```
!pip install torch torchvision
!git clone https://github.com/facebookresearch/detectron2.git
%cd detectron2
!python -m pip install -e .
```

Show hidden output

```
import torch
from detectron2.engine import DefaultTrainer, DefaultPredictor
from detectron2.config import get_cfg
from detectron2.data import DatasetCatalog, MetadataCatalog, build_detection_train_loader
from detectron2.data.datasets import register_coco_instances
from detectron2.utils.visualizer import Visualizer
import cv2
import matplotlib.pyplot as plt
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Register the dataset
register_coco_instances("car_parts_train", {}, "/content/drive/MyDrive/CarSegmentation/trainingset/annotations.json", "/content/drive/MyDriv
register_coco_instances("car_parts_val", {}, "/content/drive/MyDrive/CarSegmentation/testset/annotations.json", "/content/drive/MyDrive/CarS
```

```
car_parts_metadata = MetadataCatalog.get("car_parts_train")
dataset_dicts = DatasetCatalog.get("car_parts_train")
```

```
cfg = get_cfg()
cfg.merge_from_file("configs/COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")  # e.g., COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x
cfg.DATASETS.TRAIN = ("car_parts_train",)
cfg.DATASETS.TEST = ("car_parts_val",)
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = "detectron2://COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x/137849600/model_final_f10217.pkl"  # pre-trained model
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.00025
cfg.SOLVER.MAX_ITER = 10000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 18  #
```

```
# Disable GPU usage
cfg.MODEL.DEVICE = "cpu"
```

```
# Train the model
trainer = DefaultTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```

```
[08/05 11:41:24 d2.utils.events]:  eta: 0:09:18  iter: 8439  total_loss: 0.4227  loss_cls: 0.1215  loss_box_reg: 0.169  loss_mask: 0.1
[08/05 11:41:31 d2.utils.events]:  eta: 0:09:11  iter: 8459  total_loss: 0.4758  loss_cls: 0.1246  loss_box_reg: 0.1874  loss_mask: 0.
[08/05 11:41:38 d2.utils.events]:  eta: 0:09:04  iter: 8479  total_loss: 0.4186  loss_cls: 0.1206  loss_box_reg: 0.168  loss_mask: 0.1
[08/05 11:41:46 d2.utils.events]:  eta: 0:08:57  iter: 8499  total_loss: 0.4443  loss_cls: 0.1142  loss_box_reg: 0.1805  loss_mask: 0.
[08/05 11:41:53 d2.utils.events]:  eta: 0:08:50  iter: 8519  total_loss: 0.4448  loss_cls: 0.1117  loss_box_reg: 0.1702  loss_mask: 0.
[08/05 11:42:00 d2.utils.events]:  eta: 0:08:42  iter: 8539  total_loss: 0.4689  loss_cls: 0.1416  loss_box_reg: 0.1817  loss_mask: 0.
[08/05 11:42:08 d2.utils.events]:  eta: 0:08:35  iter: 8559  total_loss: 0.4158  loss_cls: 0.1179  loss_box_reg: 0.168  loss_mask: 0.1
[08/05 11:42:15 d2.utils.events]:  eta: 0:08:28  iter: 8579  total_loss: 0.4261  loss_cls: 0.1283  loss_box_reg: 0.1714  loss_mask: 0.
[08/05 11:42:22 d2.utils.events]:  eta: 0:08:21  iter: 8599  total_loss: 0.455  loss_cls: 0.1372  loss_box_reg: 0.1788  loss_mask: 0.1
[08/05 11:42:29 d2.utils.events]:  eta: 0:08:13  iter: 8619  total_loss: 0.4448  loss_cls: 0.1323  loss_box_reg: 0.1781  loss_mask: 0.
[08/05 11:42:36 d2.utils.events]:  eta: 0:08:06  iter: 8639  total_loss: 0.4452  loss_cls: 0.114  loss_box_reg: 0.1731  loss_mask: 0.1
[08/05 11:42:44 d2.utils.events]:  eta: 0:07:59  iter: 8659  total_loss: 0.4084  loss_cls: 0.1128  loss_box_reg: 0.1653  loss_mask: 0.
[08/05 11:42:51 d2.utils.events]:  eta: 0:07:52  iter: 8679  total_loss: 0.4484  loss_cls: 0.1353  loss_box_reg: 0.1708  loss_mask: 0.
[08/05 11:42:58 d2.utils.events]:  eta: 0:07:44  iter: 8699  total_loss: 0.4294  loss_cls: 0.1257  loss_box_reg: 0.1739  loss_mask: 0.
[08/05 11:43:05 d2.utils.events]:  eta: 0:07:37  iter: 8719  total_loss: 0.4467  loss_cls: 0.1221  loss_box_reg: 0.1765  loss_mask: 0.
[08/05 11:43:12 d2.utils.events]:  eta: 0:07:30  iter: 8739  total_loss: 0.4371  loss_cls: 0.1165  loss_box_reg: 0.1757  loss_mask: 0.
[08/05 11:43:19 d2.utils.events]:  eta: 0:07:23  iter: 8759  total_loss: 0.4272  loss_cls: 0.1238  loss_box_reg: 0.1648  loss_mask: 0.
[08/05 11:43:26 d2.utils.events]:  eta: 0:07:15  iter: 8779  total_loss: 0.4307  loss_cls: 0.1175  loss_box_reg: 0.1667  loss_mask: 0.
[08/05 11:43:34 d2.utils.events]:  eta: 0:07:09  iter: 8799  total_loss: 0.4351  loss_cls: 0.1181  loss_box_reg: 0.1768  loss_mask: 0.
[08/05 11:43:41 d2.utils.events]:  eta: 0:07:01  iter: 8819  total_loss: 0.4355  loss_cls: 0.1227  loss_box_reg: 0.1636  loss_mask: 0.
[08/05 11:43:49 d2.utils.events]:  eta: 0:06:54  iter: 8839  total_loss: 0.4196  loss_cls: 0.1121  loss_box_reg: 0.1748  loss_mask: 0.
[08/05 11:43:56 d2.utils.events]:  eta: 0:06:47  iter: 8859  total_loss: 0.4092  loss_cls: 0.1108  loss_box_reg: 0.1698  loss_mask: 0.
[08/05 11:44:03 d2.utils.events]:  eta: 0:06:40  iter: 8879  total_loss: 0.4284  loss_cls: 0.1166  loss_box_reg: 0.1699  loss_mask: 0.
[08/05 11:44:10 d2.utils.events]:  eta: 0:06:33  iter: 8899  total_loss: 0.4318  loss_cls: 0.119  loss_box_reg: 0.165  loss_mask: 0.10
[08/05 11:44:18 d2.utils.events]:  eta: 0:06:26  iter: 8919  total_loss: 0.4206  loss_cls: 0.1157  loss_box_reg: 0.1772  loss_mask: 0.
[08/05 11:44:25 d2.utils.events]:  eta: 0:06:19  iter: 8939  total_loss: 0.4215  loss_cls: 0.1067  loss_box_reg: 0.1723  loss_mask: 0.
[08/05 11:44:32 d2.utils.events]:  eta: 0:06:12  iter: 8959  total_loss: 0.4446  loss_cls: 0.1175  loss_box_reg: 0.1799  loss_mask: 0.
[08/05 11:44:39 d2.utils.events]:  eta: 0:06:05  iter: 8979  total_loss: 0.424  loss_cls: 0.1197  loss_box_reg: 0.1742  loss_mask: 0.1
[08/05 11:44:46 d2.utils.events]:  eta: 0:05:58  iter: 8999  total_loss: 0.4475  loss_cls: 0.1267  loss_box_reg: 0.1742  loss_mask: 0.
[08/05 11:44:53 d2.utils.events]:  eta: 0:05:50  iter: 9019  total_loss: 0.4069  loss_cls: 0.115  loss_box_reg: 0.1607  loss_mask: 0.1
[08/05 11:45:01 d2.utils.events]:  eta: 0:05:43  iter: 9039  total_loss: 0.4124  loss_cls: 0.124  loss_box_reg: 0.1672  loss_mask: 0.1
[08/05 11:45:08 d2.utils.events]:  eta: 0:05:36  iter: 9059  total_loss: 0.4028  loss_cls: 0.1053  loss_box_reg: 0.1627  loss_mask: 0.
[08/05 11:45:15 d2.utils.events]:  eta: 0:05:29  iter: 9079  total_loss: 0.4168  loss_cls: 0.1068  loss_box_reg: 0.1692  loss_mask: 0.
[08/05 11:45:22 d2.utils.events]:  eta: 0:05:22  iter: 9099  total_loss: 0.4096  loss_cls: 0.09827  loss_box_reg: 0.1714  loss_mask: 0
[08/05 11:45:30 d2.utils.events]:  eta: 0:05:15  iter: 9119  total_loss: 0.4586  loss_cls: 0.1259  loss_box_reg: 0.1678  loss_mask: 0.
```

```
cfg.MODEL.WEIGHTS = "/content/drive/MyDrive/model_final.pth"
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5
predictor = DefaultPredictor(cfg)
```

```
import torch

# Assuming 'trainer' is your DefaultTrainer object and 'cfg' is your configuration object

# Specify the path where you want to save the model weights
output_dir = "./output"
output_weights_file = "/content/drive/MyDrive/model_final_detectron.pth"

# Save the model weights
torch.save(trainer.model.state_dict(), output_weights_file)

print(f"Model weights saved to {output_weights_file}")
```

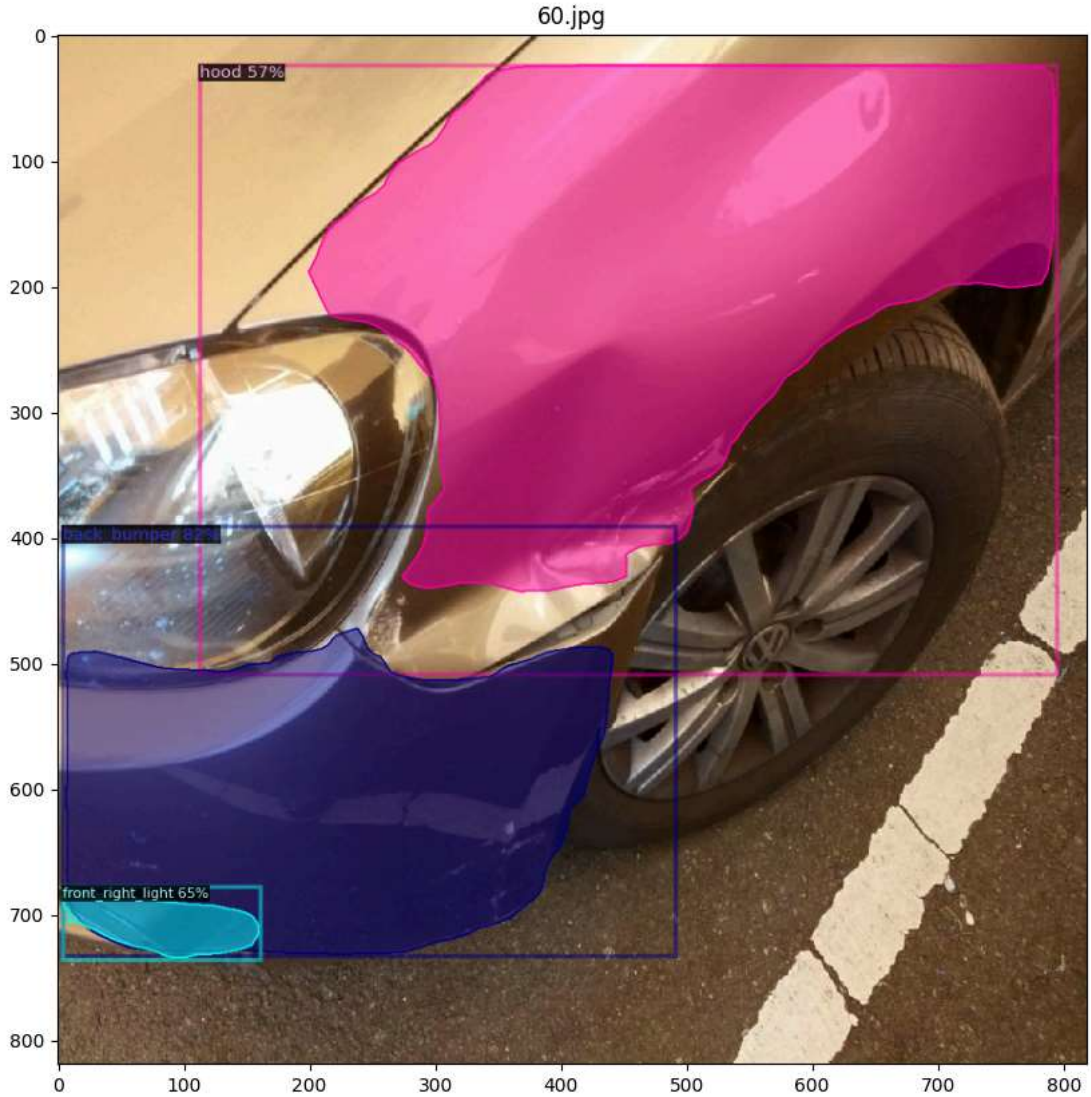
```
Model weights saved to /content/drive/MyDrive/model_final_detectron.pth
```

Start coding or generate with AI.

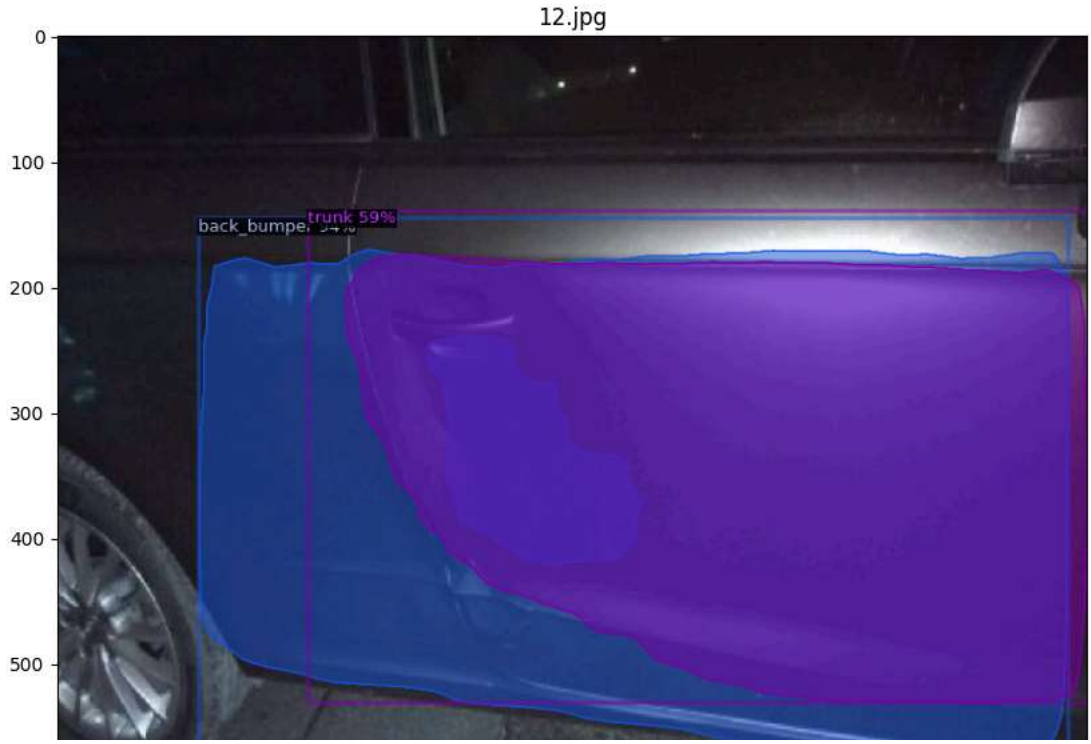
```
def show_predictions(image_path):
    im = cv2.imread(image_path)
    outputs = predictor(im)
    v = Visualizer(im[:, :, ::-1], metadata=car_parts_metadata, scale=0.8)
    v = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    plt.figure(figsize=(14, 10))
    plt.imshow(v.get_image()[:, :, ::-1])
    plt.show()
```

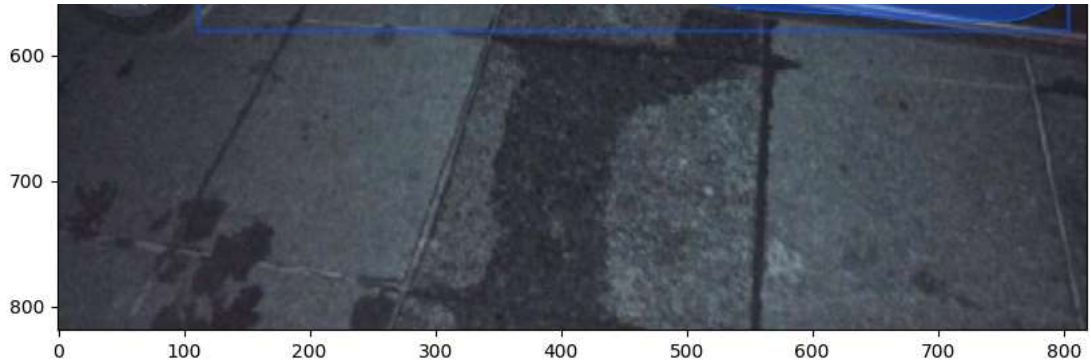
```
import os

def show_predictions2(image_path):
    # Check if the file exists
    if not os.path.isfile(image_path):
        print(f"Error: The file {image_path} does not exist.")
        return

    # Read the image
    im = cv2.imread(image_path)

    # Check if the image was successfully read
    if im is None:
        print(f"Error: Unable to read the image file {image_path}.")
        return

    # Make predictions
    outputs = predictor(im)

    # Visualize the predictions
    v = Visualizer(im[:, :, ::-1], metadata=car_parts_metadata, scale=0.8)
    v = v.draw_instance_predictions(outputs["instances"].to("cpu"))

    # Display the image
    plt.figure(figsize=(14, 10))
    plt.imshow(v.get_image()[:, :, ::-1])
    plt.show()

# Test the function with a correct image path
# Update this with the actual path to your test image



show_predictions_folder3("/content/drive/MyDrive/good/test")
```

```
Processing file: /content/drive/MyDrive/good/test/60.jpg
/usr/local/lib/python3.10/dist-packages/torch/functional.py:512: UserWarning: torch.meshgrid: in an upcoming release, it will be require
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
```

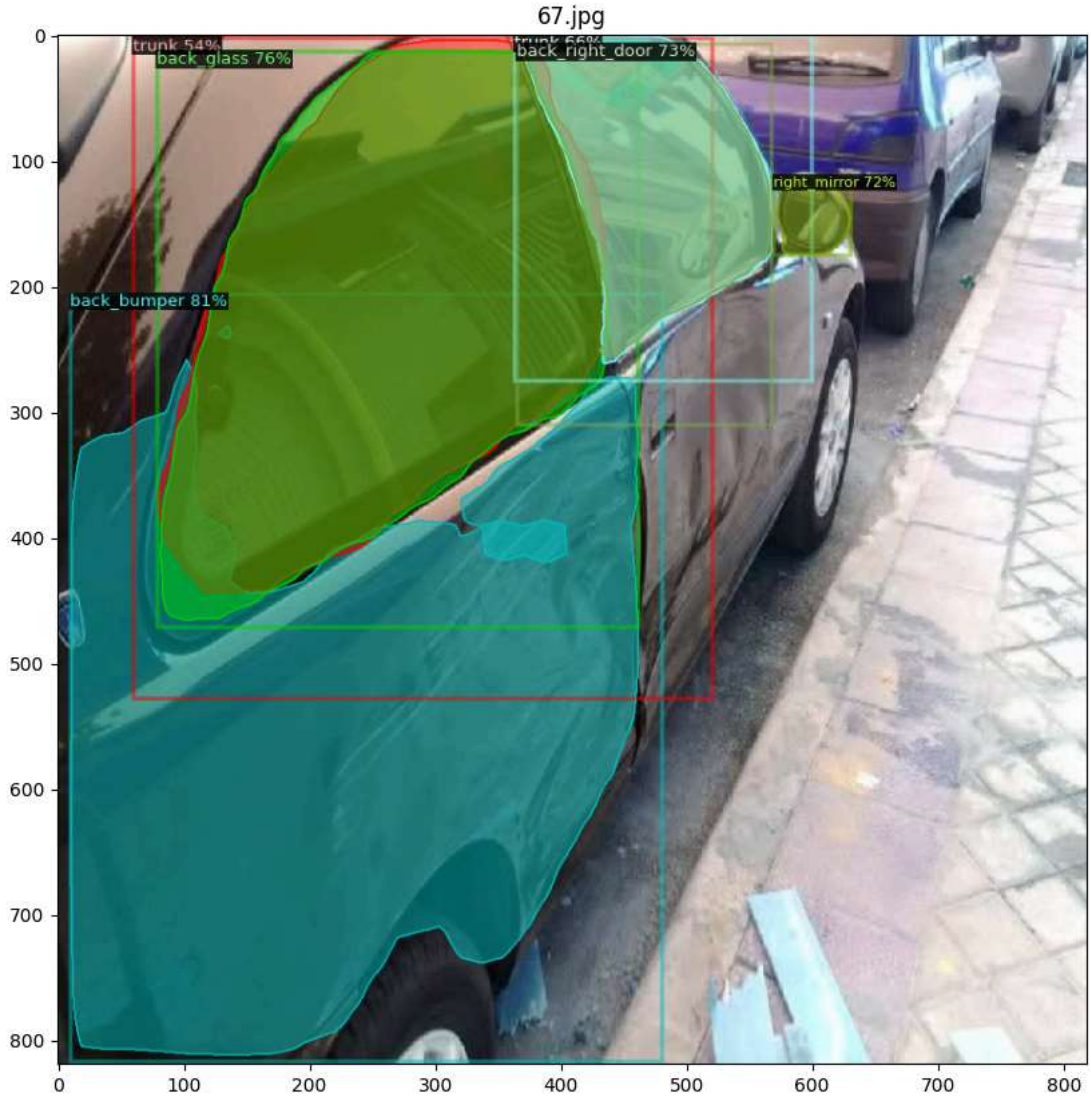



```
Processing file: /content/drive/MyDrive/good/test/12.jpg
```





```
Processing file: /content/drive/MyDrive/good/test/67.jpg
```




```
Processing file: /content/drive/MyDrive/good/test/45.jpg
```