UNIVERSITY OF MICHIGAN

Final Project Overview and Tips

NA 565, Fall 2023

Self-Driving Cars: Perception and Control

Joey Wilson

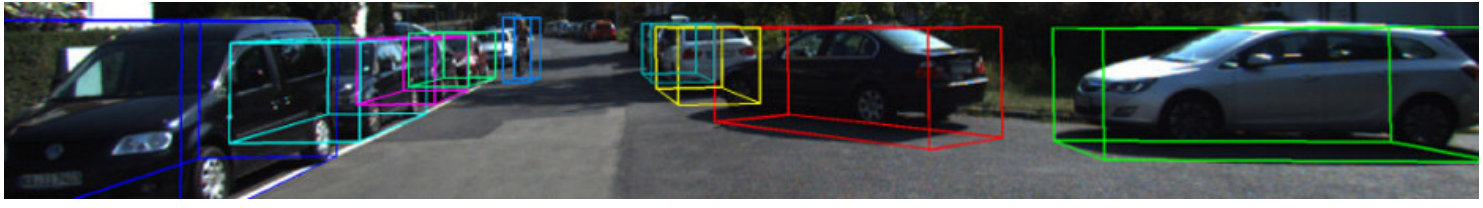
Monday, November 20, 2023

Content

- Data Overview
  - Files
  - Definitions
- Data Preparation
- Attendance
- Open-Source Networks
  - Recommendations

Task: Monocular 3D Object Detection

- Recent task proposed in 2016
- Input: RGB images from a single camera
- Output: 2D bounding boxes in pixels, and 3D bounding boxes in camera frame



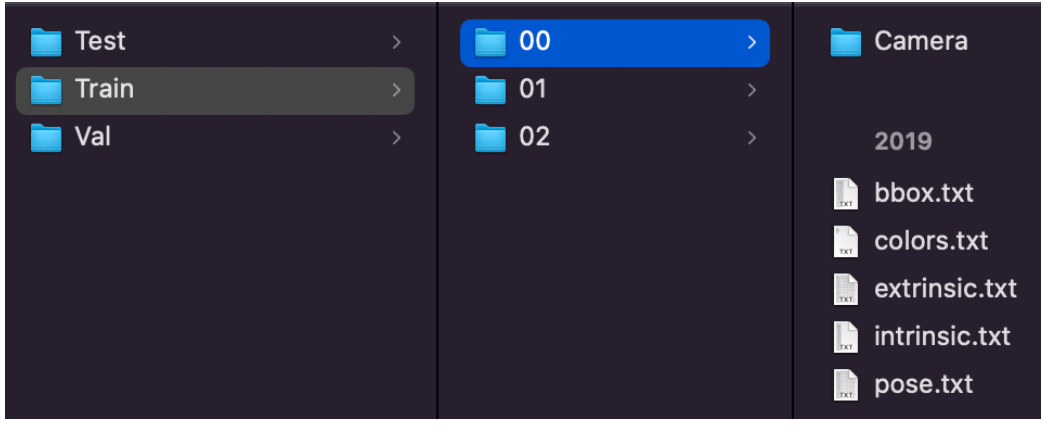
Chen, Xiaozhi, et al. "Monocular 3D Object Detection for Autonomous Driving." *CVPR*, 2016.

Dataset

- Simulated data containing clear weather and adverse weather
- Download at: https://curly-dataset-public.s3.us-east-2.amazonaws.com/NA_565/Final/FinalData.zip

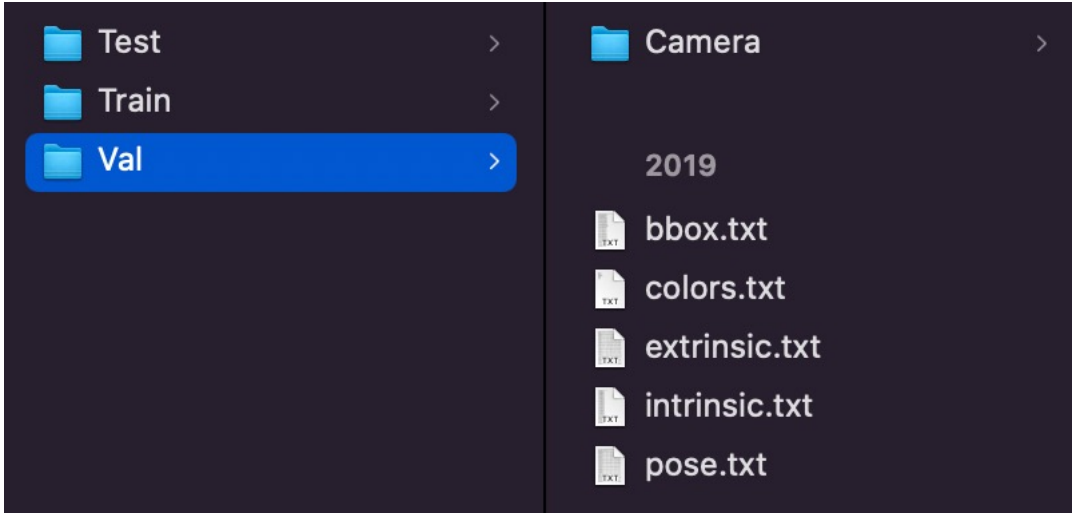
Raw Data

- Three folders: Train, Val, Test
  - Train folder contains three sequences
  - Each sequence contains data from consecutive frames



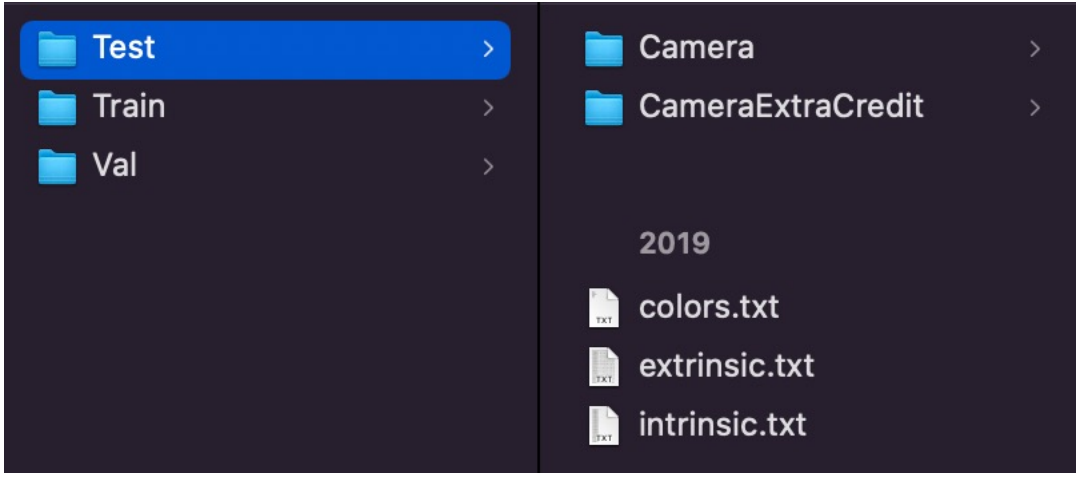
Raw Data: Val

- Three folders: Train, Val, Test
  - Val folder contains a single sequence for validation



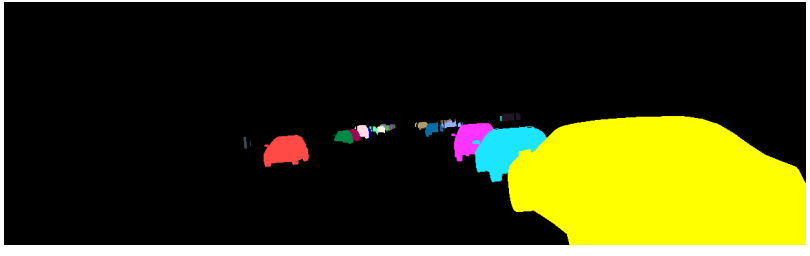
Raw Data: Test

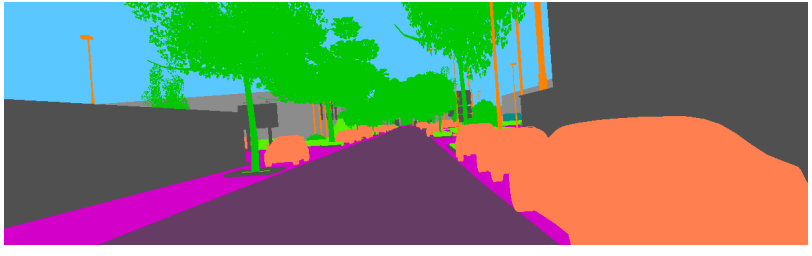
- Three folders: Train, Val, Test
  - Test folder contains two sequences (one regular, one extra credit)
  - The sequences in test are sequential from each-other, so only one set of .txt files is provided



Camera Folder

- Three images per frame
  - Raw camera (rgb_xxxxx.jpg)
  - Semantic segmentation (classgt_xxxxx.png) – only for training
  - Instance segmentation (instancegt_xxxxx.png) – only for training





2D Bounding Boxes (bbox.txt)

- Second column is camera ID, we only use camera 0
- trackID is temporally consistent instance number
- Fourth through seventh column are pixel 2D bounding boxes
- number_pixels is number of pixels of this instance in the image
- Truncation_ratio: 0 to 1 (truncated) where 1 refers to leaving image bounds
- Occupancy_ratio: float 0 (occluded) to 1 (visible) of object pixels

```
frame cameraID trackID left right top bottom number_pixels truncation_ratio occupancy_ratio isMoving
0 0 0 774 1241 169 374 75554 0.07726043 0.7891994 False
0 0 1 722 866 186 281 5433 0 0.3971491 False
0 0 2 696 771 185 246 2109 0 0.4609836 False
0 0 3 394 474 201 256 2534 0 0.5759091 False
0 0 4 509 542 195 220 459 0 0.5563636 False
0 0 5 651 682 184 208 357 0 0.4798387 False
0 0 6 522 552 195 216 219 0 0.3476191 False
0 0 7 539 566 189 212 293 0 0.4718196 False
```

Camera Intrinsic Matrix (intrinsic.txt)

- Only use camera 0
- Use the columns to construct intrinsic matrix
- Provides focal length and offset

$$\begin{pmatrix} x_I \\ y_I \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & \alpha & c_x{}' \\ 0 & f_y & c_y{}' \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix}$$

frame	cameraID	K[0,0]	K[1,1]	K[0,2]	K[1,2]
0	0	725.0087	725.0087	620.5	187
0	1	725.0087	725.0087	620.5	187
1	0	725.0087	725.0087	620.5	187
1	1	725.0087	725.0087	620.5	187
2	0	725.0087	725.0087	620.5	187
2	1	725.0087	725.0087	620.5	187

Camera Extrinsic Matrix (extrinsic.txt)

- Providing camera pose in a similar format to before
- Can optionally use pose to construct a trajectory, but this is not necessary

$$\begin{pmatrix} x_C \\ y_C \\ z_C \\ 1 \end{pmatrix} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix}$$

```
frame cameraID r1,1 r1,2 r1,3 t1 r2,1 r2,2 r2,3 t2 r3,1 r3,2 r3,3 t3 0 0 0 1
0 0 -0.265882 0.0721791 -0.9612997 7.760932 0.01772801 0.9973904 0.06998566 112.3845 0.9638427 0.001565997 -0.2664678 -0.9046764 0 0 0 1
0 1 -0.265882 0.0721791 -0.9612997 7.228207 0.01772801 0.9973904 0.06998566 112.3845 0.9638427 0.001565997 -0.2664678 -0.9046765 0 0 0 1
1 0 -0.2656336 0.0730738 -0.9613009 7.853128 0.01958271 0.9973266 0.07040109 112.3763 0.9638754 -0.000124 -0.2663544 -2.269279 0 0 0 1
1 1 -0.2656336 0.0730738 -0.9613009 7.320404 0.01958271 0.9973266 0.07040109 112.3763 0.9638754 -0.000124 -0.2663544 -2.269279 0 0 0 1
2 0 -0.2659233 0.06971319 -0.9614702 7.473315 0.01614201 0.9975639 0.06786568 112.4106 0.9638591 0.002526991 -0.2664008 -3.044499 0 0 0 1
2 1 -0.2659233 0.06971319 -0.9614702 6.940589 0.01614201 0.9975639 0.06786568 112.4105 0.9638591 0.002526991 -0.2664008 -3.0445 0 0 0 1
```

3D Bounding Boxes (pose.txt)

- Lots of parameters, the new relevant columns are:
  - Alpha: observation angle of objects in radians
  - Width, height, length of 3D bounding box in meters
  - Location in camera-space along X, Y, Z axes
  - Rotation_camera_space_y: yaw

```
frame cameraID trackID alpha width height length world_space_X world_space_Y world_space_Z rotation_world_space_y rotation_world_space_x
rotation_world_space_z camera_space_X camera_space_Y camera_space_Z rotation_camera_space_y rotation_camera_space_x rotation_camera_space_z
0 0 0 -1.996793 1.85 1.50992 4.930564 6.371316 -111 -5.044228 0.2694305 0 0 2.904048 1.4341 6.406569 -1.571205 0.001540535 -0.07224263
0 0 1 -1.796513 1.612032 1.404795 3.772344 12.85112 -111 -6.945244 0.2689006 0 0 3.008632 1.41593 13.15864 -1.571733 0.001502285 -0.07224342
0 0 2 -1.654057 1.567278 1.413269 3.158158 18.77596 -111 -8.52928 0.3388072 0 0 2.956056 1.410106 19.29135 -1.502008 0.006540446 -0.07196293
0 0 3 1.832166 1.555003 1.527328 3.576751 25.4738 -111 -1.092491 -2.856606 0 0 -5.973762 2.049313 23.76534 1.585904 -0.002663161 0.07221006
0 0 4 1.68899 1.540477 1.417371 3.504343 47.3462 -111 -7.073209 -2.88292 0 0 -6.039978 2.018503 46.44057 1.559658 -0.000763964 0.07225502
0 0 5 -1.634804 1.746308 1.512623 3.775172 48.3271 -111 -16.87533 0.2681896 0 0 3.121991 1.349885 49.99796 -1.572443 0.001450962 -0.07224449
```

Don’t Worry: Helper Scripts

- Dealing with the data can be difficult, so we provide helper scripts to convert to standard KITTI format
  - Download DevKit.zip from Canvas
  - ToKITTITrain.ipynb converts the train and val splits to KITTI format
  - Make sure to set the data_dir to where your data is stored

```
rng1 = np.random.default_rng()
data_dir = "/home/tigeriv/Data/NA565/Final/"
train_path = os.path.join(data_dir, "Train")
val_path = os.path.join(data_dir, "Val")
```

https://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d

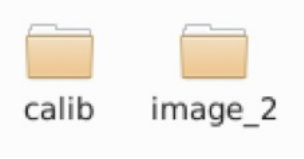
Train Helper Script

- Output will be stored in a training folder with calibration, image, and label sub-folders
  - Files are in standard KITTI, format
  - For more details check the readme.txt file in the DevKit

- Train and val are combined into a single folder, with image splits written as txt files in a new ImageSets folder

Test Helper Script

- Functions the same as the train helper script
- Creates a testing folder with only calibration and images
  - Both regular and extra credit are combined



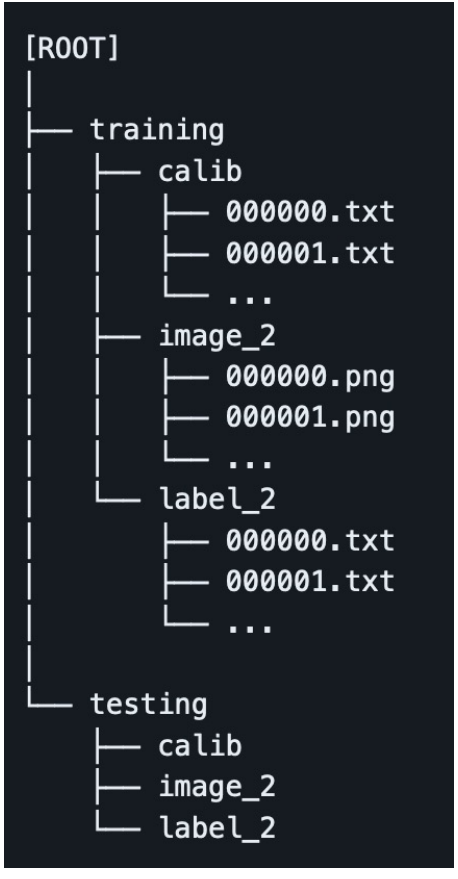
- Note: this is provided as a quick-start method. Feel fry to modify as you wish.

Open-Source Networks

- There are many open-source networks: https://github.com/BigTeacher-777/Awesome-Monocular-3D-detection
  - You do not need to create your own! Instead, try to train an open-source network on the data
- However, some may be difficult to run due to dependencies or file formats
- We found the following repositories much easier to use:
  - https://github.com/2gunsu/monocon-pytorch
  - https://github.com/zhangcheng828/MonoDetector

Training a model

- First, prepare data in expected format
  - Our notebooks do this for the repositories we list
- Clone the repository of your choice
  - For this demo, we will use monocon-pytorch
- Set up development environment
  - We will use docker for repeatability
  - nvidia/cuda:11.3.1-cudnn8-devel-ubuntu20.04




```
tigeriv@vino:/media/sde1/Joey/FA565$ docker run -it --gpus=all --shm-size 16G -v /media/sde1/Joey:/workspace/Joey 890f6fb0b159 bash
```

Container setup

- Follow the instructions in the GitHub repository

```
(base) root@fcefdc1910da:/workspace/Joey/FA565/monocon-pytorch# conda create --name DEMO python=3.8
```

```
root@fcefdc1910da:/workspace/Joey/FA565/monocon-pytorch# conda activate DEMO
(DEMO) root@fcefdc1910da:/workspace/Joey/FA565/monocon-pytorch# pip install -r requirements.txt
```

- Error we encountered:

```
ERROR: Could not find a version that satisfies the requirement numpy==1.22.4 (from versions: 1.3.0, 1.4.1, 1.5.0, 1.5.1, 1.6
.0, 1.6.1, 1.6.2, 1.7.0, 1.7.1, 1.7.2, 1.8.0, 1.8.1, 1.8.2, 1.9.0, 1.9.1, 1.9.2, 1.9.3, 1.10.0.post2, 1.10.1, 1.10.2, 1.10.4
, 1.11.0, 1.11.1, 1.11.2, 1.11.3, 1.12.0, 1.12.1, 1.13.0, 1.13.1, 1.13.3, 1.14.0, 1.14.1, 1.14.2, 1.14.3, 1.14.4, 1.14.5, 1.
14.6, 1.15.0, 1.15.1, 1.15.2, 1.15.3, 1.15.4, 1.16.0, 1.16.1, 1.16.2, 1.16.3, 1.16.4, 1.16.5, 1.16.6, 1.17.0, 1.17.1, 1.17.2
, 1.17.3, 1.17.4, 1.17.5, 1.18.0, 1.18.1, 1.18.2, 1.18.3, 1.18.4, 1.18.5, 1.19.0, 1.19.1, 1.19.2, 1.19.3, 1.19.4, 1.19.5, 1.
20.0, 1.20.1, 1.20.2, 1.20.3, 1.21.0, 1.21.1, 1.21.2, 1.21.3, 1.21.4, 1.21.5, 1.21.6)
ERROR: No matching distribution found for numpy==1.22.4
```

  - Solution: Just remove the requirement in requirements.txt for a specific numpy version
  - You will probably encounter lots of small issues. If you do, try looking online as others have probably encountered the same problem.

Container setup

- Depending on your docker setup, you may already have cuda toolkit installed
  - Check with

```
(DEMO) root@fcefdc1910da:/workspace/Joey/FA565/monocon-pytorch# nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Mon_May__3_19:15:13_PDT_2021
Cuda compilation tools, release 11.3, V11.3.109
Build cuda_11.3.r11.3/compiler.29920130_0
```

- Install Torch

```
(DEMO) root@fcefdc1910da:/workspace/Joey/FA565/monocon-pytorch# pip install torch --index-url https://download.pytorch.org/whl/cu113
```

- Edit config file to point to location of your data

```
# Data
_C.DATA = CN()
_C.DATA.ROOT = r'/workspace/Joey/FinalData'
```

Possible Error

- You may encounter this error if using docker

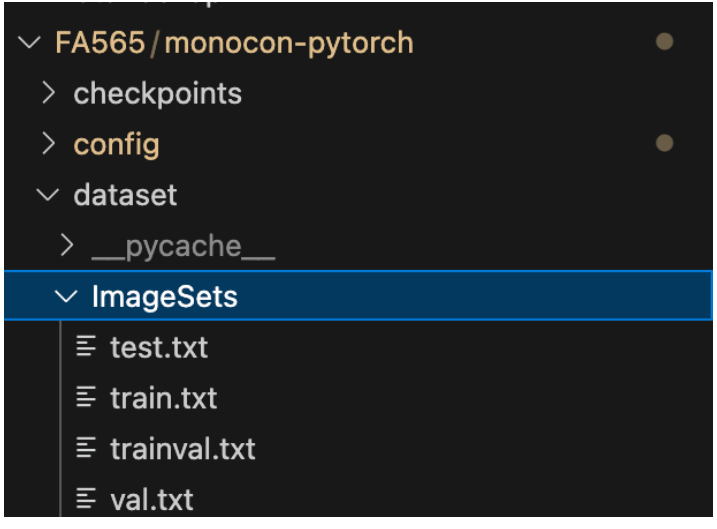
```
  File "/home/miniconda3/envs/DEMO/lib/python3.8/importlib/__init__.py", line 127, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
ImportError: libGL.so.1: cannot open shared object file: No such file or directory
```

- Solution comes from the internet by looking on stack overflow

```
apt-get update && apt-get install libgl1
apt install libgl1-mesa-glx
apt-get install -y libglib2.0-0 libsm6 libxrender1 libxext6
```

Set the splits

- In the pre-processing script we generate an ImageSets folder
  - These splits determine which files the data-loader will look for, and may be located within the dataset folder of the repository
  - Make sure you update the files, or your network will not run



Set the number of objects

- By default, many networks will predict up to 30 bounding boxes
  - However, our dataset contains more than 30
  - Increase the maximum number of objects or your network will crash

```
_C.MODEL.HEAD = CN()
_C.MODEL.HEAD.NUM_CLASSES = 3
_C.MODEL.HEAD.MAX_OBJS = 50
```

```
IndexError: Caught IndexError in DataLoader worker process 0.
Original Traceback (most recent call last):
  File "/home/miniconda3/envs/DEMO/lib/python3.8/site-packages/torch/utils/data/_utils/worker.py", line 302, in _worker_loop
    data = fetcher.fetch(index)
  File "/home/miniconda3/envs/DEMO/lib/python3.8/site-packages/torch/utils/data/_utils/fetch.py", line 49, in fetch
    data = [self.dataset[idx] for idx in possibly_batched_index]
  File "/home/miniconda3/envs/DEMO/lib/python3.8/site-packages/torch/utils/data/_utils/fetch.py", line 49, in <listcomp>
    data = [self.dataset[idx] for idx in possibly_batched_index]
  File "/workspace/Joey/FA565/monocon-pytorch/dataset/monocon_dataset.py", line 137, in __getitem__
    new_labels['gt_bboxes'][obj_idx, :] = gt_bbox
IndexError: index 36 is out of bounds for axis 0 with size 30
```

Train!

- Train your network following the commands in the repository
- Train once with default parameters, then try changing the optimizer, scheduler, or learning rate
  - Cyclic can be effective but requires lots of tuning. Exponential may be easier when starting.

```
Model: MonoConDetector (# Params: 19620261)
Optimizer: AdamW
Scheduler: CyclicScheduler
```

Another Possible Error

- If using MonoDetector, ImageNet weights are assumed to be pre-downloaded
  - These are also available online from the PyTorch model zoo

```
import torch.utils.model_zoo as model_zoo
```

```
def load_imagenet_weights(self, num_layers: int):

    NUM_LAYERS_TO_HASH = {
        34: ('dla34', 'ba72cf86'),
        46: ('dla46_c', '2bfd52c3'),
        60: ('dla60', '24839fc4'),
        102: ('dla102', 'd94d9790')}

    arch_name, hash = NUM_LAYERS_TO_HASH[num_layers]

    base_url = 'http://dl.yf.io/dla/models/imagenet'
    url = os.path.join(base_url, f'{arch_name}-{hash}.pth')

    state_dict = model_zoo.load_url(url)
    self.load_state_dict(state_dict, strict=False)
```

Inference

- We provide a helper script for monocon inference since the original repository does not provide an inference script
  - https://github.com/minghanz/monocon_na565
  - https://github.com/minghanz/monocon_infer_eval/blob/main/test.py
  - https://github.com/minghanz/monocon_infer_eval/blob/main/engine/monocon_engine.py
  - https://github.com/minghanz/monocon_infer_eval/blob/main/utils/kitti_convert_utils.py
  - You will still need to implement test.py and monocon_engine.py

Inference

- The DevKit includes an example prediction for submission to the AutoGrader
  - Due to file restrictions, we combine all prediction files (standard KITTI format) into a single prediction file
  - See the merger.py file provided in the development kit for help merging

```
0 Car 0.07726043 0 -1.996793 774 169 1241 374 1.50992 1.85 4.930564 2.904048 1.4341 6.406569 -1.571205 0.2731365143101895
0 Car 0 1 -1.796513 722 186 866 281 1.404795 1.612032 3.772344 3.008632 1.41593 13.15864 -1.571733 0.08857553951979591
0 Car 0 0 -1.654057 696 185 771 246 1.413269 1.567278 3.158158 2.956056 1.410106 19.29135 -1.502008 0.666401613669677
0 Car 0 0 1.832166 394 201 474 256 1.527328 1.555003 3.576751 -5.973762 2.049313 23.76534 1.585904 0.13359012364364364
0 Car 0 0 1.68899 509 195 542 220 1.417371 1.540477 3.504343 -6.039978 2.018503 46.44057 1.559658 0.674445017565153
0 Car 0 0 -1.634804 651 184 682 208 1.512623 1.746308 3.775172 3.121991 1.349885 49.99796 -1.572443 0.46835076942539555
0 Car 0 1 1.696156 522 195 552 216 1.360295 1.513462 4.017023 -5.932473 2.001972 52.01393 1.582591 0.8113869651662797
0 Car 0 0 1.664531 539 189 566 212 1.73687 1.652467 4.107703 -5.618606 1.966214 60.322 1.571656 0.2572656496629393
0 Car 0 1 1.670515 547 190 570 209 1.609242 1.638188 3.998 -5.678875 1.960166 66.95219 1.585898 0.6696787188516616
0 Car 0 1 1.641113 559 191 579 206 1.431419 1.697305 3.744055 -5.483585 1.928971 77.81883 1.570763 0.2930062230433441
0 Car 0 0 -1.612065 642 184 657 196 1.546081 1.6639 4.146974 3.962967 1.211496 99.37627 -1.572208 0.7571100171236819
0 Car 0 1 -3.237724 676 180 713 194 1.473782 1.69797 4.648869 9.391953 0.8298233 92.23568 -3.136248 0.20433120500690782
0 Car 0 2 -3.232058 677 177 710 194 1.909038 1.812069 4.330702 9.579693 0.8113818 95.32779 -3.131902 0.13943108768646084
0 Car 0 1 -1.524243 581 188 594 200 1.632486 1.608983 3.812409 -4.72489 1.833402 103.7177 -1.569767 0.12334575620647492
0 Car 0 0 1.614736 576 190 591 202 1.430527 1.566407 3.482533 -4.747828 1.852427 92.65812 1.56354 0.21023306721895396
0 Car 0 2 3.04617 674 180 705 193 1.630828 1.672862 4.318664 9.572704 0.8021712 101.5161 3.14019 0.8656247970998684
0 Car 0 1 1.599632 585 188 597 199 1.530483 1.680223 3.862273 -4.687437 1.812963 115.0087 1.558898 0.61248442562007
0 Car 0 2 1.622681 588 189 599 198 1.420148 1.552787 3.85067 -4.554659 1.794708 120.5156 1.584906 0.116501310224846
0 Car 0 2 1.591782 589 188 599 197 1.427379 1.478135 3.709917 -4.534988 1.785635 125.3879 1.55563 0.9991472676236095
0 Car 0 1 1.594176 590 188 601 197 1.400546 1.647438 4.024915 -4.551241 1.777405 131.379 1.559547 0.059417722174979315
0 Car 0 2 3.056136 669 181 692 191 1.639204 1.585631 3.989951 10.46329 0.6974428 127.1695 3.13823 0.07077110635811679
0 Car 0 2 3.033982 671 180 687 191 1.719717 1.682953 2.781201 10.48017 0.6921382 129.7702 3.114567 0.2535798373682666
0 Car 0 2 3.039215 667 181 686 191 1.674444 1.672834 3.425593 10.09192 0.7166066 132.0816 3.115474 0.3267070476220283
0 Car 0 2 -0.2624233 765 172 797 186 1.736666 1.796667 4.374446 23.82088 -0.2388959 107.8571 -0.04505685 0.6812836185531521
```

After Your First Submission

- Hopefully the first submission is straightforward. You need to:
  - Set up an environment with a GPU
  - Run the software and generate predictions
  - Submit predictions
- After, try to improve your score:
  - Try a different model
  - Change the parameters in the config file
  - https://github.com/minghanz/monocon_infer_eval/blob/main/config/monocon_configs.py

Happy Thanksgiving!

- Auto-grader will be up within next couple of days
- Questions?