runs
#TODO add timestamp to logs
!rm -rf cfg.OUTPUT_DIR


os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = CocoTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
[05/19 16:17:44 d2.engine.defaults]: Model:
GeneralizedRCNN(
  (backbone): FPN(
    (fpn_lateral2): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), paddin
g=(1, 1))
    (fpn_lateral3): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), paddin
g=(1, 1))
    (fpn_lateral4): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output4): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), paddin
g=(1, 1))
    (fpn_lateral5): Conv2d(2048, 256, kernel_size=(1, 1), stride=(1, 1))
    (fpn_output5): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), paddin
g=(1, 1))
```

```
    (top_block): LastLevelMaxPool()
    (bottom_up): ResNet(
      (stem): BasicStem(
        (conv1): Conv2d(
          3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=Fals
e
          (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
        )
      )
      (res2): Sequential(
        (0): BottleneckBlock(
          (shortcut): Conv2d(
            64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv1): Conv2d(
            64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
          )
          (conv2): Conv2d(
            64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse
            (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
          )
          (conv3): Conv2d(
            64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
        )
        (1): BottleneckBlock(
          (conv1): Conv2d(
            256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
          )
          (conv2): Conv2d(
            64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse
            (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
          )
          (conv3): Conv2d(
            64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
        )
        (2): BottleneckBlock(
          (conv1): Conv2d(
            256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
          )
          (conv2): Conv2d(
            64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=F
alse
            (norm): FrozenBatchNorm2d(num_features=64, eps=1e-05)
```

```
          )
          (conv3): Conv2d(
            64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
        )
      )
      (res3): Sequential(
        (0): BottleneckBlock(
          (shortcut): Conv2d(
            256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
          (conv1): Conv2d(
            256, 128, kernel_size=(1, 1), stride=(2, 2), bias=False
            (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
          )
          (conv2): Conv2d(
            128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
          )
          (conv3): Conv2d(
            128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
        )
        (1): BottleneckBlock(
          (conv1): Conv2d(
            512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
          )
          (conv2): Conv2d(
            128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
          )
          (conv3): Conv2d(
            128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
        )
        (2): BottleneckBlock(
          (conv1): Conv2d(
            512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
          )
          (conv2): Conv2d(
            128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
          )
          (conv3): Conv2d(
```

```
            128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
        )
        (3): BottleneckBlock(
          (conv1): Conv2d(
            512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
          )
          (conv2): Conv2d(
            128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=128, eps=1e-05)
          )
          (conv3): Conv2d(
            128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
        )
      )
      (res4): Sequential(
        (0): BottleneckBlock(
          (shortcut): Conv2d(
            512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False
            (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
          )
          (conv1): Conv2d(
            512, 256, kernel_size=(1, 1), stride=(2, 2), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv2): Conv2d(
            256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv3): Conv2d(
            256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
          )
        )
        (1): BottleneckBlock(
          (conv1): Conv2d(
            1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv2): Conv2d(
            256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv3): Conv2d(
            256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
```

```
          )
        )
        (2): BottleneckBlock(
          (conv1): Conv2d(
            1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv2): Conv2d(
            256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv3): Conv2d(
            256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
          )
        )
        (3): BottleneckBlock(
          (conv1): Conv2d(
            1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv2): Conv2d(
            256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv3): Conv2d(
            256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
          )
        )
        (4): BottleneckBlock(
          (conv1): Conv2d(
            1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv2): Conv2d(
            256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv3): Conv2d(
            256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
          )
        )
        (5): BottleneckBlock(
          (conv1): Conv2d(
            1024, 256, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv2): Conv2d(
```

```
            256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=256, eps=1e-05)
          )
          (conv3): Conv2d(
            256, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=1024, eps=1e-05)
          )
        )
      )
      (res5): Sequential(
        (0): BottleneckBlock(
          (shortcut): Conv2d(
            1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False
            (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
          )
          (conv1): Conv2d(
            1024, 512, kernel_size=(1, 1), stride=(2, 2), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
          (conv2): Conv2d(
            512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
          (conv3): Conv2d(
            512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
          )
        )
        (1): BottleneckBlock(
          (conv1): Conv2d(
            2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
          (conv2): Conv2d(
            512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
          (conv3): Conv2d(
            512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
          )
        )
        (2): BottleneckBlock(
          (conv1): Conv2d(
            2048, 512, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
          (conv2): Conv2d(
            512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias
=False
```

```
            (norm): FrozenBatchNorm2d(num_features=512, eps=1e-05)
          )
          (conv3): Conv2d(
            512, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False
            (norm): FrozenBatchNorm2d(num_features=2048, eps=1e-05)
          )
        )
      )
    )
  )
  (proposal_generator): RPN(
    (rpn_head): StandardRPNHead(
      (conv): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
        (activation): ReLU()
      )
      (objectness_logits): Conv2d(256, 3, kernel_size=(1, 1), stride=(1, 1))
      (anchor_deltas): Conv2d(256, 12, kernel_size=(1, 1), stride=(1, 1))
    )
    (anchor_generator): DefaultAnchorGenerator(
      (cell_anchors): BufferList()
    )
  )
  (roi_heads): StandardROIHeads(
    (box_pooler): ROIPooler(
      (level_poolers): ModuleList(
        (0): ROIAlign(output_size=(7, 7), spatial_scale=0.25, sampling_ratio=
0, aligned=True)
        (1): ROIAlign(output_size=(7, 7), spatial_scale=0.125, sampling_ratio
=0, aligned=True)
        (2): ROIAlign(output_size=(7, 7), spatial_scale=0.0625, sampling_rati
o=0, aligned=True)
        (3): ROIAlign(output_size=(7, 7), spatial_scale=0.03125, sampling_rat
io=0, aligned=True)
      )
    )
    (box_head): FastRCNNConvFCHead(
      (flatten): Flatten(start_dim=1, end_dim=-1)
      (fc1): Linear(in_features=12544, out_features=1024, bias=True)
      (fc_relu1): ReLU()
      (fc2): Linear(in_features=1024, out_features=1024, bias=True)
      (fc_relu2): ReLU()
    )
    (box_predictor): FastRCNNOutputLayers(
      (cls_score): Linear(in_features=1024, out_features=6, bias=True)
      (bbox_pred): Linear(in_features=1024, out_features=20, bias=True)
    )
    (mask_pooler): ROIPooler(
      (level_poolers): ModuleList(
        (0): ROIAlign(output_size=(14, 14), spatial_scale=0.25, sampling_rati
o=0, aligned=True)
        (1): ROIAlign(output_size=(14, 14), spatial_scale=0.125, sampling_rat
io=0, aligned=True)
```

```
        (2): ROIAlign(output_size=(14, 14), spatial_scale=0.0625, sampling_ra
tio=0, aligned=True)
        (3): ROIAlign(output_size=(14, 14), spatial_scale=0.03125, sampling_r
atio=0, aligned=True)
      )
    )
    (mask_head): MaskRCNNConvUpsampleHead(
      (mask_fcn1): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
        (activation): ReLU()
      )
      (mask_fcn2): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
        (activation): ReLU()
      )
      (mask_fcn3): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
        (activation): ReLU()
      )
      (mask_fcn4): Conv2d(
        256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)
        (activation): ReLU()
      )
      (deconv): ConvTranspose2d(256, 256, kernel_size=(2, 2), stride=(2, 2))
      (deconv_relu): ReLU()
      (predictor): Conv2d(256, 5, kernel_size=(1, 1), stride=(1, 1))
    )
  )
)
[05/19 16:17:44 d2.data.datasets.coco]: Loaded 59 images in COCO format from
damage_dataset/train/COCO_mul_train_annos.json
[05/19 16:17:44 d2.data.build]: Removed 0 images with no usable annotations.
59 images left.
[05/19 16:17:44 d2.data.build]: Distribution of instances among all 5 categor
ies:
|  category  | #instances   |   category   | #instances   |  category  | #ins
tances   |
|:----------:|:-------------|:------------:|:-------------|:----------:|:----
---------|
|  headlamp  | 64           | rear_bumper  | 24           |    door    | 32
|
|    hood    | 26           | front_bumper | 31           |            |
|
|   total    | 177          |              |              |            |
|
[05/19 16:17:44 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used i
n training: [RandomCrop(crop_type='relative_range', crop_size=[0.9, 0.9]), Re
sizeShortestEdge(short_edge_length=(640, 672, 704, 736, 768, 800), max_size=1
333, sample_style='choice'), RandomFlip()]
[05/19 16:17:44 d2.data.build]: Using training sampler TrainingSampler
[05/19 16:17:44 d2.data.common]: Serializing 59 elements to byte tensors and
concatenating them all ...
[05/19 16:17:44 d2.data.common]: Serialized dataset takes 0.03 MiB
```

```
Skip loading parameter 'roi_heads.box_predictor.cls_score.weight' to the mode
l due to incompatible shapes: (81, 1024) in the checkpoint but (6, 1024) in t
he model! You might want to double check if this is expected.
Skip loading parameter 'roi_heads.box_predictor.cls_score.bias' to the model
due to incompatible shapes: (81,) in the checkpoint but (6,) in the model! Yo
u might want to double check if this is expected.
Skip loading parameter 'roi_heads.box_predictor.bbox_pred.weight' to the mode
l due to incompatible shapes: (320, 1024) in the checkpoint but (20, 1024) in
the model! You might want to double check if this is expected.
Skip loading parameter 'roi_heads.box_predictor.bbox_pred.bias' to the model
due to incompatible shapes: (320,) in the checkpoint but (20,) in the model!
You might want to double check if this is expected.
Skip loading parameter 'roi_heads.mask_head.predictor.weight' to the model du
e to incompatible shapes: (80, 256, 1, 1) in the checkpoint but (5, 256, 1, 1
) in the model! You might want to double check if this is expected.
Skip loading parameter 'roi_heads.mask_head.predictor.bias' to the model due
to incompatible shapes: (80,) in the checkpoint but (5,) in the model! You mi
ght want to double check if this is expected.
Some model parameters or buffers are not found in the checkpoint:
roi_heads.box_predictor.bbox_pred.{bias, weight}
roi_heads.box_predictor.cls_score.{bias, weight}
roi_heads.mask_head.predictor.{bias, weight}
[05/19 16:17:44 d2.engine.train_loop]: Starting training from iteration 0
/usr/local/lib/python3.7/dist-packages/detectron2/structures/image_list.py:88
: UserWarning: __floordiv__ is deprecated, and its behavior will change in a
future version of pytorch. It currently rounds toward 0 (like the 'trunc' fun
ction NOT 'floor'). This results in incorrect rounding for negative values. T
o keep the current behavior, use torch.div(a, b, rounding_mode='trunc'), or f
or actual floor division, use torch.div(a, b, rounding_mode='floor').
  max_size = (max_size + (stride - 1)) // stride * stride
[05/19 16:17:56 d2.utils.events]:  eta: 0:22:58  iter: 19  total_loss: 2.912
loss_cls: 1.719  loss_box_reg: 0.388  loss_mask: 0.6955  loss_rpn_cls: 0.0530
1  loss_rpn_loc: 0.01793  time: 0.5663  data_time: 0.0148  lr: 2.1638e-06  ma
x_mem: 1973M
[05/19 16:18:07 d2.utils.events]:  eta: 0:22:35  iter: 39  total_loss: 2.94
loss_cls: 1.652  loss_box_reg: 0.4822  loss_mask: 0.6949  loss_rpn_cls: 0.090
88  loss_rpn_loc: 0.02903  time: 0.5644  data_time: 0.0040  lr: 3.3888e-06  m
ax_mem: 1973M
[05/19 16:18:19 d2.utils.events]:  eta: 0:22:36  iter: 59  total_loss: 2.977
loss_cls: 1.574  loss_box_reg: 0.6284  loss_mask: 0.695  loss_rpn_cls: 0.0991
5  loss_rpn_loc: 0.02686  time: 0.5663  data_time: 0.0043  lr: 4.6137e-06  ma
x_mem: 1973M
[05/19 16:18:30 d2.utils.events]:  eta: 0:22:24  iter: 79  total_loss: 2.79
loss_cls: 1.462  loss_box_reg: 0.4704  loss_mask: 0.6932  loss_rpn_cls: 0.072
08  loss_rpn_loc: 0.03234  time: 0.5629  data_time: 0.0040  lr: 5.8388e-06  m
ax_mem: 1973M
[05/19 16:18:41 d2.utils.events]:  eta: 0:22:08  iter: 99  total_loss: 2.596
loss_cls: 1.331  loss_box_reg: 0.4759  loss_mask: 0.6921  loss_rpn_cls: 0.057
05  loss_rpn_loc: 0.02713  time: 0.5631  data_time: 0.0040  lr: 7.0638e-06  m
ax_mem: 1973M
[05/19 16:18:53 d2.utils.events]:  eta: 0:22:21  iter: 119  total_loss: 2.446
loss_cls: 1.199  loss_box_reg: 0.4031  loss_mask: 0.6905  loss_rpn_cls: 0.023
```

```
loss_rpn_loc: 0.0169  time: 0.5675  data_time: 0.0043  lr: 8.2888e-06  max_me
m: 1973M
[05/19 16:19:04 d2.utils.events]:  eta: 0:21:56  iter: 139  total_loss: 2.474
loss_cls: 1.094  loss_box_reg: 0.4957  loss_mask: 0.6878  loss_rpn_cls: 0.055
94  loss_rpn_loc: 0.02608  time: 0.5637  data_time: 0.0043  lr: 9.5138e-06  m
ax_mem: 1973M
[05/19 16:19:15 d2.utils.events]:  eta: 0:21:56  iter: 159  total_loss: 2.22
loss_cls: 0.9202  loss_box_reg: 0.4855  loss_mask: 0.6864  loss_rpn_cls: 0.05
018  loss_rpn_loc: 0.02128  time: 0.5659  data_time: 0.0043  lr: 1.0739e-05
max_mem: 1973M
[05/19 16:19:27 d2.utils.events]:  eta: 0:21:49  iter: 179  total_loss: 2.061
loss_cls: 0.7976  loss_box_reg: 0.4309  loss_mask: 0.6854  loss_rpn_cls: 0.06
103  loss_rpn_loc: 0.02589  time: 0.5674  data_time: 0.0040  lr: 1.1964e-05
max_mem: 1973M
[05/19 16:19:38 d2.utils.events]:  eta: 0:21:36  iter: 199  total_loss: 2.233
loss_cls: 0.8004  loss_box_reg: 0.6196  loss_mask: 0.6818  loss_rpn_cls: 0.05
38  loss_rpn_loc: 0.02512  time: 0.5668  data_time: 0.0040  lr: 1.3189e-05  m
ax_mem: 1973M
/usr/local/lib/python3.7/dist-packages/fvcore/transforms/transform.py:724: Sh
apelyDeprecationWarning: Iteration over multi-part geometries is deprecated a
nd will be removed in Shapely 2.0. Use the `geoms` property to access the con
stituent parts of a multi-part geometry.
  for poly in cropped:
[05/19 16:19:49 d2.utils.events]:  eta: 0:21:27  iter: 219  total_loss: 1.904
loss_cls: 0.6886  loss_box_reg: 0.4745  loss_mask: 0.6796  loss_rpn_cls: 0.06
429  loss_rpn_loc: 0.02604  time: 0.5651  data_time: 0.0042  lr: 1.4414e-05
max_mem: 1973M
[05/19 16:20:01 d2.utils.events]:  eta: 0:21:16  iter: 239  total_loss: 1.795
loss_cls: 0.5902  loss_box_reg: 0.4523  loss_mask: 0.6688  loss_rpn_cls: 0.01
603  loss_rpn_loc: 0.01749  time: 0.5666  data_time: 0.0040  lr: 1.5639e-05
max_mem: 1973M
[05/19 16:20:12 d2.utils.events]:  eta: 0:21:02  iter: 259  total_loss: 1.782
loss_cls: 0.5963  loss_box_reg: 0.4172  loss_mask: 0.669  loss_rpn_cls: 0.094
4  loss_rpn_loc: 0.03156  time: 0.5658  data_time: 0.0040  lr: 1.6864e-05  ma
x_mem: 1973M
[05/19 16:20:23 d2.utils.events]:  eta: 0:20:51  iter: 279  total_loss: 1.812
loss_cls: 0.6157  loss_box_reg: 0.4697  loss_mask: 0.6676  loss_rpn_cls: 0.02
427  loss_rpn_loc: 0.02499  time: 0.5669  data_time: 0.0043  lr: 1.8089e-05
max_mem: 1973M
[05/19 16:20:35 d2.utils.events]:  eta: 0:20:36  iter: 299  total_loss: 1.829
loss_cls: 0.5827  loss_box_reg: 0.4743  loss_mask: 0.6683  loss_rpn_cls: 0.02
278  loss_rpn_loc: 0.01949  time: 0.5663  data_time: 0.0039  lr: 1.9314e-05
max_mem: 1973M
[05/19 16:20:47 d2.utils.events]:  eta: 0:20:26  iter: 319  total_loss: 1.995
loss_cls: 0.6653  loss_box_reg: 0.6234  loss_mask: 0.6568  loss_rpn_cls: 0.02
385  loss_rpn_loc: 0.01612  time: 0.5680  data_time: 0.0039  lr: 2.0539e-05
max_mem: 1973M
[05/19 16:20:58 d2.utils.events]:  eta: 0:20:14  iter: 339  total_loss: 1.596
loss_cls: 0.473  loss_box_reg: 0.4132  loss_mask: 0.6447  loss_rpn_cls: 0.028
39  loss_rpn_loc: 0.02253  time: 0.5677  data_time: 0.0041  lr: 2.1764e-05  m
ax_mem: 1973M
[05/19 16:21:09 d2.utils.events]:  eta: 0:20:03  iter: 359  total_loss: 1.708
loss_cls: 0.5335  loss_box_reg: 0.5021  loss_mask: 0.6519  loss_rpn_cls: 0.02
```

```
81  loss_rpn_loc: 0.02476  time: 0.5667  data_time: 0.0039  lr: 2.2989e-05  m
ax_mem: 1973M
[05/19 16:21:21 d2.utils.events]:  eta: 0:19:54  iter: 379  total_loss: 1.868
loss_cls: 0.5813  loss_box_reg: 0.5318  loss_mask: 0.6363  loss_rpn_cls: 0.02
757  loss_rpn_loc: 0.02497  time: 0.5677  data_time: 0.0041  lr: 2.4214e-05
max_mem: 1973M
[05/19 16:21:34 d2.data.datasets.coco]: Loaded 11 images in COCO format from
damage_dataset/val/COCO_mul_val_annos.json
[05/19 16:21:34 d2.data.build]: Distribution of instances among all 5 categor
ies:
|  category  | #instances   |   category   | #instances   |  category  | #ins
tances   |
|:----------:|:-------------|:------------:|:-------------|:----------:|:----
---------|
|  headlamp  | 12           | rear_bumper  | 5            |    door    | 10
|
|    hood    | 5            | front_bumper | 6            |            |
|
|   total    | 38           |              |              |            |
|
[05/19 16:21:34 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used i
n inference: [ResizeShortestEdge(short_edge_length=(800, 800), max_size=1333,
sample_style='choice')]
[05/19 16:21:34 d2.data.common]: Serializing 11 elements to byte tensors and
concatenating them all ...
[05/19 16:21:34 d2.data.common]: Serialized dataset takes 0.01 MiB
[05/19 16:21:34 d2.evaluation.evaluator]: Start inference on 11 batches
[05/19 16:21:44 d2.evaluation.evaluator]: Inference done 11/11. Dataloading:
0.0021 s/iter. Inference: 0.3789 s/iter. Eval: 0.5137 s/iter. Total: 0.8947 s
/iter. ETA=0:00:00
[05/19 16:21:44 d2.evaluation.evaluator]: Total inference time: 0:00:05.42828
1 (0.904714 s / iter per device, on 1 devices)
[05/19 16:21:44 d2.evaluation.evaluator]: Total inference pure compute time:
0:00:02 (0.378862 s / iter per device, on 1 devices)
[05/19 16:21:44 d2.evaluation.coco_evaluation]: Preparing results for COCO fo
rmat ...
[05/19 16:21:44 d2.evaluation.coco_evaluation]: Saving results to coco_eval/c
oco_instances_results.json
[05/19 16:21:44 d2.evaluation.coco_evaluation]: Evaluating predictions with u
nofficial COCO API...
Loading and preparing results...
DONE (t=0.00s)
creating index...
index created!
[05/19 16:21:44 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[05/19 16:21:44 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.02 seconds.
[05/19 16:21:44 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:21:44 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.02 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.0
07
```

```
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.0
16
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.0
04
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.0
14
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.0
13
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.0
21
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.1
38
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.1
70
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.1
30
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.2
02
[05/19 16:21:44 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|  AP   |  AP50  |  AP75  |  APs  |  APm  |  APl  |
|:-----:|:------:|:------:|:-----:|:-----:|:-----:|
| 0.707 | 1.568  | 0.370  |  nan  | 1.413 | 1.275 |
[05/19 16:21:44 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:21:44 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP    | category     | AP    | category   | AP    |
|:-----------|:------|:-------------|:------|:-----------|:------|
| headlamp   | 1.272 | rear_bumper  | 1.052 | door       | 0.178 |
| hood       | 0.000 | front_bumper | 1.031 |            |       |
Loading and preparing results...
DONE (t=0.05s)
creating index...
index created!
[05/19 16:21:44 d2.evaluation.fast_eval_api]: Evaluate annotation type *segm*
[05/19 16:21:44 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.04 seconds.
[05/19 16:21:44 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:21:44 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.02 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.0
07
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.0
13
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.0
07
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
```

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.0
06
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.0
16
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.0
47
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.1
32
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.1
60
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.1
40
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.1
87
[05/19 16:21:44 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|  AP   |  AP50  |  AP75  |  APs  |  APm  |  APl  |
|:-----:|:------:|:------:|:-----:|:-----:|:-----:|
| 0.726 | 1.253  | 0.731  |  nan  | 0.636 | 1.631 |
[05/19 16:21:44 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:21:44 d2.evaluation.coco_evaluation]: Per-category segm AP:
| category   | AP    | category     | AP    | category   | AP    |
|:-----------|:------|:-------------|:------|:-----------|:------|
| headlamp   | 1.216 | rear_bumper  | 0.704 | door       | 0.068 |
| hood       | 0.000 | front_bumper | 1.644 |            |       |
[05/19 16:21:44 d2.engine.defaults]: Evaluation results for car_part_val in c
sv format:
[05/19 16:21:44 d2.evaluation.testing]: copypaste: Task: bbox
[05/19 16:21:44 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:21:44 d2.evaluation.testing]: copypaste: 0.7067,1.5678,0.3703,nan,1
.4125,1.2746
[05/19 16:21:44 d2.evaluation.testing]: copypaste: Task: segm
[05/19 16:21:44 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:21:44 d2.evaluation.testing]: copypaste: 0.7263,1.2533,0.7308,nan,0
.6364,1.6308
[05/19 16:21:44 d2.utils.events]:  eta: 0:19:45  iter: 399  total_loss: 1.679
loss_cls: 0.4959  loss_box_reg: 0.5131  loss_mask: 0.6298  loss_rpn_cls: 0.01
686  loss_rpn_loc: 0.01959  time: 0.5687  data_time: 0.0040  lr: 2.5439e-05
max_mem: 1973M
[05/19 16:21:56 d2.utils.events]:  eta: 0:19:34  iter: 419  total_loss: 2.006
loss_cls: 0.6237  loss_box_reg: 0.6319  loss_mask: 0.6455  loss_rpn_cls: 0.02
417  loss_rpn_loc: 0.01975  time: 0.5693  data_time: 0.0045  lr: 2.6664e-05
max_mem: 1973M
[05/19 16:22:07 d2.utils.events]:  eta: 0:19:18  iter: 439  total_loss: 1.674
loss_cls: 0.5099  loss_box_reg: 0.5396  loss_mask: 0.628  loss_rpn_cls: 0.015
64  loss_rpn_loc: 0.01619  time: 0.5683  data_time: 0.0041  lr: 2.7889e-05  m
ax_mem: 1973M
[05/19 16:22:19 d2.utils.events]:  eta: 0:19:11  iter: 459  total_loss: 1.879
loss_cls: 0.5805  loss_box_reg: 0.6008  loss_mask: 0.6191  loss_rpn_cls: 0.01
192  loss_rpn_loc: 0.01813  time: 0.5689  data_time: 0.0037  lr: 2.9114e-05
max_mem: 1973M
```

[05/19 16:22:30 d2.utils.events]: eta: 0:19:00 iter: 479 total_loss: 1.779 loss_cls: 0.5371 loss_box_reg: 0.4604 loss_mask: 0.6042 loss_rpn_cls: 0.0269 loss_rpn_loc: 0.02171 time: 0.5690 data_time: 0.0040 lr: 3.0339e-05 max_mem: 1973M
[05/19 16:22:41 d2.utils.events]: eta: 0:18:49 iter: 499 total_loss: 1.922 loss_cls: 0.6155 loss_box_reg: 0.7148 loss_mask: 0.6178 loss_rpn_cls: 0.01176 loss_rpn_loc: 0.01917 time: 0.5690 data_time: 0.0040 lr: 3.1564e-05 max_mem: 1973M
[05/19 16:22:53 d2.utils.events]: eta: 0:18:33 iter: 519 total_loss: 1.703 loss_cls: 0.4749 loss_box_reg: 0.4438 loss_mask: 0.6062 loss_rpn_cls: 0.0245 loss_rpn_loc: 0.01799 time: 0.5685 data_time: 0.0037 lr: 3.2789e-05 max_mem: 1973M
[05/19 16:23:04 d2.utils.events]: eta: 0:18:21 iter: 539 total_loss: 1.77 loss_cls: 0.5429 loss_box_reg: 0.5844 loss_mask: 0.6112 loss_rpn_cls: 0.02495 loss_rpn_loc: 0.02221 time: 0.5682 data_time: 0.0040 lr: 3.4014e-05 max_mem: 1973M
[05/19 16:23:15 d2.utils.events]: eta: 0:18:10 iter: 559 total_loss: 1.538 loss_cls: 0.4654 loss_box_reg: 0.4335 loss_mask: 0.5902 loss_rpn_cls: 0.01866 loss_rpn_loc: 0.0174 time: 0.5681 data_time: 0.0037 lr: 3.5239e-05 max_mem: 1973M
[05/19 16:23:27 d2.utils.events]: eta: 0:17:59 iter: 579 total_loss: 1.893 loss_cls: 0.6398 loss_box_reg: 0.6764 loss_mask: 0.576 loss_rpn_cls: 0.02353 loss_rpn_loc: 0.01981 time: 0.5681 data_time: 0.0041 lr: 3.6464e-05 max_mem: 1973M
[05/19 16:23:38 d2.utils.events]: eta: 0:17:48 iter: 599 total_loss: 1.738 loss_cls: 0.5284 loss_box_reg: 0.6531 loss_mask: 0.5683 loss_rpn_cls: 0.00952 loss_rpn_loc: 0.01653 time: 0.5687 data_time: 0.0040 lr: 3.7689e-05 max_mem: 1973M
[05/19 16:23:50 d2.utils.events]: eta: 0:17:37 iter: 619 total_loss: 1.78 loss_cls: 0.5284 loss_box_reg: 0.6026 loss_mask: 0.5701 loss_rpn_cls: 0.03085 loss_rpn_loc: 0.02204 time: 0.5693 data_time: 0.0039 lr: 3.8914e-05 max_mem: 1973M
[05/19 16:24:01 d2.utils.events]: eta: 0:17:26 iter: 639 total_loss: 1.615 loss_cls: 0.4581 loss_box_reg: 0.5543 loss_mask: 0.5523 loss_rpn_cls: 0.01749 loss_rpn_loc: 0.01201 time: 0.5694 data_time: 0.0040 lr: 4.0139e-05 max_mem: 1973M
[05/19 16:24:13 d2.utils.events]: eta: 0:17:17 iter: 659 total_loss: 1.901 loss_cls: 0.5834 loss_box_reg: 0.6789 loss_mask: 0.5408 loss_rpn_cls: 0.01509 loss_rpn_loc: 0.02078 time: 0.5700 data_time: 0.0040 lr: 4.1364e-05 max_mem: 1973M
[05/19 16:24:25 d2.utils.events]: eta: 0:17:03 iter: 679 total_loss: 1.662 loss_cls: 0.5019 loss_box_reg: 0.56 loss_mask: 0.5771 loss_rpn_cls: 0.02508 loss_rpn_loc: 0.01921 time: 0.5699 data_time: 0.0039 lr: 4.2589e-05 max_mem: 1973M
[05/19 16:24:36 d2.utils.events]: eta: 0:16:51 iter: 699 total_loss: 1.684 loss_cls: 0.5184 loss_box_reg: 0.5737 loss_mask: 0.518 loss_rpn_cls: 0.01377 loss_rpn_loc: 0.01477 time: 0.5695 data_time: 0.0044 lr: 4.3814e-05 max_mem: 1973M
[05/19 16:24:47 d2.utils.events]: eta: 0:16:40 iter: 719 total_loss: 1.817 loss_cls: 0.5514 loss_box_reg: 0.7039 loss_mask: 0.5124 loss_rpn_cls: 0.01088 loss_rpn_loc: 0.01296 time: 0.5696 data_time: 0.0042 lr: 4.5039e-05 max_mem: 1973M

```
[05/19 16:24:59 d2.utils.events]:  eta: 0:16:30  iter: 739  total_loss: 1.578
loss_cls: 0.4883  loss_box_reg: 0.5709  loss_mask: 0.4762  loss_rpn_cls: 0.01
497  loss_rpn_loc: 0.01215  time: 0.5702  data_time: 0.0039  lr: 4.6264e-05
max_mem: 1973M
[05/19 16:25:11 d2.utils.events]:  eta: 0:16:21  iter: 759  total_loss: 1.788
loss_cls: 0.5291  loss_box_reg: 0.6949  loss_mask: 0.5273  loss_rpn_cls: 0.02
248  loss_rpn_loc: 0.02161  time: 0.5706  data_time: 0.0041  lr: 4.7489e-05
max_mem: 1973M
[05/19 16:25:22 d2.utils.events]:  eta: 0:16:08  iter: 779  total_loss: 1.75
loss_cls: 0.5302  loss_box_reg: 0.6109  loss_mask: 0.5269  loss_rpn_cls: 0.01
418  loss_rpn_loc: 0.01828  time: 0.5702  data_time: 0.0040  lr: 4.8714e-05
max_mem: 1973M
[05/19 16:25:35 d2.data.datasets.coco]: Loaded 11 images in COCO format from
damage_dataset/val/COCO_mul_val_annos.json
[05/19 16:25:35 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used i
n inference: [ResizeShortestEdge(short_edge_length=(800, 800), max_size=1333,
sample_style='choice')]
[05/19 16:25:35 d2.data.common]: Serializing 11 elements to byte tensors and
concatenating them all ...
[05/19 16:25:35 d2.data.common]: Serialized dataset takes 0.01 MiB
[05/19 16:25:35 d2.evaluation.evaluator]: Start inference on 11 batches
[05/19 16:25:45 d2.evaluation.evaluator]: Inference done 11/11. Dataloading:
0.0017 s/iter. Inference: 0.3804 s/iter. Eval: 0.4693 s/iter. Total: 0.8513 s
/iter. ETA=0:00:00
[05/19 16:25:45 d2.evaluation.evaluator]: Total inference time: 0:00:05.22639
6 (0.871066 s / iter per device, on 1 devices)
[05/19 16:25:45 d2.evaluation.evaluator]: Total inference pure compute time:
0:00:02 (0.380384 s / iter per device, on 1 devices)
[05/19 16:25:45 d2.evaluation.coco_evaluation]: Preparing results for COCO fo
rmat ...
[05/19 16:25:45 d2.evaluation.coco_evaluation]: Saving results to coco_eval/c
oco_instances_results.json
[05/19 16:25:45 d2.evaluation.coco_evaluation]: Evaluating predictions with u
nofficial COCO API...
Loading and preparing results...
DONE (t=0.00s)
creating index...
index created!
[05/19 16:25:45 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[05/19 16:25:45 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.04 seconds.
[05/19 16:25:45 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:25:45 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.05 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.0
59
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.1
19
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.0
46
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
```

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.2
16
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.0
80
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.0
74
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.2
99
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.3
57
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.3
67
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.3
77
[05/19 16:25:45 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|  AP   |  AP50  |  AP75  |  APs  |  APm   |  APl  |
|:-----:|:------:|:------:|:-----:|:------:|:-----:|
| 5.898 | 11.873 | 4.612  |  nan  | 21.611 | 8.010 |
[05/19 16:25:45 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:25:45 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category     | AP    | category   | AP    |
|:-----------|:-------|:-------------|:------|:-----------|:------|
| headlamp   | 13.549 | rear_bumper  | 8.484 | door       | 2.181 |
| hood       | 0.861  | front_bumper | 4.416 |            |       |
Loading and preparing results...
DONE (t=0.07s)
creating index...
index created!
[05/19 16:25:45 d2.evaluation.fast_eval_api]: Evaluate annotation type *segm*
[05/19 16:25:45 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.06 seconds.
[05/19 16:25:45 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:25:45 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.03 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.0
62
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.1
02
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.0
70
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.0
94
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.0
86
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.0
79
```

```
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.2
76
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.3
42
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.3
55
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.3
62
[05/19 16:25:45 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|  AP   |  AP50  |  AP75  |  APs  |  APm  |  APl  |
|:-----:|:------:|:------:|:-----:|:-----:|:-----:|
| 6.158 | 10.192 | 7.000  |  nan  | 9.412 | 8.574 |
[05/19 16:25:45 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:25:45 d2.evaluation.coco_evaluation]: Per-category segm AP:
| category   | AP     | category     | AP    | category   | AP    |
|:-----------|:-------|:-------------|:------|:-----------|:------|
| headlamp   | 14.620 | rear_bumper  | 9.009 | door       | 1.361 |
| hood       | 0.598  | front_bumper | 5.201 |            |       |
[05/19 16:25:45 d2.engine.defaults]: Evaluation results for car_part_val in c
sv format:
[05/19 16:25:45 d2.evaluation.testing]: copypaste: Task: bbox
[05/19 16:25:45 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:25:45 d2.evaluation.testing]: copypaste: 5.8981,11.8728,4.6115,nan,
21.6106,8.0098
[05/19 16:25:45 d2.evaluation.testing]: copypaste: Task: segm
[05/19 16:25:45 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:25:45 d2.evaluation.testing]: copypaste: 6.1579,10.1923,6.9999,nan,
9.4122,8.5737
[05/19 16:25:45 d2.utils.events]:  eta: 0:15:58  iter: 799  total_loss: 1.654
loss_cls: 0.5052  loss_box_reg: 0.5849  loss_mask: 0.471  loss_rpn_cls: 0.013
37  loss_rpn_loc: 0.01942  time: 0.5707  data_time: 0.0039  lr: 4.9939e-05  m
ax_mem: 1973M
[05/19 16:25:57 d2.utils.events]:  eta: 0:15:48  iter: 819  total_loss: 1.576
loss_cls: 0.4259  loss_box_reg: 0.5673  loss_mask: 0.4915  loss_rpn_cls: 0.01
456  loss_rpn_loc: 0.01705  time: 0.5710  data_time: 0.0044  lr: 5e-05  max_m
em: 1973M
[05/19 16:26:09 d2.utils.events]:  eta: 0:15:36  iter: 839  total_loss: 1.687
loss_cls: 0.4692  loss_box_reg: 0.5968  loss_mask: 0.4919  loss_rpn_cls: 0.01
449  loss_rpn_loc: 0.01868  time: 0.5713  data_time: 0.0041  lr: 5e-05  max_m
em: 1973M
[05/19 16:26:21 d2.utils.events]:  eta: 0:15:27  iter: 859  total_loss: 1.636
loss_cls: 0.4951  loss_box_reg: 0.5674  loss_mask: 0.476  loss_rpn_cls: 0.011
76  loss_rpn_loc: 0.01603  time: 0.5719  data_time: 0.0043  lr: 5e-05  max_me
m: 1973M
[05/19 16:26:32 d2.utils.events]:  eta: 0:15:16  iter: 879  total_loss: 1.632
loss_cls: 0.4648  loss_box_reg: 0.6361  loss_mask: 0.4416  loss_rpn_cls: 0.01
078  loss_rpn_loc: 0.01448  time: 0.5721  data_time: 0.0037  lr: 5e-05  max_m
em: 1973M
[05/19 16:26:44 d2.utils.events]:  eta: 0:15:04  iter: 899  total_loss: 1.336
loss_cls: 0.3644  loss_box_reg: 0.5517  loss_mask: 0.4835  loss_rpn_cls: 0.00
```

8206 loss_rpn_loc: 0.01477 time: 0.5720 data_time: 0.0038 lr: 5e-05 max_
mem: 1973M
[05/19 16:26:55 d2.utils.events]: eta: 0:14:53 iter: 919 total_loss: 1.624
loss_cls: 0.4793 loss_box_reg: 0.6665 loss_mask: 0.4269 loss_rpn_cls: 0.01
634 loss_rpn_loc: 0.0154 time: 0.5720 data_time: 0.0040 lr: 5e-05 max_me
m: 1973M
[05/19 16:27:07 d2.utils.events]: eta: 0:14:42 iter: 939 total_loss: 1.511
loss_cls: 0.453 loss_box_reg: 0.6032 loss_mask: 0.4347 loss_rpn_cls: 0.008
292 loss_rpn_loc: 0.01305 time: 0.5725 data_time: 0.0041 lr: 5e-05 max_m
em: 1973M
[05/19 16:27:19 d2.utils.events]: eta: 0:14:31 iter: 959 total_loss: 1.508
loss_cls: 0.4651 loss_box_reg: 0.6068 loss_mask: 0.4207 loss_rpn_cls: 0.00
7813 loss_rpn_loc: 0.013 time: 0.5729 data_time: 0.0040 lr: 5e-05 max_me
m: 1973M
[05/19 16:27:31 d2.utils.events]: eta: 0:14:21 iter: 979 total_loss: 1.63
loss_cls: 0.4982 loss_box_reg: 0.7252 loss_mask: 0.3903 loss_rpn_cls: 0.00
5221 loss_rpn_loc: 0.01375 time: 0.5732 data_time: 0.0040 lr: 5e-05 max_
mem: 1973M
[05/19 16:27:42 d2.utils.events]: eta: 0:14:09 iter: 999 total_loss: 1.447
loss_cls: 0.4453 loss_box_reg: 0.5296 loss_mask: 0.4043 loss_rpn_cls: 0.01
307 loss_rpn_loc: 0.01279 time: 0.5734 data_time: 0.0038 lr: 5e-05 max_m
em: 1973M
[05/19 16:27:54 d2.utils.events]: eta: 0:13:58 iter: 1019 total_loss: 1.53
4 loss_cls: 0.4484 loss_box_reg: 0.6541 loss_mask: 0.4062 loss_rpn_cls: 0
.01049 loss_rpn_loc: 0.01563 time: 0.5736 data_time: 0.0038 lr: 5e-05 ma
x_mem: 1973M
[05/19 16:28:06 d2.utils.events]: eta: 0:13:48 iter: 1039 total_loss: 1.56
3 loss_cls: 0.4441 loss_box_reg: 0.6367 loss_mask: 0.377 loss_rpn_cls: 0.
01439 loss_rpn_loc: 0.02127 time: 0.5742 data_time: 0.0037 lr: 5e-05 max
_mem: 1973M
[05/19 16:28:18 d2.utils.events]: eta: 0:13:37 iter: 1059 total_loss: 1.56
8 loss_cls: 0.4703 loss_box_reg: 0.6634 loss_mask: 0.3592 loss_rpn_cls: 0
.007671 loss_rpn_loc: 0.01275 time: 0.5742 data_time: 0.0048 lr: 5e-05 m
ax_mem: 1973M
[05/19 16:28:29 d2.utils.events]: eta: 0:13:26 iter: 1079 total_loss: 1.39
4 loss_cls: 0.3938 loss_box_reg: 0.6155 loss_mask: 0.3718 loss_rpn_cls: 0
.01028 loss_rpn_loc: 0.01061 time: 0.5744 data_time: 0.0039 lr: 5e-05 ma
x_mem: 1973M
[05/19 16:28:41 d2.utils.events]: eta: 0:13:15 iter: 1099 total_loss: 1.45
5 loss_cls: 0.3879 loss_box_reg: 0.6113 loss_mask: 0.3384 loss_rpn_cls: 0
.00731 loss_rpn_loc: 0.01341 time: 0.5744 data_time: 0.0037 lr: 5e-05 ma
x_mem: 1973M
[05/19 16:28:52 d2.utils.events]: eta: 0:13:03 iter: 1119 total_loss: 1.45
4 loss_cls: 0.4259 loss_box_reg: 0.5707 loss_mask: 0.3253 loss_rpn_cls: 0
.01511 loss_rpn_loc: 0.02098 time: 0.5744 data_time: 0.0040 lr: 5e-05 ma
x_mem: 1973M
[05/19 16:29:04 d2.utils.events]: eta: 0:12:53 iter: 1139 total_loss: 1.39
loss_cls: 0.3677 loss_box_reg: 0.6058 loss_mask: 0.315 loss_rpn_cls: 0.009
18 loss_rpn_loc: 0.01608 time: 0.5747 data_time: 0.0038 lr: 5e-05 max_me
m: 1973M
[05/19 16:29:16 d2.utils.events]: eta: 0:12:41 iter: 1159 total_loss: 1.34
4 loss_cls: 0.3841 loss_box_reg: 0.5768 loss_mask: 0.343 loss_rpn_cls: 0.

```
009491  loss_rpn_loc: 0.0191  time: 0.5750  data_time: 0.0039  lr: 5e-05  max
_mem: 1973M
[05/19 16:29:28 d2.utils.events]:  eta: 0:12:30  iter: 1179  total_loss: 1.37
9  loss_cls: 0.3609  loss_box_reg: 0.6155  loss_mask: 0.3188  loss_rpn_cls: 0
.01023  loss_rpn_loc: 0.01353  time: 0.5750  data_time: 0.0041  lr: 5e-05  ma
x_mem: 1973M
[05/19 16:29:40 d2.data.datasets.coco]: Loaded 11 images in COCO format from
damage_dataset/val/COCO_mul_val_annos.json
[05/19 16:29:40 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used i
n inference: [ResizeShortestEdge(short_edge_length=(800, 800), max_size=1333,
sample_style='choice')]
[05/19 16:29:40 d2.data.common]: Serializing 11 elements to byte tensors and
concatenating them all ...
[05/19 16:29:40 d2.data.common]: Serialized dataset takes 0.01 MiB
[05/19 16:29:40 d2.evaluation.evaluator]: Start inference on 11 batches
[05/19 16:29:48 d2.evaluation.evaluator]: Inference done 11/11. Dataloading:
0.0018 s/iter. Inference: 0.3441 s/iter. Eval: 0.3631 s/iter. Total: 0.7090 s
/iter. ETA=0:00:00
[05/19 16:29:49 d2.evaluation.evaluator]: Total inference time: 0:00:04.31249
3 (0.718749 s / iter per device, on 1 devices)
[05/19 16:29:49 d2.evaluation.evaluator]: Total inference pure compute time:
0:00:02 (0.344144 s / iter per device, on 1 devices)
[05/19 16:29:49 d2.evaluation.coco_evaluation]: Preparing results for COCO fo
rmat ...
[05/19 16:29:49 d2.evaluation.coco_evaluation]: Saving results to coco_eval/c
oco_instances_results.json
[05/19 16:29:49 d2.evaluation.coco_evaluation]: Evaluating predictions with u
nofficial COCO API...
Loading and preparing results...
DONE (t=0.00s)
creating index...
index created!
[05/19 16:29:49 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[05/19 16:29:49 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.02 seconds.
[05/19 16:29:49 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:29:49 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.01 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.1
10
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.2
52
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.0
77
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.3
10
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.1
27
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.1
48
```

```
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.4
17
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.4
37
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.3
32
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.4
98
[05/19 16:29:49 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 10.983 | 25.187 | 7.696  |  nan  | 30.970 | 12.670 |
[05/19 16:29:49 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:29:49 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category     | AP     | category   | AP     |
|:-----------|:-------|:-------------|:-------|:-----------|:-------|
| headlamp   | 18.129 | rear_bumper  | 8.686  | door       | 11.230 |
| hood       | 3.993  | front_bumper | 12.877 |            |        |
Loading and preparing results...
DONE (t=0.02s)
creating index...
index created!
[05/19 16:29:49 d2.evaluation.fast_eval_api]: Evaluate annotation type *segm*
[05/19 16:29:49 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.03 seconds.
[05/19 16:29:49 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:29:49 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.02 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.1
67
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.2
62
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.2
04
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.3
99
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.1
84
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.2
71
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.4
74
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.4
87
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
```

```
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.4
46
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.5
17
[05/19 16:29:49 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 16.742 | 26.240 | 20.368 |  nan  | 39.924 | 18.378 |
[05/19 16:29:49 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:29:49 d2.evaluation.coco_evaluation]: Per-category segm AP:
| category   | AP     | category     | AP     | category   | AP     |
|:-----------|:-------|:-------------|:-------|:-----------|:-------|
| headlamp   | 21.344 | rear_bumper  | 11.924 | door       | 18.698 |
| hood       | 9.675  | front_bumper | 22.069 |            |        |
[05/19 16:29:49 d2.engine.defaults]: Evaluation results for car_part_val in c
sv format:
[05/19 16:29:49 d2.evaluation.testing]: copypaste: Task: bbox
[05/19 16:29:49 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:29:49 d2.evaluation.testing]: copypaste: 10.9831,25.1871,7.6958,nan
,30.9703,12.6704
[05/19 16:29:49 d2.evaluation.testing]: copypaste: Task: segm
[05/19 16:29:49 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:29:49 d2.evaluation.testing]: copypaste: 16.7420,26.2404,20.3683,na
n,39.9236,18.3775
[05/19 16:29:49 d2.utils.events]:  eta: 0:12:18  iter: 1199  total_loss: 1.35
8  loss_cls: 0.3763  loss_box_reg: 0.5969  loss_mask: 0.2697  loss_rpn_cls: 0
.01445  loss_rpn_loc: 0.0133  time: 0.5750  data_time: 0.0038  lr: 5e-05  max
_mem: 1973M
[05/19 16:30:01 d2.utils.events]:  eta: 0:12:08  iter: 1219  total_loss: 1.39
2  loss_cls: 0.3834  loss_box_reg: 0.6443  loss_mask: 0.2988  loss_rpn_cls: 0
.01985  loss_rpn_loc: 0.0182  time: 0.5754  data_time: 0.0041  lr: 5e-05  max
_mem: 1973M
[05/19 16:30:12 d2.utils.events]:  eta: 0:11:57  iter: 1239  total_loss: 1.19
3  loss_cls: 0.3186  loss_box_reg: 0.5825  loss_mask: 0.2567  loss_rpn_cls: 0
.007344  loss_rpn_loc: 0.01505  time: 0.5752  data_time: 0.0037  lr: 5e-05  m
ax_mem: 1973M
[05/19 16:30:24 d2.utils.events]:  eta: 0:11:46  iter: 1259  total_loss: 1.35
2  loss_cls: 0.3608  loss_box_reg: 0.6159  loss_mask: 0.2717  loss_rpn_cls: 0
.01479  loss_rpn_loc: 0.0185  time: 0.5754  data_time: 0.0039  lr: 5e-05  max
_mem: 1973M
[05/19 16:30:35 d2.utils.events]:  eta: 0:11:34  iter: 1279  total_loss: 1.24
7  loss_cls: 0.3232  loss_box_reg: 0.5642  loss_mask: 0.2664  loss_rpn_cls: 0
.004215  loss_rpn_loc: 0.01309  time: 0.5754  data_time: 0.0043  lr: 5e-05  m
ax_mem: 1973M
[05/19 16:30:48 d2.utils.events]:  eta: 0:11:25  iter: 1299  total_loss: 1.16
3  loss_cls: 0.3041  loss_box_reg: 0.5601  loss_mask: 0.2539  loss_rpn_cls: 0
.007191  loss_rpn_loc: 0.01105  time: 0.5758  data_time: 0.0038  lr: 5e-05  m
ax_mem: 1973M
[05/19 16:30:59 d2.utils.events]:  eta: 0:11:15  iter: 1319  total_loss: 1.24
6  loss_cls: 0.3141  loss_box_reg: 0.552  loss_mask: 0.2717  loss_rpn_cls: 0.
01647  loss_rpn_loc: 0.02313  time: 0.5761  data_time: 0.0041  lr: 5e-05  max
_mem: 1973M
```

```
[05/19 16:31:11 d2.utils.events]:  eta: 0:11:04  iter: 1339  total_loss: 1.07
6  loss_cls: 0.2722  loss_box_reg: 0.5406  loss_mask: 0.2461  loss_rpn_cls: 0
.003541  loss_rpn_loc: 0.01514  time: 0.5763  data_time: 0.0040  lr: 5e-05  m
ax_mem: 1973M
[05/19 16:31:23 d2.utils.events]:  eta: 0:10:52  iter: 1359  total_loss: 1.14
7  loss_cls: 0.3112  loss_box_reg: 0.5273  loss_mask: 0.2399  loss_rpn_cls: 0
.002606  loss_rpn_loc: 0.01359  time: 0.5763  data_time: 0.0038  lr: 5e-05  m
ax_mem: 1973M
[05/19 16:31:35 d2.utils.events]:  eta: 0:10:41  iter: 1379  total_loss: 1.05
8  loss_cls: 0.2627  loss_box_reg: 0.5083  loss_mask: 0.2268  loss_rpn_cls: 0
.003226  loss_rpn_loc: 0.01657  time: 0.5765  data_time: 0.0037  lr: 5e-05  m
ax_mem: 1973M
[05/19 16:31:46 d2.utils.events]:  eta: 0:10:30  iter: 1399  total_loss: 1.15
6  loss_cls: 0.3071  loss_box_reg: 0.5028  loss_mask: 0.2443  loss_rpn_cls: 0
.006123  loss_rpn_loc: 0.0191  time: 0.5767  data_time: 0.0038  lr: 5e-05  ma
x_mem: 1973M
[05/19 16:31:58 d2.utils.events]:  eta: 0:10:18  iter: 1419  total_loss: 1.02
7  loss_cls: 0.3091  loss_box_reg: 0.4665  loss_mask: 0.2057  loss_rpn_cls: 0
.005477  loss_rpn_loc: 0.01548  time: 0.5767  data_time: 0.0039  lr: 5e-05  m
ax_mem: 1973M
[05/19 16:32:10 d2.utils.events]:  eta: 0:10:08  iter: 1439  total_loss: 0.91
15  loss_cls: 0.2786  loss_box_reg: 0.4367  loss_mask: 0.2042  loss_rpn_cls:
0.003848  loss_rpn_loc: 0.02047  time: 0.5773  data_time: 0.0043  lr: 5e-05
max_mem: 1973M
[05/19 16:32:22 d2.utils.events]:  eta: 0:09:56  iter: 1459  total_loss: 1.06
6  loss_cls: 0.2925  loss_box_reg: 0.4858  loss_mask: 0.2318  loss_rpn_cls: 0
.01086  loss_rpn_loc: 0.01477  time: 0.5772  data_time: 0.0039  lr: 5e-05  ma
x_mem: 1973M
[05/19 16:32:34 d2.utils.events]:  eta: 0:09:45  iter: 1479  total_loss: 0.81
48  loss_cls: 0.234  loss_box_reg: 0.3763  loss_mask: 0.1899  loss_rpn_cls: 0
.002402  loss_rpn_loc: 0.01685  time: 0.5774  data_time: 0.0039  lr: 5e-05  m
ax_mem: 1973M
[05/19 16:32:46 d2.utils.events]:  eta: 0:09:34  iter: 1499  total_loss: 1.07
7  loss_cls: 0.3001  loss_box_reg: 0.4956  loss_mask: 0.2005  loss_rpn_cls: 0
.009581  loss_rpn_loc: 0.02184  time: 0.5776  data_time: 0.0037  lr: 5e-05  m
ax_mem: 1973M
[05/19 16:32:57 d2.utils.events]:  eta: 0:09:23  iter: 1519  total_loss: 0.88
64  loss_cls: 0.2222  loss_box_reg: 0.4327  loss_mask: 0.2237  loss_rpn_cls:
0.00247  loss_rpn_loc: 0.01417  time: 0.5774  data_time: 0.0039  lr: 5e-05  m
ax_mem: 1973M
[05/19 16:33:08 d2.utils.events]:  eta: 0:09:11  iter: 1539  total_loss: 0.85
47  loss_cls: 0.2173  loss_box_reg: 0.4033  loss_mask: 0.1721  loss_rpn_cls:
0.005746  loss_rpn_loc: 0.01305  time: 0.5774  data_time: 0.0038  lr: 5e-05
max_mem: 1973M
[05/19 16:33:20 d2.utils.events]:  eta: 0:09:00  iter: 1559  total_loss: 0.87
77  loss_cls: 0.2639  loss_box_reg: 0.3845  loss_mask: 0.1809  loss_rpn_cls:
0.004612  loss_rpn_loc: 0.0131  time: 0.5773  data_time: 0.0042  lr: 5e-05  m
ax_mem: 1973M
[05/19 16:33:31 d2.utils.events]:  eta: 0:08:48  iter: 1579  total_loss: 0.88
54  loss_cls: 0.2309  loss_box_reg: 0.3992  loss_mask: 0.2042  loss_rpn_cls:
0.003125  loss_rpn_loc: 0.01559  time: 0.5773  data_time: 0.0038  lr: 5e-05
max_mem: 1973M
```

```
[05/19 16:33:44 d2.data.datasets.coco]: Loaded 11 images in COCO format from
damage_dataset/val/COCO_mul_val_annos.json
[05/19 16:33:44 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used i
n inference: [ResizeShortestEdge(short_edge_length=(800, 800), max_size=1333,
sample_style='choice')]
[05/19 16:33:44 d2.data.common]: Serializing 11 elements to byte tensors and
concatenating them all ...
[05/19 16:33:44 d2.data.common]: Serialized dataset takes 0.01 MiB
[05/19 16:33:44 d2.evaluation.evaluator]: Start inference on 11 batches
[05/19 16:33:51 d2.evaluation.evaluator]: Inference done 11/11. Dataloading:
0.0018 s/iter. Inference: 0.3182 s/iter. Eval: 0.1946 s/iter. Total: 0.5146 s
/iter. ETA=0:00:00
[05/19 16:33:51 d2.evaluation.evaluator]: Total inference time: 0:00:03.17825
2 (0.529709 s / iter per device, on 1 devices)
[05/19 16:33:51 d2.evaluation.evaluator]: Total inference pure compute time:
0:00:01 (0.318199 s / iter per device, on 1 devices)
[05/19 16:33:51 d2.evaluation.coco_evaluation]: Preparing results for COCO fo
rmat ...
[05/19 16:33:51 d2.evaluation.coco_evaluation]: Saving results to coco_eval/c
oco_instances_results.json
[05/19 16:33:51 d2.evaluation.coco_evaluation]: Evaluating predictions with u
nofficial COCO API...
Loading and preparing results...
DONE (t=0.00s)
creating index...
index created!
[05/19 16:33:51 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[05/19 16:33:51 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.03 seconds.
[05/19 16:33:51 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:33:51 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.04 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.2
91
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.5
04
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.2
57
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.4
22
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.2
68
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.3
68
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.5
23
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.5
26
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
```

```
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.4
57
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.5
56
[05/19 16:33:51 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 29.125 | 50.371 | 25.720 |  nan  | 42.205 | 26.806 |
[05/19 16:33:51 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:33:51 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category     | AP     | category   | AP     |
|:-----------|:-------|:-------------|:-------|:-----------|:-------|
| headlamp   | 31.565 | rear_bumper  | 22.773 | door       | 33.574 |
| hood       | 33.011 | front_bumper | 24.702 |            |        |
Loading and preparing results...
DONE (t=0.02s)
creating index...
index created!
[05/19 16:33:51 d2.evaluation.fast_eval_api]: Evaluate annotation type *segm*
[05/19 16:33:51 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.03 seconds.
[05/19 16:33:51 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:33:51 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.03 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.3
40
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.4
82
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.3
86
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.4
84
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.3
00
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.4
11
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.5
22
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.5
25
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.5
40
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.5
24
[05/19 16:33:51 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
```

| 34.010 | 48.189 | 38.610 | nan | 48.440 | 29.979 |
[05/19 16:33:51 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[05/19 16:33:51 d2.evaluation.coco_evaluation]: Per-category segm AP:
category	AP	category	AP	category	AP
headlamp	30.611	rear_bumper	21.404	door	43.500
hood	36.845	front_bumper	37.690		
[05/19 16:33:51 d2.engine.defaults]: Evaluation results for car_part_val in csv format:
[05/19 16:33:51 d2.evaluation.testing]: copypaste: Task: bbox
[05/19 16:33:51 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:33:51 d2.evaluation.testing]: copypaste: 29.1247,50.3714,25.7201,nan,42.2046,26.8057
[05/19 16:33:51 d2.evaluation.testing]: copypaste: Task: segm
[05/19 16:33:51 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:33:51 d2.evaluation.testing]: copypaste: 34.0100,48.1887,38.6096,nan,48.4404,29.9786
[05/19 16:33:51 d2.utils.events]: eta: 0:08:37 iter: 1599 total_loss: 0.8612 loss_cls: 0.2212 loss_box_reg: 0.3836 loss_mask: 0.1931 loss_rpn_cls: 0.003225 loss_rpn_loc: 0.01717 time: 0.5774 data_time: 0.0038 lr: 5e-05 max_mem: 1973M
[05/19 16:34:03 d2.utils.events]: eta: 0:08:26 iter: 1619 total_loss: 0.7946 loss_cls: 0.1946 loss_box_reg: 0.3684 loss_mask: 0.1845 loss_rpn_cls: 0.003005 loss_rpn_loc: 0.01859 time: 0.5779 data_time: 0.0043 lr: 5e-05 max_mem: 1973M
[05/19 16:34:15 d2.utils.events]: eta: 0:08:14 iter: 1639 total_loss: 0.7232 loss_cls: 0.2119 loss_box_reg: 0.3667 loss_mask: 0.1762 loss_rpn_cls: 0.002419 loss_rpn_loc: 0.01465 time: 0.5777 data_time: 0.0041 lr: 5e-05 max_mem: 1973M
[05/19 16:34:27 d2.utils.events]: eta: 0:08:03 iter: 1659 total_loss: 0.7122 loss_cls: 0.2015 loss_box_reg: 0.3879 loss_mask: 0.1729 loss_rpn_cls: 0.007644 loss_rpn_loc: 0.01507 time: 0.5779 data_time: 0.0037 lr: 5e-05 max_mem: 1973M
[05/19 16:34:38 d2.utils.events]: eta: 0:07:53 iter: 1679 total_loss: 0.7054 loss_cls: 0.1775 loss_box_reg: 0.3209 loss_mask: 0.1508 loss_rpn_cls: 0.004641 loss_rpn_loc: 0.01073 time: 0.5780 data_time: 0.0039 lr: 5e-05 max_mem: 1973M
[05/19 16:34:50 d2.utils.events]: eta: 0:07:41 iter: 1699 total_loss: 0.7431 loss_cls: 0.181 loss_box_reg: 0.3728 loss_mask: 0.1941 loss_rpn_cls: 0.006242 loss_rpn_loc: 0.0174 time: 0.5779 data_time: 0.0039 lr: 5e-05 max_mem: 1973M
[05/19 16:35:01 d2.utils.events]: eta: 0:07:30 iter: 1719 total_loss: 0.8182 loss_cls: 0.2189 loss_box_reg: 0.3448 loss_mask: 0.1693 loss_rpn_cls: 0.01335 loss_rpn_loc: 0.0181 time: 0.5776 data_time: 0.0038 lr: 5e-05 max_mem: 1973M
[05/19 16:35:13 d2.utils.events]: eta: 0:07:18 iter: 1739 total_loss: 0.7425 loss_cls: 0.179 loss_box_reg: 0.3365 loss_mask: 0.1874 loss_rpn_cls: 0.001968 loss_rpn_loc: 0.01409 time: 0.5779 data_time: 0.0042 lr: 5e-05 max_mem: 1973M
[05/19 16:35:25 d2.utils.events]: eta: 0:07:06 iter: 1759 total_loss: 0.7746 loss_cls: 0.1957 loss_box_reg: 0.3579 loss_mask: 0.1805 loss_rpn_cls:

0.001925 loss_rpn_loc: 0.01719 time: 0.5783 data_time: 0.0038 lr: 5e-05
max_mem: 1973M
[05/19 16:35:36 d2.utils.events]: eta: 0:06:55 iter: 1779 total_loss: 0.68
3 loss_cls: 0.1724 loss_box_reg: 0.3378 loss_mask: 0.1588 loss_rpn_cls: 0
.005584 loss_rpn_loc: 0.01703 time: 0.5780 data_time: 0.0037 lr: 5e-05 m
ax_mem: 1973M
[05/19 16:35:48 d2.utils.events]: eta: 0:06:43 iter: 1799 total_loss: 0.68
04 loss_cls: 0.1594 loss_box_reg: 0.3313 loss_mask: 0.1376 loss_rpn_cls:
0.003003 loss_rpn_loc: 0.0133 time: 0.5781 data_time: 0.0039 lr: 5e-05 m
ax_mem: 1973M
[05/19 16:35:59 d2.utils.events]: eta: 0:06:31 iter: 1819 total_loss: 0.80
21 loss_cls: 0.1872 loss_box_reg: 0.3782 loss_mask: 0.1791 loss_rpn_cls:
0.006496 loss_rpn_loc: 0.0139 time: 0.5779 data_time: 0.0040 lr: 5e-05 m
ax_mem: 1973M
[05/19 16:36:11 d2.utils.events]: eta: 0:06:20 iter: 1839 total_loss: 0.60
35 loss_cls: 0.1458 loss_box_reg: 0.2211 loss_mask: 0.1479 loss_rpn_cls:
0.001613 loss_rpn_loc: 0.009983 time: 0.5780 data_time: 0.0039 lr: 5e-05
max_mem: 1973M
[05/19 16:36:23 d2.utils.events]: eta: 0:06:08 iter: 1859 total_loss: 0.68
76 loss_cls: 0.1643 loss_box_reg: 0.3071 loss_mask: 0.1661 loss_rpn_cls:
0.002917 loss_rpn_loc: 0.01456 time: 0.5781 data_time: 0.0038 lr: 5e-05
max_mem: 1973M
[05/19 16:36:34 d2.utils.events]: eta: 0:05:57 iter: 1879 total_loss: 0.71
87 loss_cls: 0.1678 loss_box_reg: 0.3363 loss_mask: 0.1622 loss_rpn_cls:
0.0009991 loss_rpn_loc: 0.01448 time: 0.5781 data_time: 0.0039 lr: 5e-05
max_mem: 1973M
[05/19 16:36:46 d2.utils.events]: eta: 0:05:46 iter: 1899 total_loss: 0.65
89 loss_cls: 0.1693 loss_box_reg: 0.2975 loss_mask: 0.1414 loss_rpn_cls:
0.003699 loss_rpn_loc: 0.01363 time: 0.5782 data_time: 0.0039 lr: 5e-05
max_mem: 1973M
[05/19 16:36:58 d2.utils.events]: eta: 0:05:35 iter: 1919 total_loss: 0.69
59 loss_cls: 0.1645 loss_box_reg: 0.325 loss_mask: 0.1622 loss_rpn_cls: 0
.007974 loss_rpn_loc: 0.01651 time: 0.5784 data_time: 0.0040 lr: 5e-05 m
ax_mem: 1973M
[05/19 16:37:10 d2.utils.events]: eta: 0:05:23 iter: 1939 total_loss: 0.73
21 loss_cls: 0.1723 loss_box_reg: 0.3294 loss_mask: 0.2008 loss_rpn_cls:
0.009495 loss_rpn_loc: 0.01195 time: 0.5786 data_time: 0.0042 lr: 5e-05
max_mem: 1973M
[05/19 16:37:22 d2.utils.events]: eta: 0:05:12 iter: 1959 total_loss: 0.65
65 loss_cls: 0.1797 loss_box_reg: 0.2884 loss_mask: 0.1473 loss_rpn_cls:
0.002564 loss_rpn_loc: 0.01295 time: 0.5786 data_time: 0.0044 lr: 5e-05
max_mem: 1973M
[05/19 16:37:33 d2.utils.events]: eta: 0:05:00 iter: 1979 total_loss: 0.59
47 loss_cls: 0.1529 loss_box_reg: 0.2673 loss_mask: 0.1379 loss_rpn_cls:
0.002535 loss_rpn_loc: 0.01477 time: 0.5785 data_time: 0.0040 lr: 5e-05
max_mem: 1973M
[05/19 16:37:46 d2.data.datasets.coco]: Loaded 11 images in COCO format from
damage_dataset/val/COCO_mul_val_annos.json
[05/19 16:37:46 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used i
n inference: [ResizeShortestEdge(short_edge_length=(800, 800), max_size=1333,
sample_style='choice')]
[05/19 16:37:46 d2.data.common]: Serializing 11 elements to byte tensors and
concatenating them all ...

```
[05/19 16:37:46 d2.data.common]: Serialized dataset takes 0.01 MiB
[05/19 16:37:46 d2.evaluation.evaluator]: Start inference on 11 batches
[05/19 16:37:51 d2.evaluation.evaluator]: Inference done 11/11. Dataloading:
0.0016 s/iter. Inference: 0.3012 s/iter. Eval: 0.1513 s/iter. Total: 0.4541 s
/iter. ETA=0:00:00
[05/19 16:37:51 d2.evaluation.evaluator]: Total inference time: 0:00:02.78967
1 (0.464945 s / iter per device, on 1 devices)
[05/19 16:37:51 d2.evaluation.evaluator]: Total inference pure compute time:
0:00:01 (0.301229 s / iter per device, on 1 devices)
[05/19 16:37:51 d2.evaluation.coco_evaluation]: Preparing results for COCO fo
rmat ...
[05/19 16:37:51 d2.evaluation.coco_evaluation]: Saving results to coco_eval/c
oco_instances_results.json
[05/19 16:37:51 d2.evaluation.coco_evaluation]: Evaluating predictions with u
nofficial COCO API...
Loading and preparing results...
DONE (t=0.00s)
creating index...
index created!
[05/19 16:37:51 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[05/19 16:37:51 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.02 seconds.
[05/19 16:37:51 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:37:51 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.01 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.3
09
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.5
29
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.2
63
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.4
16
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.2
79
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.3
48
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.5
08
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.5
15
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.4
82
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.5
29
[05/19 16:37:51 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
```

```
| 30.948 | 52.880 | 26.270 |  nan  | 41.602 | 27.902 |
[05/19 16:37:51 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:37:51 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category     | AP     | category   | AP     |
|:-----------|:-------|:-------------|:-------|:-----------|:-------|
| headlamp   | 35.509 | rear_bumper  | 8.583  | door       | 46.110 |
| hood       | 39.797 | front_bumper | 24.738 |            |        |
Loading and preparing results...
DONE (t=0.01s)
creating index...
index created!
[05/19 16:37:52 d2.evaluation.fast_eval_api]: Evaluate annotation type *segm*
[05/19 16:37:52 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.02 seconds.
[05/19 16:37:52 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:37:52 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.02 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.3
46
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.5
08
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.3
86
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.5
04
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.3
09
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.4
00
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.4
94
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.4
96
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.5
44
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.5
00
[05/19 16:37:52 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 34.630 | 50.803 | 38.579 |  nan  | 50.369 | 30.941 |
[05/19 16:37:52 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:37:52 d2.evaluation.coco_evaluation]: Per-category segm AP:
| category   | AP     | category     | AP     | category   | AP     |
|:-----------|:-------|:-------------|:-------|:-----------|:-------|
| headlamp   | 35.879 | rear_bumper  | 6.763  | door       | 47.870 |
```

| hood | 40.495 | front_bumper | 42.144 | | |
[05/19 16:37:52 d2.engine.defaults]: Evaluation results for car_part_val in csv format:
[05/19 16:37:52 d2.evaluation.testing]: copypaste: Task: bbox
[05/19 16:37:52 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:37:52 d2.evaluation.testing]: copypaste: 30.9477,52.8799,26.2704,nan,41.6017,27.9022
[05/19 16:37:52 d2.evaluation.testing]: copypaste: Task: segm
[05/19 16:37:52 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:37:52 d2.evaluation.testing]: copypaste: 34.6302,50.8032,38.5791,nan,50.3686,30.9412
[05/19 16:37:52 d2.utils.events]: eta: 0:04:48 iter: 1999 total_loss: 0.6315 loss_cls: 0.1668 loss_box_reg: 0.2878 loss_mask: 0.1624 loss_rpn_cls: 0.004165 loss_rpn_loc: 0.0109 time: 0.5786 data_time: 0.0044 lr: 5e-05 max_mem: 1973M
[05/19 16:38:03 d2.utils.events]: eta: 0:04:36 iter: 2019 total_loss: 0.5999 loss_cls: 0.1694 loss_box_reg: 0.2643 loss_mask: 0.1311 loss_rpn_cls: 0.002982 loss_rpn_loc: 0.0114 time: 0.5785 data_time: 0.0048 lr: 2.5e-06 max_mem: 1973M
[05/19 16:38:15 d2.utils.events]: eta: 0:04:25 iter: 2039 total_loss: 0.6916 loss_cls: 0.1653 loss_box_reg: 0.2958 loss_mask: 0.1475 loss_rpn_cls: 0.003466 loss_rpn_loc: 0.01282 time: 0.5785 data_time: 0.0045 lr: 2.5e-06 max_mem: 1973M
[05/19 16:38:27 d2.utils.events]: eta: 0:04:13 iter: 2059 total_loss: 0.5947 loss_cls: 0.1579 loss_box_reg: 0.2775 loss_mask: 0.1609 loss_rpn_cls: 0.001282 loss_rpn_loc: 0.01237 time: 0.5787 data_time: 0.0041 lr: 2.5e-06 max_mem: 1973M
[05/19 16:38:38 d2.utils.events]: eta: 0:04:02 iter: 2079 total_loss: 0.555 loss_cls: 0.1594 loss_box_reg: 0.2472 loss_mask: 0.1333 loss_rpn_cls: 0.001669 loss_rpn_loc: 0.01134 time: 0.5788 data_time: 0.0040 lr: 2.5e-06 max_mem: 1973M
[05/19 16:38:50 d2.utils.events]: eta: 0:03:50 iter: 2099 total_loss: 0.6481 loss_cls: 0.1681 loss_box_reg: 0.2757 loss_mask: 0.1586 loss_rpn_cls: 0.002496 loss_rpn_loc: 0.01087 time: 0.5787 data_time: 0.0040 lr: 2.5e-06 max_mem: 1973M
[05/19 16:39:01 d2.utils.events]: eta: 0:03:38 iter: 2119 total_loss: 0.5258 loss_cls: 0.1478 loss_box_reg: 0.2369 loss_mask: 0.133 loss_rpn_cls: 0.002972 loss_rpn_loc: 0.01127 time: 0.5786 data_time: 0.0040 lr: 2.5e-06 max_mem: 1973M
[05/19 16:39:13 d2.utils.events]: eta: 0:03:27 iter: 2139 total_loss: 0.6017 loss_cls: 0.1519 loss_box_reg: 0.2813 loss_mask: 0.1432 loss_rpn_cls: 0.0008025 loss_rpn_loc: 0.01235 time: 0.5785 data_time: 0.0040 lr: 2.5e-06 max_mem: 1973M
[05/19 16:39:24 d2.utils.events]: eta: 0:03:15 iter: 2159 total_loss: 0.5686 loss_cls: 0.1733 loss_box_reg: 0.2636 loss_mask: 0.1305 loss_rpn_cls: 0.002543 loss_rpn_loc: 0.01119 time: 0.5784 data_time: 0.0038 lr: 2.5e-06 max_mem: 1973M
[05/19 16:39:36 d2.utils.events]: eta: 0:03:04 iter: 2179 total_loss: 0.6461 loss_cls: 0.1594 loss_box_reg: 0.2894 loss_mask: 0.1761 loss_rpn_cls: 0.005262 loss_rpn_loc: 0.01564 time: 0.5785 data_time: 0.0040 lr: 2.5e-06 max_mem: 1973M
[05/19 16:39:47 d2.utils.events]: eta: 0:02:52 iter: 2199 total_loss: 0.555 loss_cls: 0.1336 loss_box_reg: 0.2472 loss_mask: 0.1453 loss_rpn_cls: 0

```
.0006854  loss_rpn_loc: 0.01164  time: 0.5785  data_time: 0.0038  lr: 2.5e-06
max_mem: 1973M
[05/19 16:39:59 d2.utils.events]:  eta: 0:02:41  iter: 2219  total_loss: 0.57
66  loss_cls: 0.168  loss_box_reg: 0.252  loss_mask: 0.1348  loss_rpn_cls: 0.
001924  loss_rpn_loc: 0.01417  time: 0.5784  data_time: 0.0037  lr: 2.5e-06
max_mem: 1973M
[05/19 16:40:10 d2.utils.events]:  eta: 0:02:29  iter: 2239  total_loss: 0.57
89  loss_cls: 0.1534  loss_box_reg: 0.2831  loss_mask: 0.1505  loss_rpn_cls:
0.007307  loss_rpn_loc: 0.0147  time: 0.5783  data_time: 0.0039  lr: 2.5e-06
max_mem: 1973M
[05/19 16:40:21 d2.utils.events]:  eta: 0:02:17  iter: 2259  total_loss: 0.69
54  loss_cls: 0.1608  loss_box_reg: 0.3095  loss_mask: 0.1469  loss_rpn_cls:
0.005565  loss_rpn_loc: 0.01313  time: 0.5782  data_time: 0.0037  lr: 2.5e-06
max_mem: 1973M
[05/19 16:40:33 d2.utils.events]:  eta: 0:02:06  iter: 2279  total_loss: 0.57
9  loss_cls: 0.1492  loss_box_reg: 0.2755  loss_mask: 0.1301  loss_rpn_cls: 0
.001439  loss_rpn_loc: 0.01454  time: 0.5784  data_time: 0.0037  lr: 2.5e-06
max_mem: 1973M
[05/19 16:40:45 d2.utils.events]:  eta: 0:01:54  iter: 2299  total_loss: 0.55
91  loss_cls: 0.1635  loss_box_reg: 0.2535  loss_mask: 0.1567  loss_rpn_cls:
0.0009827  loss_rpn_loc: 0.01301  time: 0.5786  data_time: 0.0039  lr: 2.5e-0
6  max_mem: 1973M
[05/19 16:40:57 d2.utils.events]:  eta: 0:01:43  iter: 2319  total_loss: 0.53
59  loss_cls: 0.1546  loss_box_reg: 0.2433  loss_mask: 0.1364  loss_rpn_cls:
0.002618  loss_rpn_loc: 0.01046  time: 0.5786  data_time: 0.0038  lr: 2.5e-06
max_mem: 1973M
[05/19 16:41:09 d2.utils.events]:  eta: 0:01:31  iter: 2339  total_loss: 0.57
59  loss_cls: 0.1673  loss_box_reg: 0.2731  loss_mask: 0.1443  loss_rpn_cls:
0.001201  loss_rpn_loc: 0.01203  time: 0.5787  data_time: 0.0040  lr: 2.5e-06
max_mem: 1973M
[05/19 16:41:21 d2.utils.events]:  eta: 0:01:20  iter: 2359  total_loss: 0.61
78  loss_cls: 0.1466  loss_box_reg: 0.2872  loss_mask: 0.1597  loss_rpn_cls:
0.004857  loss_rpn_loc: 0.01733  time: 0.5788  data_time: 0.0040  lr: 2.5e-06
max_mem: 1973M
[05/19 16:41:32 d2.utils.events]:  eta: 0:01:09  iter: 2379  total_loss: 0.57
3  loss_cls: 0.1553  loss_box_reg: 0.2319  loss_mask: 0.1307  loss_rpn_cls: 0
.001485  loss_rpn_loc: 0.0151  time: 0.5788  data_time: 0.0038  lr: 2.5e-06
max_mem: 1973M
[05/19 16:41:45 d2.data.datasets.coco]: Loaded 11 images in COCO format from
damage_dataset/val/COCO_mul_val_annos.json
[05/19 16:41:45 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used i
n inference: [ResizeShortestEdge(short_edge_length=(800, 800), max_size=1333,
sample_style='choice')]
[05/19 16:41:45 d2.data.common]: Serializing 11 elements to byte tensors and
concatenating them all ...
[05/19 16:41:45 d2.data.common]: Serialized dataset takes 0.01 MiB
[05/19 16:41:45 d2.evaluation.evaluator]: Start inference on 11 batches
[05/19 16:41:50 d2.evaluation.evaluator]: Inference done 11/11. Dataloading:
0.0018 s/iter. Inference: 0.3082 s/iter. Eval: 0.1520 s/iter. Total: 0.4621 s
/iter. ETA=0:00:00
[05/19 16:41:50 d2.evaluation.evaluator]: Total inference time: 0:00:02.87082
1 (0.478470 s / iter per device, on 1 devices)
```

```
[05/19 16:41:50 d2.evaluation.evaluator]: Total inference pure compute time:
0:00:01 (0.308245 s / iter per device, on 1 devices)
[05/19 16:41:50 d2.evaluation.coco_evaluation]: Preparing results for COCO fo
rmat ...
[05/19 16:41:50 d2.evaluation.coco_evaluation]: Saving results to coco_eval/c
oco_instances_results.json
[05/19 16:41:51 d2.evaluation.coco_evaluation]: Evaluating predictions with u
nofficial COCO API...
Loading and preparing results...
DONE (t=0.00s)
creating index...
index created!
[05/19 16:41:51 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[05/19 16:41:51 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.04 seconds.
[05/19 16:41:51 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:41:51 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.04 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.3
40
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.5
31
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.3
29
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.4
13
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.3
23
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.4
06
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.5
34
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.5
40
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.4
55
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.5
72
[05/19 16:41:51 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 34.003 | 53.125 | 32.918 |  nan  | 41.337 | 32.271 |
[05/19 16:41:51 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:41:51 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category     | AP     | category   | AP     |
|:-----------|:-------|:-------------|:-------|:-----------|:-------|
| headlamp   | 35.239 | rear_bumper  | 9.193  | door       | 43.313 |
```

| hood | 43.911 | front_bumper | 38.359 | | |
Loading and preparing results...
DONE (t=0.02s)
creating index...
index created!
[05/19 16:41:51 d2.evaluation.fast_eval_api]: Evaluate annotation type *segm*
[05/19 16:41:51 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.04 seconds.
[05/19 16:41:51 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
[05/19 16:41:51 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.04 seconds.
 Average Precision (AP) @[IoU=0.50:0.95 | area=   all | maxDets=100] = 0.345
 Average Precision (AP) @[IoU=0.50 | area=   all | maxDets=100] = 0.513
 Average Precision (AP) @[IoU=0.75 | area=   all | maxDets=100] = 0.392
 Average Precision (AP) @[IoU=0.50:0.95 | area= small | maxDets=100] = -1.000
 Average Precision (AP) @[IoU=0.50:0.95 | area=medium | maxDets=100] = 0.504
 Average Precision (AP) @[IoU=0.50:0.95 | area= large | maxDets=100] = 0.313
 Average Recall   (AR) @[IoU=0.50:0.95 | area=   all | maxDets= 1] = 0.400
 Average Recall   (AR) @[IoU=0.50:0.95 | area=   all | maxDets= 10] = 0.490
 Average Recall   (AR) @[IoU=0.50:0.95 | area=   all | maxDets=100] = 0.492
 Average Recall   (AR) @[IoU=0.50:0.95 | area= small | maxDets=100] = -1.000
 Average Recall   (AR) @[IoU=0.50:0.95 | area=medium | maxDets=100] = 0.540
 Average Recall   (AR) @[IoU=0.50:0.95 | area= large | maxDets=100] = 0.497
[05/19 16:41:51 d2.evaluation.coco_evaluation]: Evaluation results for segm:
AP	AP50	AP75	APs	APm	APl
34.508	51.254	39.222	nan	50.392	31.283
[05/19 16:41:51 d2.evaluation.coco_evaluation]: Some metrics cannot be computed and is shown as NaN.
[05/19 16:41:51 d2.evaluation.coco_evaluation]: Per-category segm AP:
category	AP	category	AP	category	AP
headlamp	35.847	rear_bumper	6.763	door	46.759
hood	40.495	front_bumper	42.678		
[05/19 16:41:51 d2.engine.defaults]: Evaluation results for car_part_val in csv format:
[05/19 16:41:51 d2.evaluation.testing]: copypaste: Task: bbox
[05/19 16:41:51 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:41:51 d2.evaluation.testing]: copypaste: 34.0030,53.1246,32.9182,nan,41.3368,32.2706

```
[05/19 16:41:51 d2.evaluation.testing]: copypaste: Task: segm
[05/19 16:41:51 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:41:51 d2.evaluation.testing]: copypaste: 34.5084,51.2536,39.2221,na
n,50.3916,31.2825
[05/19 16:41:51 d2.utils.events]:  eta: 0:00:57  iter: 2399  total_loss: 0.61
83  loss_cls: 0.1444  loss_box_reg: 0.2721  loss_mask: 0.1484  loss_rpn_cls:
0.001132  loss_rpn_loc: 0.01297  time: 0.5788  data_time: 0.0039  lr: 2.5e-06
max_mem: 1973M
[05/19 16:42:03 d2.utils.events]:  eta: 0:00:46  iter: 2419  total_loss: 0.66
97  loss_cls: 0.1661  loss_box_reg: 0.311  loss_mask: 0.1546  loss_rpn_cls: 0
.008387  loss_rpn_loc: 0.01424  time: 0.5791  data_time: 0.0044  lr: 2.5e-06
max_mem: 1973M
[05/19 16:42:15 d2.utils.events]:  eta: 0:00:34  iter: 2439  total_loss: 0.59
41  loss_cls: 0.1529  loss_box_reg: 0.2742  loss_mask: 0.1447  loss_rpn_cls:
0.001156  loss_rpn_loc: 0.01217  time: 0.5791  data_time: 0.0038  lr: 2.5e-06
max_mem: 1973M
[05/19 16:42:26 d2.utils.events]:  eta: 0:00:23  iter: 2459  total_loss: 0.52
04  loss_cls: 0.1442  loss_box_reg: 0.2447  loss_mask: 0.1214  loss_rpn_cls:
0.0012  loss_rpn_loc: 0.01345  time: 0.5790  data_time: 0.0038  lr: 2.5e-06
max_mem: 1973M
[05/19 16:42:38 d2.utils.events]:  eta: 0:00:11  iter: 2479  total_loss: 0.65
82  loss_cls: 0.1742  loss_box_reg: 0.3016  loss_mask: 0.1681  loss_rpn_cls:
0.002508  loss_rpn_loc: 0.01202  time: 0.5790  data_time: 0.0039  lr: 2.5e-06
max_mem: 1973M
[05/19 16:42:50 d2.utils.events]:  eta: 0:00:00  iter: 2499  total_loss: 0.59
7  loss_cls: 0.1434  loss_box_reg: 0.265  loss_mask: 0.1635  loss_rpn_cls: 0.
002852  loss_rpn_loc: 0.009996  time: 0.5789  data_time: 0.0041  lr: 2.5e-06
max_mem: 1973M
[05/19 16:42:50 d2.engine.hooks]: Overall training speed: 2498 iterations in
0:24:06 (0.5789 s / it)
[05/19 16:42:50 d2.engine.hooks]: Total training time: 0:25:04 (0:00:58 on ho
oks)
[05/19 16:42:50 d2.data.datasets.coco]: Loaded 11 images in COCO format from
damage_dataset/val/COCO_mul_val_annos.json
[05/19 16:42:50 d2.data.dataset_mapper]: [DatasetMapper] Augmentations used i
n inference: [ResizeShortestEdge(short_edge_length=(800, 800), max_size=1333,
sample_style='choice')]
[05/19 16:42:50 d2.data.common]: Serializing 11 elements to byte tensors and
concatenating them all ...
[05/19 16:42:50 d2.data.common]: Serialized dataset takes 0.01 MiB
[05/19 16:42:50 d2.evaluation.evaluator]: Start inference on 11 batches
[05/19 16:42:56 d2.evaluation.evaluator]: Inference done 11/11. Dataloading:
0.0018 s/iter. Inference: 0.3048 s/iter. Eval: 0.1592 s/iter. Total: 0.4657 s
/iter. ETA=0:00:00
[05/19 16:42:56 d2.evaluation.evaluator]: Total inference time: 0:00:02.91283
1 (0.485472 s / iter per device, on 1 devices)
[05/19 16:42:56 d2.evaluation.evaluator]: Total inference pure compute time:
0:00:01 (0.304769 s / iter per device, on 1 devices)
[05/19 16:42:56 d2.evaluation.coco_evaluation]: Preparing results for COCO fo
rmat ...
[05/19 16:42:56 d2.evaluation.coco_evaluation]: Saving results to coco_eval/c
oco_instances_results.json
```

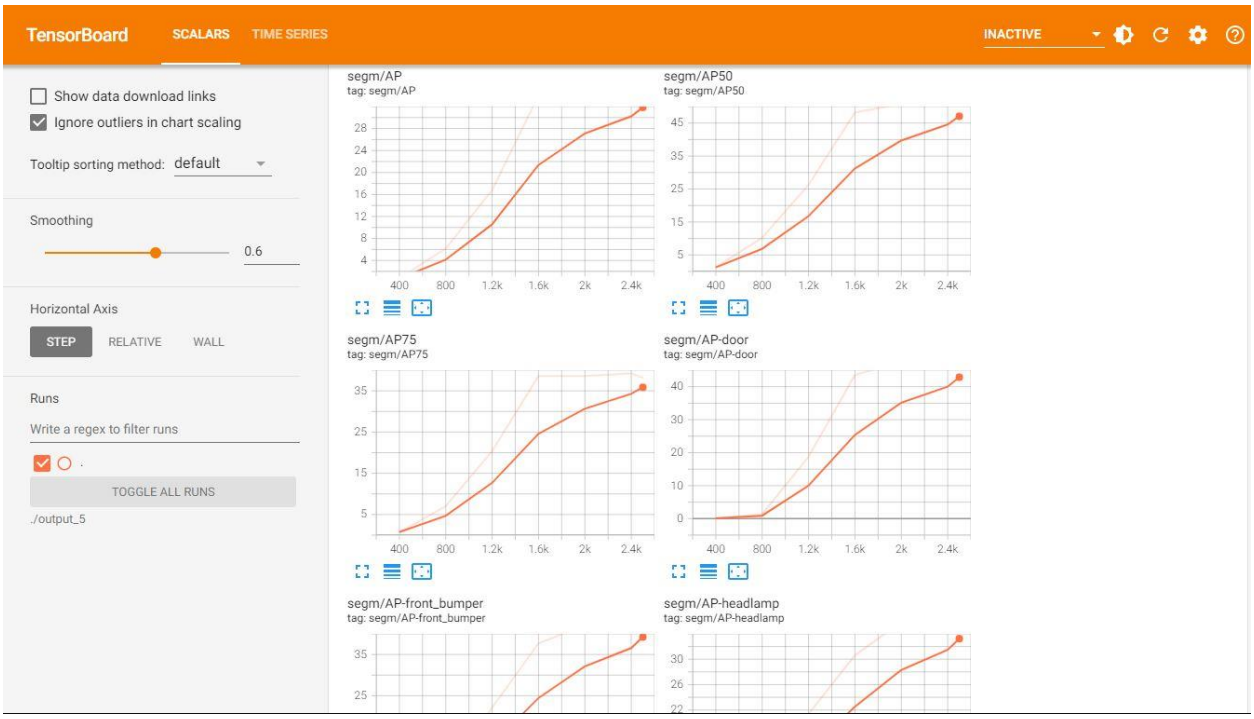
```
[05/19 16:42:56 d2.evaluation.coco_evaluation]: Evaluating predictions with u
nofficial COCO API...
Loading and preparing results...
DONE (t=0.00s)
creating index...
index created!
[05/19 16:42:56 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[05/19 16:42:56 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.04 seconds.
[05/19 16:42:56 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:42:56 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.05 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.3
28
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.5
26
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.3
18
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.4
04
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.3
11
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.4
00
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.5
28
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.5
34
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.4
45
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.5
69
[05/19 16:42:56 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 32.779 | 52.591 | 31.844 |  nan  | 40.371 | 31.089 |
[05/19 16:42:56 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:42:56 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP     | category     | AP     | category   | AP     |
|:-----------|:-------|:-------------|:-------|:-----------|:-------|
| headlamp   | 35.169 | rear_bumper  | 7.176  | door       | 39.921 |
| hood       | 44.522 | front_bumper | 37.109 |            |        |
Loading and preparing results...
DONE (t=0.02s)
creating index...
index created!
[05/19 16:42:56 d2.evaluation.fast_eval_api]: Evaluate annotation type *segm*
```

```
[05/19 16:42:56 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finishe
d in 0.04 seconds.
[05/19 16:42:56 d2.evaluation.fast_eval_api]: Accumulating evaluation results
...
[05/19 16:42:56 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finis
hed in 0.03 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.3
42
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.5
07
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.3
82
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.5
04
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.3
04
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.4
03
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.4
90
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.4
92
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.
000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.5
40
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.4
97
[05/19 16:42:56 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|   AP   |  AP50  |  AP75  |  APs  |  APm   |  APl   |
|:------:|:------:|:------:|:-----:|:------:|:------:|
| 34.190 | 50.657 | 38.152 |  nan  | 50.387 | 30.425 |
[05/19 16:42:56 d2.evaluation.coco_evaluation]: Some metrics cannot be comput
ed and is shown as NaN.
[05/19 16:42:56 d2.evaluation.coco_evaluation]: Per-category segm AP:
| category   | AP     | category     | AP     | category   | AP     |
|:-----------|:-------|:-------------|:-------|:-----------|:-------|
| headlamp   | 35.793 | rear_bumper  | 4.609  | door       | 46.983 |
| hood       | 40.495 | front_bumper | 43.070 |            |        |
[05/19 16:42:56 d2.engine.defaults]: Evaluation results for car_part_val in c
sv format:
[05/19 16:42:56 d2.evaluation.testing]: copypaste: Task: bbox
[05/19 16:42:56 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:42:56 d2.evaluation.testing]: copypaste: 32.7794,52.5907,31.8439,na
n,40.3710,31.0894
[05/19 16:42:56 d2.evaluation.testing]: copypaste: Task: segm
[05/19 16:42:56 d2.evaluation.testing]: copypaste: AP,AP50,AP75,APs,APm,APl
[05/19 16:42:56 d2.evaluation.testing]: copypaste: 34.1899,50.6567,38.1518,na
n,50.3871,30.4249
```

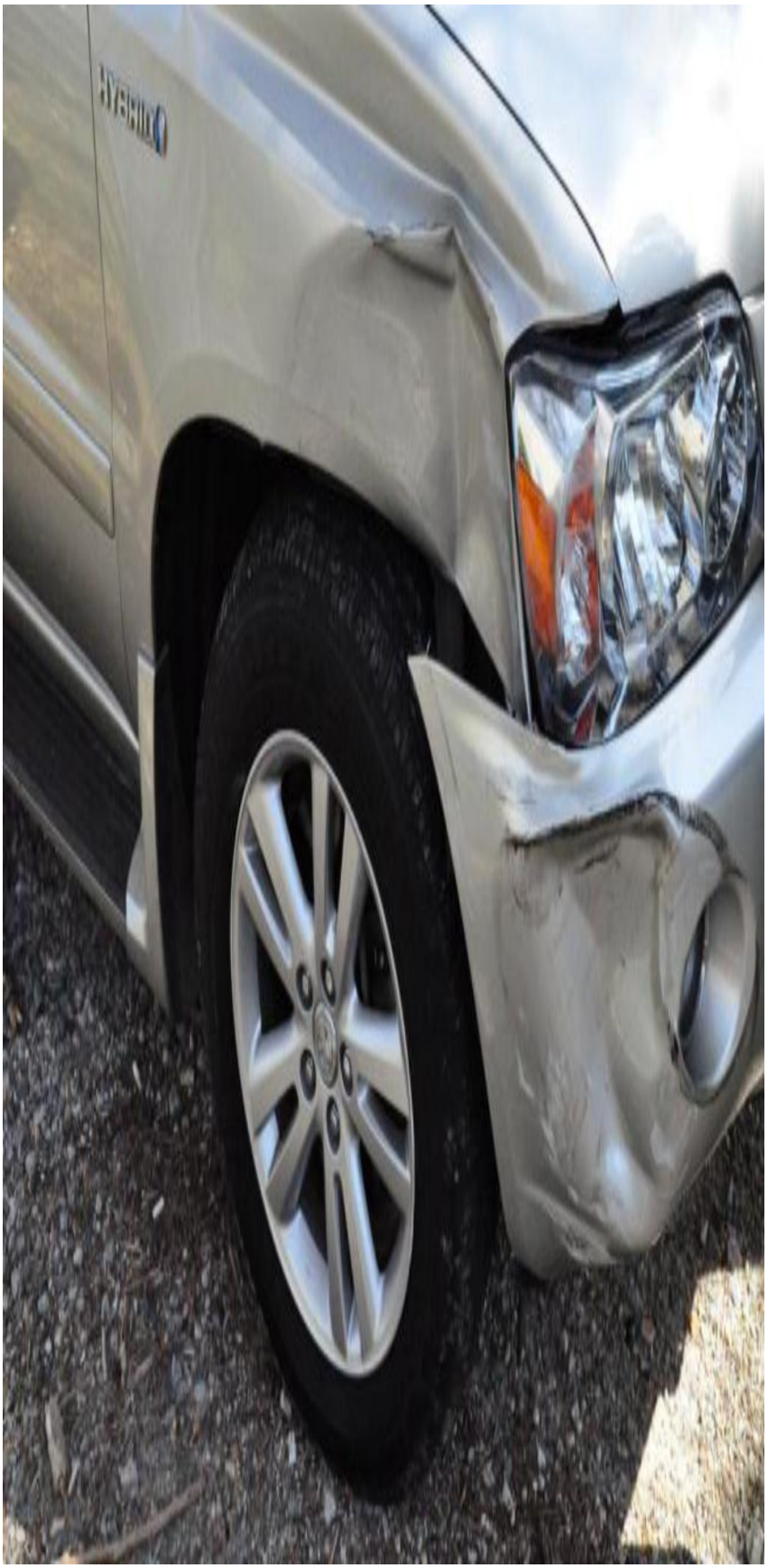
In [ ]:

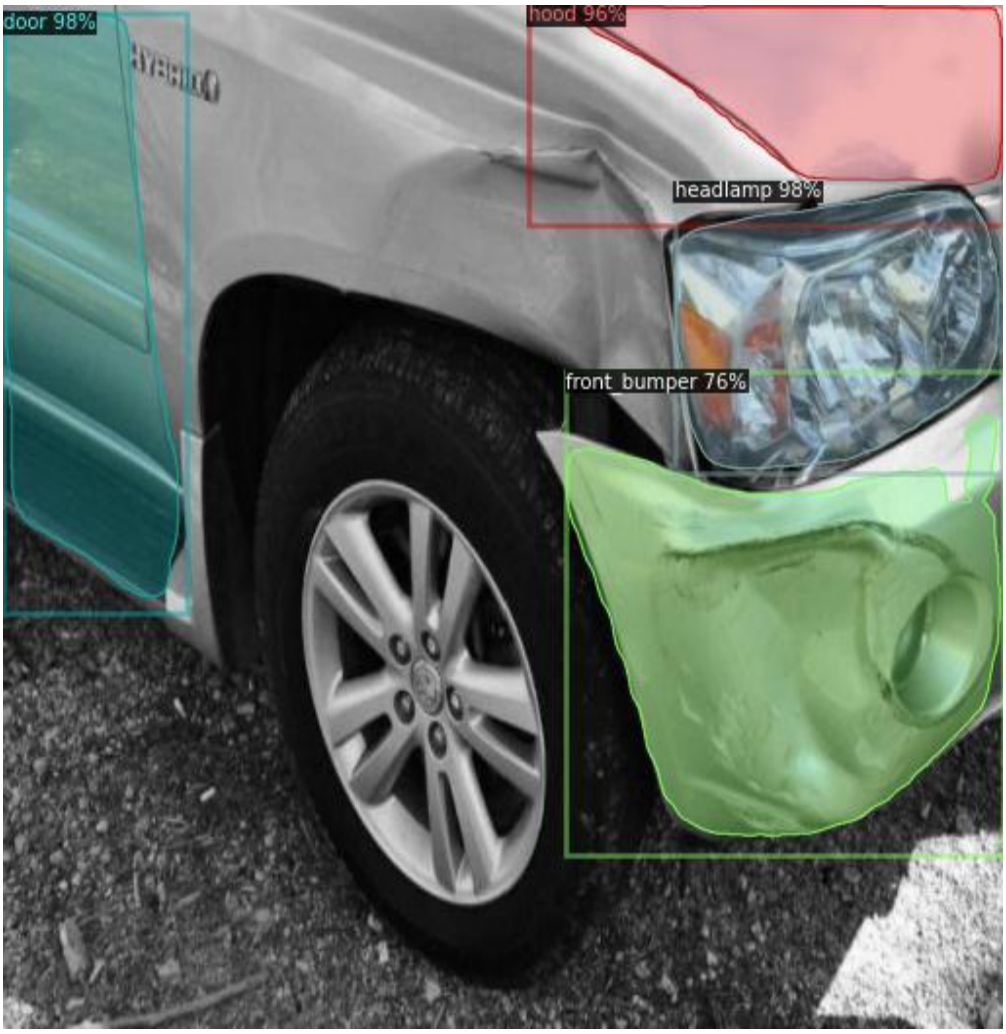
```
%reload_ext tensorboard
%tensorboard --logdir ./output_2
```

Average Precision vs Epochs



Loss vs Epochs





Predictions of car part detection model

Preparing test data

As seen from above model **2** detects two moderate damages and model **3** detects 4 car parts among which **front bumper** segmentation is **overlapping** with one **damage** segmentation. Hence we calculate **dice coefficient** and put it under “front_bumper_dice” column.

Another damage in the front fender (beside headlamp) does not overlap any segmentation from the second model because the training data did not have any annotations for the **fender** of any car. Thus it will go to the unknown column.
Test data of the above image

Final prediction of price

Now as mentioned earlier the training dataset was **too small** to fit a model so instead of fitting a model (supposedly model **1**), for every row in test data I calculated mean price of identical rows in train data. In the end I added all the estimated prices from each row to get the total price.

Future work

- A large dataset would help us train an actual machine learning model for model **1** instead of the calculating mean as we did above.
- Proper annotation of all external parts of car will improve model **3** performance and we won't need the 'unknown' column in the train or test data. (Eg. In the above test example if the front fender of cars were annotated in train data, overlap would've happened between the damage and the fender segmentation outputs from the two models **2** & **3** respectively and we would've filled the 'fender_dice' column instead of the unknown column).

- Proper price estimates will make the model **1** outputs reliable and such data is only present with the insurance company.
- In reality price of repairing a car does not only depend on the damages but on many other factors too like - **brand of car**, **age**, **paint color**, **model number** & so on. These too are available to the insurance company and we can add these as features to the train data to improve model performance.
- We can also use **IOU** score to find the percentage of overlap between two segmentation masks.
- Segmented damaged area images can also be added to the training dataset for every row which would make use of **CNN+MLP** model to train model **1**.
- It only makes sense to find the percentage overlap using dice coefficient if the damage as well as the whole car part where the damage has occurred is visible (Eg. In the above test example the whole front bumper of the car is not visible, therefore the dice coefficient between the damage and the front bumper doesn't make much sense).

Deployment

The code can be found here. I have deployed the model using Flask framework. Follow the steps on GitHub to deploy the model in your local machine. This is a short video of my web-app:

Deployed w