# 2.6.5 Cityscapes数据集、图像分割与城市空间对象统计

### 2.6.5.1 [Cityscapes数据集](https://www.cityscapes-dataset.com/)①

Cityscapes数据集集中于城市街道场景的语义解释（semantic understanding），这非常适用于对城市空间内容的分析。这个数据集的主要特点有[1]：

**1. Polygonal annotations 多边形标注**

* Dense semantic segmentation 密集的语义分割
* Instance segmentation for vehicle and people 车辆和行人的实例（对象）分割

**2. Complexity 复杂性**

* 30 classes 30个分类
* Class Definitions for a list of all classes 分类定义如下：

| Group | Classes |
| --- | --- |
| flat | road · sidewalk · parking+ · rail track+ |
| human | person*· rider* |
| vehicle | car*· truck* · bus*· on rails* · motorcycle*· bicycle* · caravan*+ · trailer*+ |
| construction | building · wall · fence · guard rail+ · bridge+ · tunnel+ |
| object | pole · pole group+ · traffic sign · traffic light |
| nature | vegetation · terrain |
| sky | sky |
| void | ground+ · dynamic+ · static+ |

\* 包含单个实例对象注释。但是，如果多个对象（实例）之间界限不清楚，整个组（crowd/group），包含多个对象被标记在一起，并标注为组，例如车辆组。

+ 该标签不包括在任何评估中，并视为无效。（这是关于模型评估的说明）

**3. Diversity 多样性**

* 50 cities 包含有50座城市
* Several months (spring, summer, fall) 春夏秋等多季节变化
* Daytime 白天
* Good/medium weather conditions 好/中等天气状况
* Manually selected frames 手动选择的帧
* Large number of dynamic objects 大量的动态对象
* Varying scene layout 多样的场景布局
* Varying background 多样的背景

**4. Volume 量**

* 5,000 annotated images with fine annotations 5,000张精标注
* 20,000 annotated images with coarse annotations 20,000张粗标注

**5. metadata 元数据**

* Preceding and trailing video frames. Each annotated image is the 20th image from a 30 frame video snippets (1.8s) 前后视频帧。每一幅注释图像都是30帧视频片段（1.8s）中的第20幅图像。
* Corresponding right stereo views 对应右侧立体视图
* GPS coordinates GPS坐标
* Ego-motion data from vehicle odometry 汽车里程测量的运动数据
* Outside temperature from vehicle sensor 来自车辆传感器的外部温度

**6. Extensions by other researchers 其他研究者的扩展**

* Bounding box annotations of people 行人锚框标注
* Images augmented with fog and rain 雨雾图像增广

**7. Benchmark suite and evaluation server 基准测试套件和评估服务器**

* Pixel-level semantic labeling 像素级语义标签
* Instance-level semantic labeling 实例（对象）级语义标签
* Panoptic semantic labeling 展示全景的语义标签

图像分割训练的数据集在[Cityscapes下载页](https://www.cityscapes-dataset.com/login/)③ 注册后获取。主要包括两个文件，一个是，11GB大小的leftImg8bit\_trainvaltest.zip，为训练、验证和测试的图像；二是，241ＭＢ大小的gtFine\_trainvaltest.zip，为训练和验证图像数据集对应的精细标注。

迁移 [Hierarchical Multi-Scale Attention for Semantic Segmentation](https://github.com/NVIDIA/semantic-segmentation)[2]代码的前半部分，读取Cityscapes数据集，理解该数据集的文件结构和数据结构。

#### 1）参数管理

应用（App）涉及到的参数很多，为了便于管理，使用[argparse](https://docs.python.org/3/library/argparse.html)④命令行参数解析包，可以将参数和代码分离开来，方便读取命令行参数，尤其适合于参数的频繁修改。同时，在程序执行过程中，为避免参数变化导致难以调试或难以理解代码，在参数配置完之后，需要将mutable参数转变为immutable参数，迁移代码类class AttrDict(dict)实现。AttrDict类可以定义类的属性，并通过该类的immutable方法，实现类属性的批量类型转换（mutable到immutable，或反之）。

# semantic-segmentation-main\train.py  
import argparse #  
# Argument Parser  
parser = argparse.ArgumentParser(description='Semantic Segmentation')  
  
#...  
parser.add\_argument('--start\_epoch', type=int, default=0)  
parser.add\_argument('--max\_epoch', type=int, default=180)  
parser.add\_argument('--global\_rank', default=0, type=int,help='parameter used by apex library')  
parser.add\_argument('--test\_mode', action='store\_true', default=False,help=('Minimum testing to verify nothing failed, ''Runs code for 1 epoch of train and val'))  
parser.add\_argument('--init\_decoder', default=False, action='store\_true',help='initialize decoder with kaiming normal')  
parser.add\_argument('--syncbn', action='store\_true', default=False,help='Use Synchronized BN')  
parser.add\_argument('--extra\_scales', type=str, default='0.5,2.0')  
parser.add\_argument('--set\_cityscapes\_root', type=str, default=None,help='override cityscapes default root dir')  
parser.add\_argument('--dataset', type=str, default='cityscapes',help='cityscapes, mapillary, camvid, kitti')  
parser.add\_argument('--result\_dir', type=str, default='./logs\_sementic',help='where to write log output')  
parser.add\_argument('--cv', type=int, default=0,help=('Cross-validation split id to use. Default # of splits set' ' to 3 in config'))  
  
parser.add\_argument('--crop\_size', type=str, default='448',help='dynamically scale training images down to this size') #896  
parser.add\_argument('--scale\_min', type=float, default=0.5, help='dynamically scale training images down to this size')  
parser.add\_argument('--scale\_max', type=float, default=2.0,help='dynamically scale training images up to this size')  
parser.add\_argument('--full\_crop\_training', action='store\_true', default=False,help='Full Crop Training')  
parser.add\_argument('--pre\_size', type=int, default=None,help=('resize long edge of images to this before'' augmentation'))  
parser.add\_argument('--color\_aug', type=float,default=0.25, help='level of color augmentation')  
parser.add\_argument('--jointwtborder', action='store\_true', default=False,help='Enable boundary label relaxation')  
parser.add\_argument('--dump\_augmentation\_images', action='store\_true', default=False,help='Dump Augmentated Images for sanity check')  
  
parser.add\_argument('--rmi\_loss', action='store\_true', default=False,help='use RMI loss')  
parser.add\_argument('--img\_wt\_loss', action='store\_true', default=False,help='per-image class-weighted loss')  
parser.add\_argument('--arch', type=str, default='deepv3.DeepV3Plus',help='Network architecture. We have DeepSRNX50V3PlusD (backbone: ResNeXt50)and deepWV3Plus (backbone: WideResNet38).')  
parser.add\_argument('--trunk', type=str, default='resnet101',help='trunk model, can be: resnet101 (default), resnet50')  
parser.add\_argument('--apex', action='store\_true', default=False,help='Use Nvidia Apex Distributed Data Parallel')  
parser.add\_argument('--optimizer', type=str, default='sgd', help='optimizer')  
parser.add\_argument('--lr', type=float, default=0.002)  
parser.add\_argument('--weight\_decay', type=float, default=1e-4)  
parser.add\_argument('--momentum', type=float, default=0.9)  
parser.add\_argument('--lr\_schedule', type=str, default='poly',help='name of lr schedule: poly')  
parser.add\_argument('--poly\_exp', type=float, default=1.0,help='polynomial LR exponent')  
  
#...  
  
args = parser.parse\_args([]) # JupyterLab需要在args = parser.parse\_args()中传入空的[]，否则引发异常  
print("--syncbn=%f"%args.syncbn)  
  
args.world\_size = 1  
print("增加新的参数-world\_size=%d"%args.world\_size)  
print("args参数解析:",args)

--syncbn=0.000000  
增加新的参数-world\_size=1  
args参数解析: Namespace(apex=False, arch='deepv3.DeepV3Plus', color\_aug=0.25, crop\_size='448', cv=0, dataset='cityscapes', dump\_augmentation\_images=False, extra\_scales='0.5,2.0', full\_crop\_training=False, global\_rank=0, img\_wt\_loss=False, init\_decoder=False, jointwtborder=False, lr=0.002, lr\_schedule='poly', max\_epoch=180, momentum=0.9, optimizer='sgd', poly\_exp=1.0, pre\_size=None, result\_dir='./logs\_sementic', rmi\_loss=False, scale\_max=2.0, scale\_min=0.5, set\_cityscapes\_root=None, start\_epoch=0, syncbn=False, test\_mode=False, trunk='resnet101', weight\_decay=0.0001, world\_size=1)

# semantic-segmentation-main\utils\attr\_dict.py  
"""  
# Code adapted from:  
# https://github.com/facebookresearch/Detectron/blob/master/detectron/utils/collections.py  
  
Source License  
# Copyright (c) 2017-present, Facebook, Inc.  
#  
# Licensed under the Apache License, Version 2.0 (the "License");  
# you may not use this file except in compliance with the License.  
# You may obtain a copy of the License at  
#  
# http://www.apache.org/licenses/LICENSE-2.0  
#  
# Unless required by applicable law or agreed to in writing, software  
# distributed under the License is distributed on an "AS IS" BASIS,  
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
# See the License for the specific language governing permissions and  
# limitations under the License.  
##############################################################################  
#  
# Based on:  
# --------------------------------------------------------  
# Fast R-CNN  
# Copyright (c) 2015 Microsoft  
# Licensed under The MIT License [see LICENSE for details]  
# Written by Ross Girshick  
# --------------------------------------------------------  
"""  
  
class AttrDict(dict):  
  
 IMMUTABLE = '\_\_immutable\_\_'  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super(AttrDict, self).\_\_init\_\_(\*args, \*\*kwargs)  
 self.\_\_dict\_\_[AttrDict.IMMUTABLE] = False  
  
 def \_\_getattr\_\_(self, name):  
 if name in self.\_\_dict\_\_:  
 return self.\_\_dict\_\_[name]  
 elif name in self:  
 return self[name]  
 else:  
 raise AttributeError(name)  
  
 def \_\_setattr\_\_(self, name, value):  
 if not self.\_\_dict\_\_[AttrDict.IMMUTABLE]:  
 if name in self.\_\_dict\_\_:  
 self.\_\_dict\_\_[name] = value  
 else:  
 self[name] = value  
 else:  
 raise AttributeError(  
 'Attempted to set "{}" to "{}", but AttrDict is immutable'.  
 format(name, value)  
 )  
  
 def immutable(self, is\_immutable):  
 """Set immutability to is\_immutable and recursively apply the setting  
 to all nested AttrDicts.  
 """  
 self.\_\_dict\_\_[AttrDict.IMMUTABLE] = is\_immutable  
 # Recursively set immutable state  
 for v in self.\_\_dict\_\_.values():  
 if isinstance(v, AttrDict):  
 v.immutable(is\_immutable)  
 for v in self.values():  
 if isinstance(v, AttrDict):  
 v.immutable(is\_immutable)  
  
 def is\_immutable(self):  
 return self.\_\_dict\_\_[AttrDict.IMMUTABLE]

实例化类AttrDict，使用类的属性存储参数，并配置参数immutable与mutable互相转换的方法。

# semantic-segmentation-main\config.py  
import os  
  
\_\_C = AttrDict() # 非嵌套字典  
cfg=\_\_C  
print("cfg=",cfg)  
  
# ...  
\_\_C.GLOBAL\_RANK = 0  
\_\_C.EPOCH = 0  
  
# Absolute path to a location to keep some large files, not in this dir.  
\_\_C.ASSETS\_PATH = r'G:\data\Cityscapes\_assets'  
  
print("参数——非嵌套字典:",cfg)  
  
# Attribute Dictionary for Options  
\_\_C.OPTIONS = AttrDict() # 嵌套字典  
\_\_C.OPTIONS.TEST\_MODE = False  
\_\_C.OPTIONS.INIT\_DECODER = False  
  
# Attribute Dictionary for Model  
\_\_C.MODEL = AttrDict()  
\_\_C.MODEL.EXTRA\_SCALES = '0.5,1.5'  
\_\_C.MODEL.BNFUNC = None  
  
WEIGHTS\_PATH = os.path.join(\_\_C.ASSETS\_PATH, 'seg\_weights')  
\_\_C.MODEL.WRN38\_CHECKPOINT =os.path.join(WEIGHTS\_PATH, 'wider\_resnet38.pth.tar')  
  
# Attribute Dictionary for Dataset  
\_\_C.DATASET = AttrDict()  
\_\_C.DATASET.CITYSCAPES\_DIR =os.path.join(\_\_C.ASSETS\_PATH, 'data/Cityscapes')  
\_\_C.DATASET.IGNORE\_LABEL = 255  
\_\_C.DATASET.NUM\_CLASSES = 0  
\_\_C.DATASET.CV = 1 # cv\_split - 0,1,2,3  
\_\_C.DATASET.CUSTOM\_COARSE\_PROB = None  
\_\_C.DATASET.CLASS\_UNIFORM\_PCT = 0.5  
\_\_C.DATASET.NAME = ''  
\_\_C.DATASET.CLASS\_UNIFORM\_TILE = 1024  
\_\_C.DATASET.CENTROID\_ROOT =os.path.join(\_\_C.ASSETS\_PATH, 'uniform\_centroids')  
\_\_C.DATASET.CITYSCAPES\_CUSTOMCOARSE=os.path.join(\_\_C.ASSETS\_PATH, 'data/Cityscapes/autolabelled')  
  
\_\_C.DATASET.MEAN = [0.485, 0.456, 0.406]  
\_\_C.DATASET.STD = [0.229, 0.224, 0.225]  
\_\_C.DATASET.DUMP\_IMAGES = False  
  
# This enables there to always be translation augmentation during random crop  
\_\_C.DATASET.TRANSLATE\_AUG\_FIX = False  
# ...  
  
print("参数——含嵌套字典:",cfg)

cfg= {}  
参数——非嵌套字典: {'GLOBAL\_RANK': 0, 'EPOCH': 0, 'ASSETS\_PATH': 'G:\\data\\Cityscapes\_assets'}  
参数——含嵌套字典: {'GLOBAL\_RANK': 0, 'EPOCH': 0, 'ASSETS\_PATH': 'G:\\data\\Cityscapes\_assets', 'OPTIONS': {'TEST\_MODE': False, 'INIT\_DECODER': False}, 'MODEL': {'EXTRA\_SCALES': '0.5,1.5', 'BNFUNC': None, 'WRN38\_CHECKPOINT': 'G:\\data\\Cityscapes\_assets\\seg\_weights\\wider\_resnet38.pth.tar'}, 'DATASET': {'CITYSCAPES\_DIR': 'G:\\data\\Cityscapes\_assets\\data/Cityscapes', 'IGNORE\_LABEL': 255, 'NUM\_CLASSES': 0, 'CV': 1, 'CUSTOM\_COARSE\_PROB': None, 'CLASS\_UNIFORM\_PCT': 0.5, 'NAME': '', 'CLASS\_UNIFORM\_TILE': 1024, 'CENTROID\_ROOT': 'G:\\data\\Cityscapes\_assets\\uniform\_centroids', 'CITYSCAPES\_CUSTOMCOARSE': 'G:\\data\\Cityscapes\_assets\\data/Cityscapes/autolabelled', 'MEAN': [0.485, 0.456, 0.406], 'STD': [0.229, 0.224, 0.225], 'DUMP\_IMAGES': False, 'TRANSLATE\_AUG\_FIX': False}}

assert\_and\_infer\_cfg函数可以将argparse定义的参数args，有选择性的存储到类AttrDict的实例cfg(\_\_C)中，并配置为immutable类型。

通过AttrDict的immutable方法可以转换参数的mutable和immutable的类型，从而修改参数。

cfg.immutable(False)  
cfg.MODEL.EXTRA\_SCALES='0.9,1.9'  
print("mutable——updated\_cfg.MODEL.EXTRA\_SCALES=",cfg.MODEL.EXTRA\_SCALES)  
cfg.immutable(True)

mutable——updated\_cfg.MODEL.EXTRA\_SCALES= 0.9,1.9

#### 2）cityscapes数据读取

* cityscapes的标签数据处理

Cityscapes数据集的处理工具，可以从[cityscapesScripts](https://github.com/mcordts/cityscapesScripts)⑤中查找。下述代码namedtuple对象labels，创建了不同类别之间（列之间）的映射关系，并通过name2label,id2label,trainId2label,label2trainid,trainId2name,trainId2color和category2labels定义了不同列转换的映射关系。方便后续cityscapes标签的变换。

# semantic-segmentation-main\datasets\cityscapes\_labels.py  
"""  
# File taken from https://github.com/mcordts/cityscapesScripts/  
# License File Available at:  
# https://github.com/mcordts/cityscapesScripts/blob/master/license.txt  
  
# ----------------------  
# The Cityscapes Dataset  
# ----------------------  
#  
#  
# License agreement  
# -----------------  
#  
# This dataset is made freely available to academic and non-academic entities for non-commercial purposes such as academic research, teaching, scientific publications, or personal experimentation. Permission is granted to use the data given that you agree:  
#  
# 1. That the dataset comes "AS IS", without express or implied warranty. Although every effort has been made to ensure accuracy, we (Daimler AG, MPI Informatics, TU Darmstadt) do not accept any responsibility for errors or omissions.  
# 2. That you include a reference to the Cityscapes Dataset in any work that makes use of the dataset. For research papers, cite our preferred publication as listed on our website; for other media cite our preferred publication as listed on our website or link to the Cityscapes website.  
# 3. That you do not distribute this dataset or modified versions. It is permissible to distribute derivative works in as far as they are abstract representations of this dataset (such as models trained on it or additional annotations that do not directly include any of our data) and do not allow to recover the dataset or something similar in character.  
# 4. That you may not use the dataset or any derivative work for commercial purposes as, for example, licensing or selling the data, or using the data with a purpose to procure a commercial gain.  
# 5. That all rights not expressly granted to you are reserved by us (Daimler AG, MPI Informatics, TU Darmstadt).  
#  
#  
# Contact  
# -------  
#  
# Marius Cordts, Mohamed Omran  
# www.cityscapes-dataset.net  
  
"""  
from collections import namedtuple  
#--------------------------------------------------------------------------------  
# Definitions  
#--------------------------------------------------------------------------------  
  
# a label and all meta information  
Label = namedtuple( 'Label' , [  
  
 'name' , # The identifier of this label, e.g. 'car', 'person', ... . 标签的标识符，例如'car', 'person'等  
 # We use them to uniquely name a class 使用'name'命名唯一的类  
  
 'id' , # An integer ID that is associated with this label. 与标签关联的整形ID  
 # The IDs are used to represent the label in ground truth images ID被用于表示真实图像的标签  
 # An ID of -1 means that this label does not have an ID and thus is ignored when creating ground truth images (e.g. license plate). 为-1值的ID意为这个标签没有ID（被忽略），例如车牌（涉及到公共安全），在创建真实图像分类时就会标识为-1  
 # Do not modify these IDs, since exactly these IDs are expected by the evaluation server. 不要修改这些IDs，因为这些IDs是真实服务器所期望的的值  
  
 'trainId' , # Feel free to modify these IDs as suitable for your method. Then create ground truth images with train IDs, using the tools provided in the 'preparation' folder. However, make sure to validate or submit results to our evaluation server using the regular IDs above!  
 #这列IDs可以随意修改，以满足不同的训练目的。在创建自己的真实图像分类时，可以在cityscapesScripts GitHub仓库中的preparation文件夹下寻找创建工具。但是，在验证模型，以及向评估服务器提交结果时，还是需要使用上述同一的ID  
 # For trainIds, multiple labels might have the same ID. Then, these labels are mapped to the same class in the ground truth images. For the inverse mapping, we use the label that is defined first in the list below.  
 # For example, mapping all void-type classes to the same ID in training, might make sense for some approaches. Max value is 255!  
 #对于'trainId'，多个标签可能具有相同的ID。然后这些标签映射到真实图像中同一类。例如,对于某些方法,将所有void类型的类映射到训练中的同一个ID可能是有意义的，值为255  
  
 'category' , # The name of the category that this label belongs to 此标签所属类别的名称  
  
 'categoryId' , # The ID of this category. Used to create ground truth images on category level. 这个类别的ID,用于在类别水平上创建真实图像分类  
  
 'hasInstances', # Whether this label distinguishes between single instances or not 这个标签用于区分是否有单个实例(对象)  
  
 'ignoreInEval', # Whether pixels having this class as ground truth label are ignored during evaluations or not 在评估中,像素有作为真实类标的分类被忽略,或者未被忽略  
  
 'color' , # The color of this label 类标对应的颜色  
 ] )  
  
#--------------------------------------------------------------------------------  
# A list of all labels  
#--------------------------------------------------------------------------------  
  
# Please adapt the train IDs as appropriate for you approach.  
# Note that you might want to ignore labels with ID 255 during training.  
# Further note that the current train IDs are only a suggestion. You can use whatever you like.  
# Make sure to provide your results using the original IDs and not the training IDs.  
# Note that many IDs are ignored in evaluation and thus you never need to predict these!  
  
labels = [  
 # name id trainId category catId hasInstances ignoreInEval color  
 Label( 'unlabeled' , 0 , 255 , 'void' , 0 , False , True , ( 0, 0, 0) ),  
 Label( 'ego\_vehicle' , 1 , 255 , 'void' , 0 , False , True , ( 0, 0, 0) ),  
 Label( 'rectification\_border' , 2 , 255 , 'void' , 0 , False , True , ( 0, 0, 0) ),  
 Label( 'out\_of\_roi' , 3 , 255 , 'void' , 0 , False , True , ( 0, 0, 0) ),  
 Label( 'static' , 4 , 255 , 'void' , 0 , False , True , ( 0, 0, 0) ),  
 Label( 'dynamic' , 5 , 255 , 'void' , 0 , False , True , (111, 74, 0) ),  
 Label( 'ground' , 6 , 255 , 'void' , 0 , False , True , ( 81, 0, 81) ),  
 Label( 'road' , 7 , 0 , 'flat' , 1 , False , False , (128, 64,128) ),  
 Label( 'sidewalk' , 8 , 1 , 'flat' , 1 , False , False , (244, 35,232) ),  
 Label( 'parking' , 9 , 255 , 'flat' , 1 , False , True , (250,170,160) ),  
 Label( 'rail\_track' , 10 , 255 , 'flat' , 1 , False , True , (230,150,140) ),  
 Label( 'building' , 11 , 2 , 'construction' , 2 , False , False , ( 70, 70, 70) ),  
 Label( 'wall' , 12 , 3 , 'construction' , 2 , False , False , (102,102,156) ),  
 Label( 'fence' , 13 , 4 , 'construction' , 2 , False , False , (190,153,153) ),  
 Label( 'guard\_rail' , 14 , 255 , 'construction' , 2 , False , True , (180,165,180) ),  
 Label( 'bridge' , 15 , 255 , 'construction' , 2 , False , True , (150,100,100) ),  
 Label( 'tunnel' , 16 , 255 , 'construction' , 2 , False , True , (150,120, 90) ),  
 Label( 'pole' , 17 , 5 , 'object' , 3 , False , False , (153,153,153) ),  
 Label( 'polegroup' , 18 , 255 , 'object' , 3 , False , True , (153,153,153) ),  
 Label( 'traffic\_light' , 19 , 6 , 'object' , 3 , False , False , (250,170, 30) ),  
 Label( 'traffic\_sign' , 20 , 7 , 'object' , 3 , False , False , (220,220, 0) ),  
 Label( 'vegetation' , 21 , 8 , 'nature' , 4 , False , False , (107,142, 35) ),  
 Label( 'terrain' , 22 , 9 , 'nature' , 4 , False , False , (152,251,152) ),  
 Label( 'sky' , 23 , 10 , 'sky' , 5 , False , False , ( 70,130,180) ),  
 Label( 'person' , 24 , 11 , 'human' , 6 , True , False , (220, 20, 60) ),  
 Label( 'rider' , 25 , 12 , 'human' , 6 , True , False , (255, 0, 0) ),  
 Label( 'car' , 26 , 13 , 'vehicle' , 7 , True , False , ( 0, 0,142) ),  
 Label( 'truck' , 27 , 14 , 'vehicle' , 7 , True , False , ( 0, 0, 70) ),  
 Label( 'bus' , 28 , 15 , 'vehicle' , 7 , True , False , ( 0, 60,100) ),  
 Label( 'caravan' , 29 , 255 , 'vehicle' , 7 , True , True , ( 0, 0, 90) ),  
 Label( 'trailer' , 30 , 255 , 'vehicle' , 7 , True , True , ( 0, 0,110) ),  
 Label( 'train' , 31 , 16 , 'vehicle' , 7 , True , False , ( 0, 80,100) ),  
 Label( 'motorcycle' , 32 , 17 , 'vehicle' , 7 , True , False , ( 0, 0,230) ),  
 Label( 'bicycle' , 33 , 18 , 'vehicle' , 7 , True , False , (119, 11, 32) ),  
 Label( 'license\_plate' , -1 , -1 , 'vehicle' , 7 , False , True , ( 0, 0,142) ),  
]  
  
#--------------------------------------------------------------------------------  
# Create dictionaries for a fast lookup  
#--------------------------------------------------------------------------------  
  
# Please refer to the main method below for example usages!  
  
# name to label object  
name2label = { label.name : label for label in labels }  
# id to label object  
id2label = { label.id : label for label in labels }  
# trainId to label object  
trainId2label = { label.trainId : label for label in reversed(labels) }  
# label2trainid  
label2trainid = { label.id : label.trainId for label in labels }  
# trainId to label object  
trainId2name = { label.trainId : label.name for label in labels }  
trainId2color = { label.trainId : label.color for label in labels }  
# category to list of label objects  
category2labels = {}  
for label in labels:  
 category = label.category  
 if category in category2labels:  
 category2labels[category].append(label)  
 else:  
 category2labels[category] = [label]  
   
#--------------------------------------------------------------------------------  
# Assure single instance name  
#--------------------------------------------------------------------------------  
  
# returns the label name that describes a single instance (if possible)  
# e.g. input | output  
# ----------------------  
# car | car  
# cargroup | car  
# foo | None  
# foogroup | None  
# skygroup | None  
def assureSingleInstanceName( name ):  
 # if the name is known, it is not a group  
 if name in name2label:  
 return name  
 # test if the name actually denotes a group  
 if not name.endswith("group"):  
 return None  
 # remove group  
 name = name[:-len("group")]  
 # test if the new name exists  
 if not name in name2label:  
 return None  
 # test if the new name denotes a label that actually has instances  
 if not name2label[name].hasInstances:  
 return None  
 # all good then  
 return name  
   
print(assureSingleInstanceName('ego\_vehicle' ))

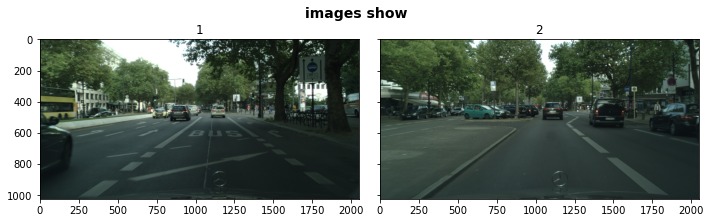
ego\_vehicle

# semantic-segmentation-main\datasets\cityscapes.py  
root = cfg.DATASET.CITYSCAPES\_DIR  
id\_to\_trainid =label2trainid  
print("id\_to\_trainid:",id\_to\_trainid)  
print("\_"\*50)  
trainid\_to\_name = trainId2name  
print("trainid\_to\_name:",trainid\_to\_name)

id\_to\_trainid: {0: 255, 1: 255, 2: 255, 3: 255, 4: 255, 5: 255, 6: 255, 7: 0, 8: 1, 9: 255, 10: 255, 11: 2, 12: 3, 13: 4, 14: 255, 15: 255, 16: 255, 17: 5, 18: 255, 19: 6, 20: 7, 21: 8, 22: 9, 23: 10, 24: 11, 25: 12, 26: 13, 27: 14, 28: 15, 29: 255, 30: 255, 31: 16, 32: 17, 33: 18, -1: -1}  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
trainid\_to\_name: {255: 'trailer', 0: 'road', 1: 'sidewalk', 2: 'building', 3: 'wall', 4: 'fence', 5: 'pole', 6: 'traffic\_light', 7: 'traffic\_sign', 8: 'vegetation', 9: 'terrain', 10: 'sky', 11: 'person', 12: 'rider', 13: 'car', 14: 'truck', 15: 'bus', 16: 'train', 17: 'motorcycle', 18: 'bicycle', -1: 'license\_plate'}

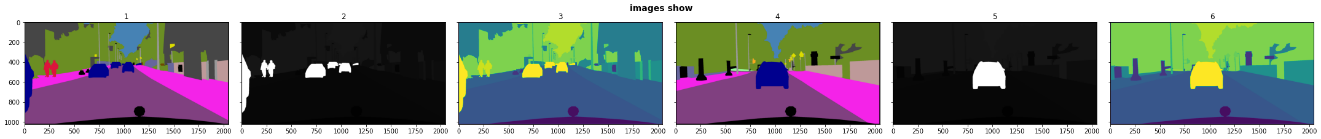
# semantic-segmentation-main\datasets\cityscapes.py  
def fill\_colormap():  
 palette = [128, 64, 128,  
 244, 35, 232,  
 70, 70, 70,  
 102, 102, 156,  
 190, 153, 153,  
 153, 153, 153,  
 250, 170, 30,  
 220, 220, 0,  
 107, 142, 35,  
 152, 251, 152,  
 70, 130, 180,  
 220, 20, 60,  
 255, 0, 0,  
 0, 0, 142,  
 0, 0, 70,  
 0, 60, 100,  
 0, 80, 100,  
 0, 0, 230,  
 119, 11, 32]  
 zero\_pad = 256 \* 3 - len(palette)  
 for i in range(zero\_pad):  
 palette.append(0)  
 return palette  
  
color\_mapping=fill\_colormap()

# semantic-segmentation-main\datasets\cityscapes.py  
import os  
import os.path as path  
img\_root = path.join(root, 'leftImg8bit\_trainvaltest/leftImg8bit')  
mask\_root = path.join(root, 'gtFine\_trainvaltest/gtFine')  
  
import util\_misc  
imgs\_fn=util\_misc.filePath\_extraction(img\_root,["png"])   
imgs\_root=list(imgs\_fn.keys())[0]  
imgsFn\_lst=imgs\_fn[imgs\_root]  
imgsFn\_lst.sort()  
imgsFn\_lst\_=imgsFn\_lst[:2]  
  
columns=2  
scale=1  
  
util\_misc.imgs\_layoutShow(imgs\_root,imgsFn\_lst\_,columns,scale,figsize=(10,3))  
print(imgsFn\_lst\_)



['berlin\_000000\_000019\_leftImg8bit.png', 'berlin\_000001\_000019\_leftImg8bit.png']

mask\_fn=util\_misc.filePath\_extraction(mask\_root,["png"])   
mask\_root\_list=list(mask\_fn.keys())  
mask\_root\_list.sort()  
mask\_root\_=mask\_root\_list[15]  
maskFn\_lst=mask\_fn[mask\_root\_]  
#maskFn\_lst.sort()  
maskFn\_lst\_=maskFn\_lst[:6]  
columns=6  
scale=1  
  
util\_misc.imgs\_layoutShow(mask\_root\_,maskFn\_lst\_,columns,scale,figsize=(30,3))  
print(maskFn\_lst\_)



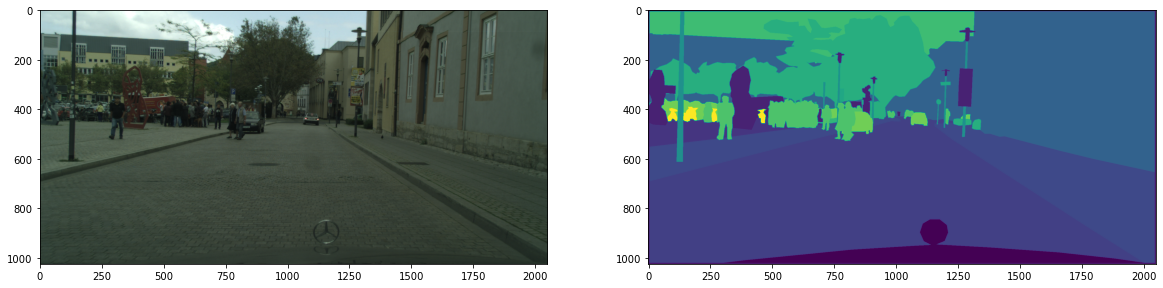
['jena\_000000\_000019\_gtFine\_color.png', 'jena\_000000\_000019\_gtFine\_instanceIds.png', 'jena\_000000\_000019\_gtFine\_labelIds.png', 'jena\_000001\_000019\_gtFine\_color.png', 'jena\_000001\_000019\_gtFine\_instanceIds.png', 'jena\_000001\_000019\_gtFine\_labelIds.png']

#### 3）torchvision.datasets.Cityscapes方法读取cityscapes数据

输入参数root为包含有leftImg8bit\_trainextra.zip、leftImg8bit\_trainvaltest.zip、gtCoarse.zip和gtFine\_trainvaltest.zip所在位置的文件夹。

from torchvision.datasets import Cityscapes  
dataset=Cityscapes(r'G:\data\cityscapes', split='train', mode='fine',target\_type='semantic')  
img, smnt=dataset[0]

import matplotlib.pyplot as plt  
plt.rcParams["figure.figsize"] = (20,20)  
plt.subplot(121)  
plt.imshow(img)  
plt.subplot(122)  
plt.imshow(smnt)  
plt.show()



### 2.6.5.2 开放神经网络交换——ONNX（Open Neural Network，Exchage）与Netron网络可视化工具

#### 1）开放神经网络交换——ONNX（Open Neural Network，Exchage）

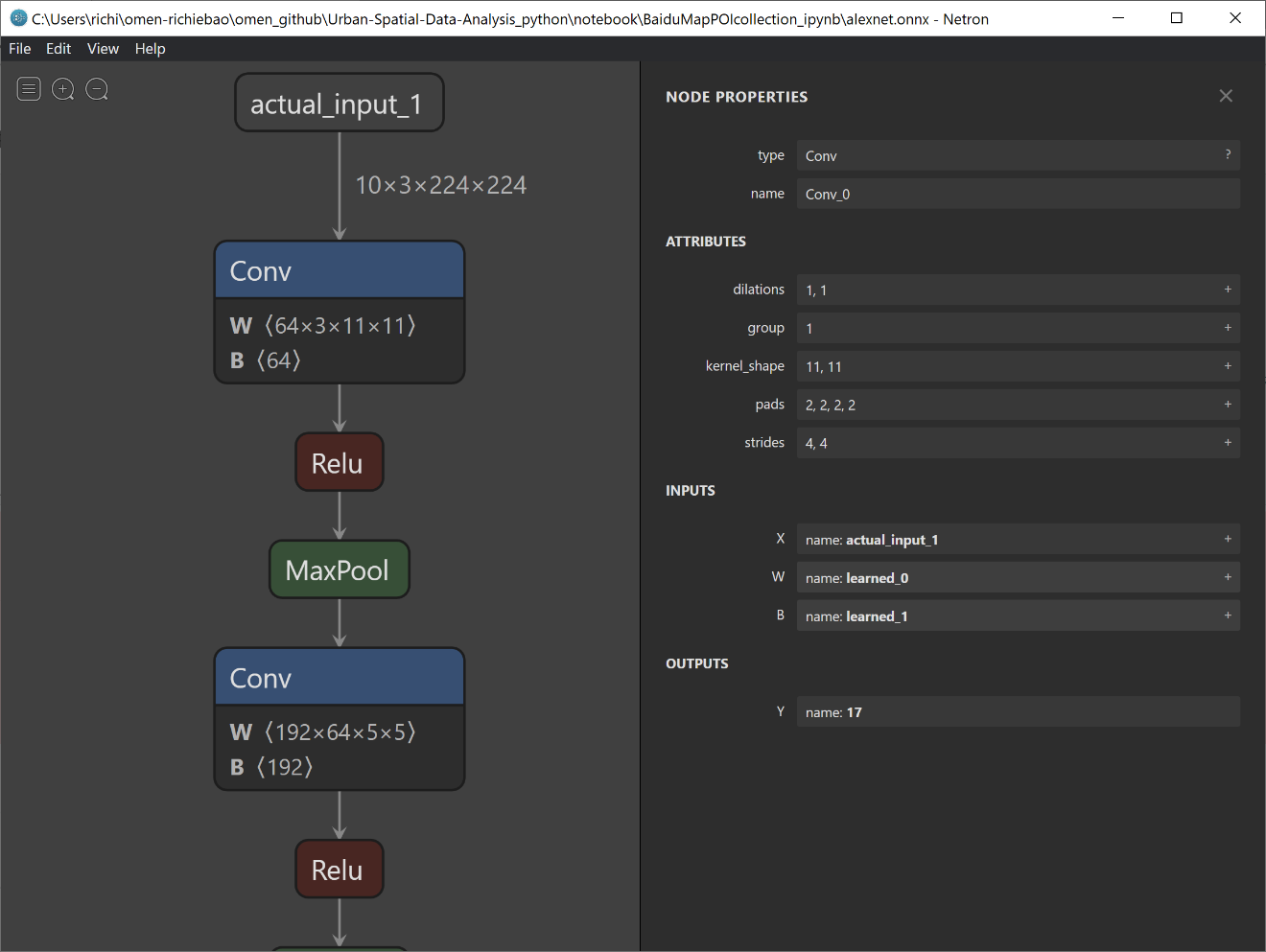
[开放神经网络交换——ONNX](https://onnx.ai/supported-tools.html)⑥是针对深度学习所设计的开放式文件格式，用于存储训练好的模型。ONNX相信人工智能社区需要更强的互操作性，不会被局限于一种人工智能框架，使得不同类型的人工智能框架（例如PyTorch，MXNet)可以采用相同格式存储模型数据并交互，共享模型。例如使用PyTorch的人工智能框架（深度学习），并利用[torchvision.models](https://pytorch.org/vision/0.8/models.html)⑦（预训练）模型库，实现城市空间下行人的对象检测来估算人流量变化；应用对象分割（Instance Segmentation），统计城市空间对象内容，建立关联网络结构分析对象之间的关系。因此预训练好的模型可以帮助其它领域的研究者迅速应用已有的研究模型，避免重新构建模型网络，及不菲的训练时间成本（即使有多GPU加速，有些海量数据集或高度复杂的网络，训练时间长度也相当惊人）。非计算机科学领域的研究者，往往也不具有理想的硬件条件，因此预训练好的模型共享，显得尤为重要，这也是将人工智能从研究带到现实的有效途径。

[ONNX Model Zoo](https://github.com/onnx/models)⑧汇集了已有的大量模型，包括的种类有，

1. Vision 视觉类：Image Classification 图像分类，Object Detection & Image Segmentation 对象检测和图像分割，Body, Face & Gesture Analysis 人体，人脸和姿势分析，Image Manipulation 图像处理；
2. Language 语言类：Machine Comprehension 机器理解，Machine Translation 机器翻译，Language Modelling 语言模型；
3. 及其它类：Visual Question Answering & Dialog 视觉问答&对话，Speech & Audio Processing 语音和音频处理，和其它有趣的模型。

#### 2）[Netron](https://github.com/lutzroeder/Netron)⑨ 网络可视化工具

tensorboard可以在深度学习模型建立过程中帮助分析网络结构，显示训练图像和预测精度，及损失曲线等。但是如果想更加方便的查看模型网络结构，Netron可以直接读取开放神经网络交换格式(.onnx)。torch.onnx.export方法可以将PyTorch模型导出为.onnx交换格式，默认export\_params=True，保存预训练模型的参数。直接用Netron工具打开，可以查看到下述模型的网络结构。



### 2.6.5.3 DUC（Dense Upsampling Convolution）图像分割（object\_detection\_segmentation）

目前torchvision.models图像分割部分的预训练模型主要是针对COCO数据集子集Pascal VOC，包括有20个分类，并且分类中包括了部分室内物品，这不能够满足室外城市街道环境的语义分割。在ONNX Model Zoo汇集的大量模型中，语义分割部分可获取的模型目前只有[DUC](https://github.com/onnx/models/tree/master/vision/object_detection_segmentation/duc)⑩，具体实现的方法在[Inference demo for DUC models](https://github.com/onnx/models/blob/master/vision/object_detection_segmentation/duc/dependencies/duc-inference.ipynb)⑪ notebook中给出了阐述，使用的深度学习框架是[MXNet](https://mxnet.apache.org/)⑫，MXNet在PyTorch和TensorFlow双重夹击下发展缓慢，其架构从安装到应用并不友好，因此不建议使用。但是因为该DUC模型提供了已经训练好的模型，因此下述仍然使用MXNet框架。具体安装及相关依赖库在notebook文件中均有说明。

DUC所使用的数据集为针对建筑外街道环境的Cityscapes数据集。

当读者察看该部分时，最好先在torchvision.models中查看是否有类似针对Cityscapes数据集的图像分割预训练模型，或者在ONNX Model Zoo中是否有新的更新，及任何深度学习模型库中查找比较选择，使用最新和预测高分的预训练模型，进行图像的语义分割。

* A-调入依赖库

import mxnet as mx  
import cv2 as cv  
import numpy as np  
import os  
from PIL import Image  
import math  
from collections import namedtuple  
from mxnet.contrib.onnx import import\_model  
import cityscapes\_labels

* B-图像预处理，其颜色通道RGB均减去RGB的均值，并转换为MXNet的ndarray数据格式。

def preprocess(im):  
 # Convert to float32  
 test\_img = im.astype(np.float32)  
 # Extrapolate image with a small border in order obtain an accurate reshaped image after DUC layer  
 test\_shape = [im.shape[0],im.shape[1]]  
 cell\_shapes = [math.ceil(l / 8)\*8 for l in test\_shape]  
 test\_img = cv.copyMakeBorder(test\_img, 0, max(0, int(cell\_shapes[0]) - im.shape[0]), 0, max(0, int(cell\_shapes[1]) - im.shape[1]), cv.BORDER\_CONSTANT, value=rgb\_mean)  
 test\_img = np.transpose(test\_img, (2, 0, 1))  
 # subtract rbg mean  
 for i in range(3):  
 test\_img[i] -= rgb\_mean[i]  
 test\_img = np.expand\_dims(test\_img, axis=0)  
 # convert to ndarray  
 test\_img = mx.ndarray.array(test\_img)  
 return test\_img

* C-get\_palette()：返回用于生成输出分割图的预定义调色板； colorize(): 使用由模型生成的输出类标和get\_palette()建立的分割图调色板构建分割映射；predict(): 向模型传入预处理图像，执行前向传播，并将预测输出数据重新调整为输入图像的形状，使用colorize()生成彩色分割的分类掩码图。

def get\_palette():  
 # get train id to color mappings from file  
 trainId2colors = {label.trainId: label.color for label in cityscapes\_labels.labels}  
 # prepare and return palette  
 palette = [0] \* 256 \* 3  
 for trainId in trainId2colors:  
 colors = trainId2colors[trainId]  
 if trainId == 255:  
 colors = (0, 0, 0)  
 for i in range(3):  
 palette[trainId \* 3 + i] = colors[i]  
 return palette  
  
def colorize(labels):  
 # generate colorized image from output labels and color palette  
 result\_img = Image.fromarray(labels).convert('P')  
 result\_img.putpalette(get\_palette())  
 return np.array(result\_img.convert('RGB'))  
  
def predict(imgs):  
 # get input and output dimensions  
 result\_height, result\_width = result\_shape  
 \_, \_, img\_height, img\_width = imgs.shape  
 # set downsampling rate  
 ds\_rate = 8  
 # set cell width  
 cell\_width = 2  
 # number of output label classes  
 label\_num = 19  
   
 # Perform forward pass  
 batch = namedtuple('Batch', ['data'])  
 mod.forward(batch([imgs]),is\_train=False)  
 labels = mod.get\_outputs()[0].asnumpy().squeeze()  
  
 # re-arrange output  
 test\_width = int((int(img\_width) / ds\_rate) \* ds\_rate)  
 test\_height = int((int(img\_height) / ds\_rate) \* ds\_rate)  
 feat\_width = int(test\_width / ds\_rate)  
 feat\_height = int(test\_height / ds\_rate)  
 labels = labels.reshape((label\_num, 4, 4, feat\_height, feat\_width))  
 labels = np.transpose(labels, (0, 3, 1, 4, 2))  
 labels = labels.reshape((label\_num, int(test\_height / cell\_width), int(test\_width / cell\_width)))  
  
 labels = labels[:, :int(img\_height / cell\_width),:int(img\_width / cell\_width)]  
 labels = np.transpose(labels, [1, 2, 0])  
 labels = cv.resize(labels, (result\_width, result\_height), interpolation=cv.INTER\_LINEAR)  
 labels = np.transpose(labels, [2, 0, 1])  
   
 # get softmax output  
 softmax = labels  
   
 # get classification labels  
 results = np.argmax(labels, axis=0).astype(np.uint8)  
 raw\_labels = results  
  
 # comput confidence score  
 confidence = float(np.max(softmax, axis=0).mean())  
  
 # generate segmented image  
 result\_img = Image.fromarray(colorize(raw\_labels)).resize(result\_shape[::-1])  
   
 # generate blended image  
 blended\_img = Image.fromarray(cv.addWeighted(im[:, :, ::-1], 0.5, np.array(result\_img), 0.5, 0))  
  
 return confidence, result\_img, blended\_img, raw\_labels

* D-加载预训练的模型。导入ONNX预训练模型到MxNet中，使用符号文件（symbol file）定义模型，使用参数文件（params file）绑定参数。

def get\_model(ctx, model\_path):  
 # import ONNX model into MXNet symbols and params  
 sym,arg,aux = import\_model(model\_path)  
 # define network module  
 mod = mx.mod.Module(symbol=sym, data\_names=['data'], context=ctx, label\_names=None)  
 # bind parameters to the network  
 mod.bind(for\_training=False, data\_shapes=[('data', (1, 3, im.shape[0], im.shape[1]))], label\_shapes=mod.\_label\_shapes)  
 mod.set\_params(arg\_params=arg, aux\_params=aux,allow\_missing=True, allow\_extra=True)  
 return mod

* E-给出一张[KITTI](http://www.cvlibs.net/datasets/kitti/index.php)⑬数据中的影像并显示

im = cv.imread('./data/0000000389.png')[:, :, ::-1]  
# set output shape (same as input shape)  
result\_shape = [im.shape[0],im.shape[1]]  
# set rgb mean of input image (used in mean subtraction)  
rgb\_mean = cv.mean(im)

# display input image  
Image.fromarray(im)

* F-可以手动从<https://s3.amazonaws.com/onnx-model-zoo/duc/ResNet101_DUC_HDC.onnx>下载预训练模型。并设置使用GPU，还是CPU。

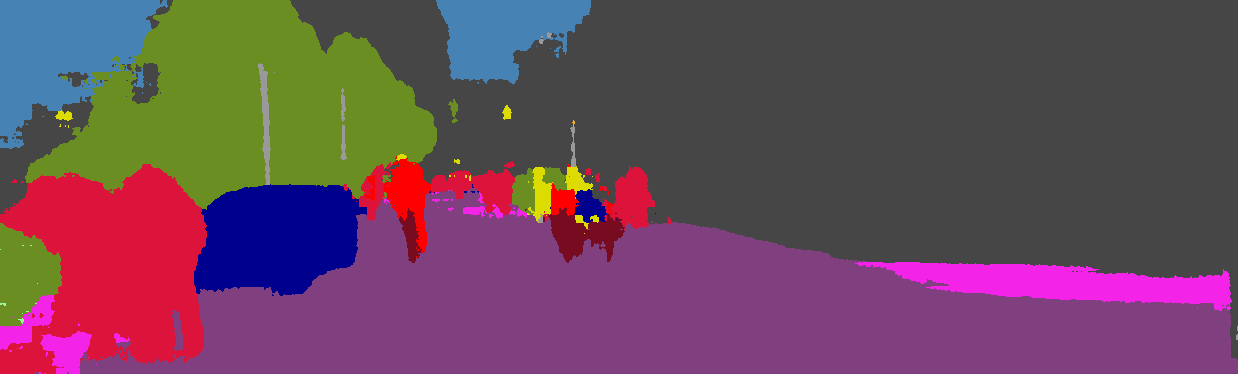
# Download ONNX model  
# mx.test\_utils.download('https://s3.amazonaws.com/onnx-model-zoo/duc/ResNet101\_DUC\_HDC.onnx')  
# Determine and set context  
if len(mx.test\_utils.list\_gpus())==0:  
 ctx = mx.cpu()  
else:  
 ctx = mx.gpu(0)

# Load ONNX model  
mod = get\_model(ctx, r'./model/ResNet101\_DUC\_HDC.onnx')  
print("The model is loaded...")

The model is loaded...

* G-处理输入图像，并执行预测，查看预测结果

pre = preprocess(im)  
conf,result\_img,blended\_img,raw = predict(pre)  
result\_img



* H-混合输出。分割图与真实图叠合，方便观察预测结果。

blended\_img



* I-查看精度（confidence score）。为SoftMax回归分类模型输出的联合概率分布最大值。数值位于[0,1]，数值越大，像素属于某一分类可能性越大。

print('Confidence = %f' %(conf))

Confidence = 0.929088

### 2.6.5.4 城市空间要素组成，时空量度，绿视率和均衡度

cityscapes数据集，标签/分类包括主要的城市街道场景内容，这为城市空间的分析提供了基础的数据支持，例如对于固定行进流线，视野方向和宽度下，通过标签vegetation可以计算绿视率（Green view index），当绿视率达到一定水平，会让行人在街道空间中觉得舒适；通过sky可以获知视野下所见天空的比例，这与天空视域因子（Sky View Factor，SVF）可以比较研究；对于其它项，例如car、truck和bus 可以初步判断某一时刻街道的交通情况；person和rider则可以初步判断行人情况。根据待分析的内容可以有意识的选择对应的要素进行分析，也可以综合考虑所有因素，计算每一位置的信息熵和均衡度，比较不同位置的混杂程度，通常混杂比较高的位置可能感觉会比较热闹，而低的区域则相对简单和冷清。

因为将DUC预训练模型用于KITTI数据集，无人驾驶项目拍摄的连续图像，是固定行进流线，视野方向和宽度的，这可以保证图像具有统一的属性，避免因为拍摄上下角度变化的问题，使得图像之间不具有比较性。预测计算实际较长，为了避免数据丢失，将DUC预测的结果，conf 精度/概率；result\_img 语义分割的掩码图像（颜色区分）；blended\_img 叠合掩码和实际的图像，可方便查看分割与实际之间的差异；raw trainId数字索引，分别保存为图像格式、及存储在列表下，用pickle保存。

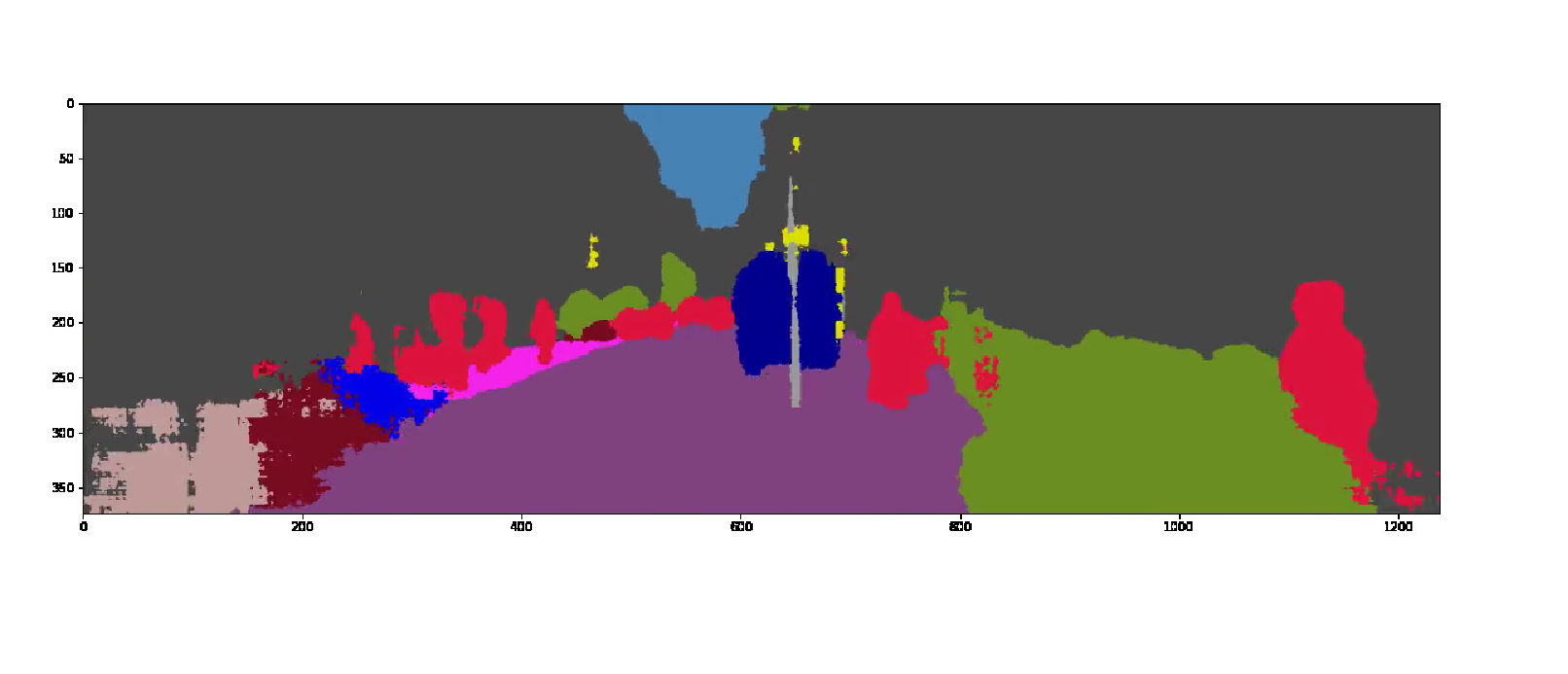
import glob  
drive\_29\_0071\_img\_fp\_list=glob.glob(r"G:/data/2011\_09\_29\_drive\_0071\_sync/image\_03/data/\*.png")  
drive\_29\_0071\_img\_fp\_list.sort()

def sementicSeg\_DUC\_pred(DUC\_output\_root,img\_fps,preprocess,predict):  
 '''  
 function - DUC图像分割，及预测图像保存  
 '''  
 from tqdm.auto import tqdm  
 import os,pickle  
 import cv2 as cv  
 from tqdm.auto import tqdm  
   
 DUC\_output\_root=DUC\_output\_root  
 conf\_list=[]  
 raw\_list=[]  
 for i,img in enumerate(tqdm(img\_fps)):  
 im=cv.imread(img)[:, :, ::-1]  
 # set output shape (same as input shape)  
 result\_shape = [im.shape[0],im.shape[1]]  
 # set rgb mean of input image (used in mean subtraction)  
 rgb\_mean = cv.mean(im)  
 pre=preprocess(im)  
 conf,result\_img,blended\_img,raw=predict(pre)  
 conf\_list.append(conf)  
 raw\_list.append(raw)  
  
 if not os.path.exists(os.path.join(DUC\_output\_root,"result\_img")):  
 os.makedirs(os.path.join(DUC\_output\_root,"result\_img"))  
 if not os.path.exists(os.path.join(DUC\_output\_root,"blended\_img")):  
 os.makedirs(os.path.join(DUC\_output\_root,"blended\_img"))   
 result\_img.save(os.path.join(DUC\_output\_root,"result\_img/result\_img\_{}.png".format(i)))  
 blended\_img.save(os.path.join(DUC\_output\_root,"blended\_img/blended\_img\_{}.png".format(i)))  
  
 with open(os.path.join(DUC\_output\_root,'KITTI\_DUC\_confi.pkl'),'wb') as f1:   
 pickle.dump(conf\_list,f1)   
 with open(os.path.join(DUC\_output\_root,'KITTI\_DUC\_raw.pkl'),'wb') as f2:   
 pickle.dump(raw\_list,f2)   
   
DUC\_output\_root=r'G:\data\data\_processed\KITTI\_DUC'  
sementicSeg\_DUC\_pred(DUC\_output\_root,img\_fps=drive\_29\_0071\_img\_fp\_list,preprocess=preprocess,predict=predict)

0%| | 0/1059 [00:00<?, ?it/s]

import matplotlib.pyplot as plt  
import matplotlib.animation as animation  
import matplotlib.image as mpimg  
from IPython.display import HTML  
import os,glob  
  
DUC\_output\_root=r'G:\data\data\_processed\KITTI\_DUC'  
result\_img\_fp\_list=glob.glob(DUC\_output\_root+r'/result\_img/\*.png')  
  
fig=plt.figure(figsize=(20,10))  
ims=[[plt.imshow(mpimg.imread(f),animated=True)] for f in result\_img\_fp\_list[:]]  
print("finished reading the imgs.")  
  
ani=animation.ArtistAnimation(fig, ims, interval=50, blit=True,repeat\_delay=1000) # conda install -c conda-forge ffmpeg  
ani.save(os.path.join(DUC\_output\_root,r'DUC\_result\_imgs.mp4'))  
print(".mp4 saved.")  
HTML(ani.to\_html5\_video())

finished reading the imgs.  
.mp4 saved.



对每一位置计算所有对象的频数，可以得知各对象在图像所代表的视野下占的比例。

def DUC\_pred\_frequency\_moment(KITTI\_DUC\_raw\_fp,KITTI\_DUC\_confi\_fp):  
 '''  
 function - 读取DUC语义分割结果保存的'KITTI\_DUC\_confi.pkl'和'KITTI\_DUC\_raw.pkl'文件，用于位置图像对象/语义分割类别频数统计  
 '''  
 import pickle  
 import numpy as np  
 import pandas as pd  
 from tqdm.auto import tqdm  
   
 with open(KITTI\_DUC\_raw\_fp,'rb') as f:  
 KITTI\_DUC\_raw=pickle.load(f)  
 with open(KITTI\_DUC\_confi\_fp,'rb') as f:  
 KITTI\_DUC\_confi=pickle.load(f)   
  
 unique\_id\_all=np.unique(np.stack(KITTI\_DUC\_raw))  
 print("所有出现的id:{}".format(unique\_id\_all)) #对应trainId  
 id\_info\_df=pd.DataFrame(columns=unique\_id\_all)  
   
 for seg in tqdm(KITTI\_DUC\_raw):  
 unique\_id, counts\_id=np.unique(seg, return\_counts=True)  
 id\_fre\_dic=dict(list(zip(unique\_id, counts\_id)))  
 for i in unique\_id\_all:  
 if i not in unique\_id.tolist():  
 id\_fre\_dic.setdefault(i,0)  
 id\_info\_df=id\_info\_df.append(id\_fre\_dic,ignore\_index=True)   
   
 id\_info\_df["confidence"]=KITTI\_DUC\_confi  
 return id\_info\_df,unique\_id\_all  
   
KITTI\_DUC\_raw\_fp=r'G:\data\data\_processed\KITTI\_DUC\KITTI\_DUC\_raw.pkl'   
KITTI\_DUC\_confi\_fp=r'G:\data\data\_processed\KITTI\_DUC\KITTI\_DUC\_confi.pkl'  
id\_info\_df,unique\_id\_all=DUC\_pred\_frequency\_moment(KITTI\_DUC\_raw\_fp,KITTI\_DUC\_confi\_fp)

所有出现的id:[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18]  
  
  
  
 0%| | 0/1059 [00:00<?, ?it/s]

读取KITTI数据集的经纬度位置信息和时间戳。和频数的DataFrame格式数据合并在一个DataFrame之下，方便后续数据处理。

import util\_A  
KITTI\_info\_fp=r'G:\data\2011\_09\_29\_drive\_0071\_sync\oxts\data'  
timestamps\_fp=r'G:\data\2011\_09\_29\_drive\_0071\_sync\image\_03\timestamps.txt'  
drive\_29\_0071\_info=util\_A.KITTI\_info(KITTI\_info\_fp,timestamps\_fp)  
drive\_29\_0071\_info\_coordi=drive\_29\_0071\_info[['lat','lon','timestamps\_']]

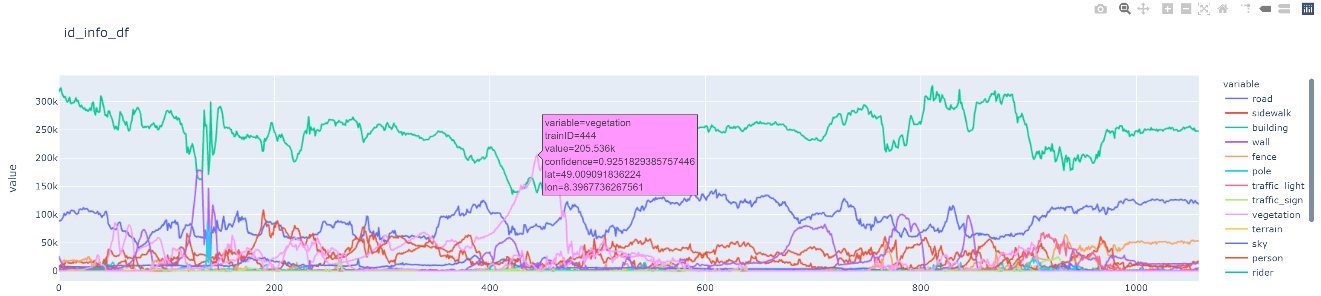
trainID\_label\_mapping={id\_:trainId2label[id\_].name for id\_ in unique\_id\_all} #建立trainID到类别的映射字典，trainId2label方法向上，在cityscapes的标签数据处理部分  
id\_info\_df=id\_info\_df.rename(columns=trainID\_label\_mapping)  
id\_info\_df['trainID']=id\_info\_df.index  
id\_info\_df=id\_info\_df.join(drive\_29\_0071\_info\_coordi)  
id\_info\_df

|  | **road** | **sidewalk** | **building** | **wall** | **fence** | **pole** | **traffic\_light** | **traffic\_sign** | **vegetation** | **terrain** | **...** | **truck** | **bus** | **train** | **motorcycle** | **bicycle** | **confidence** | **trainID** | **lat** | **lon** | **timestamps\_** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 84650 | 10430 | 319765 | 1696 | 0 | 1588 | 0 | 260 | 2589 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0.957152 | 0 | 49.008650 | 8.398092 | 2011-09-29 13:54:59.990872576 |
| **1** | 86454 | 9873 | 322688 | 2318 | 413 | 1229 | 0 | 84 | 1693 | 0 | ... | 0 | 0 | 0 | 4 | 652 | 0.957426 | 1 | 49.008777 | 8.397611 | 2011-09-29 13:55:00.094612992 |
| **2** | 87182 | 12023 | 325361 | 2264 | 65 | 595 | 0 | 175 | 1250 | 0 | ... | 0 | 0 | 0 | 8 | 451 | 0.957314 | 2 | 49.009162 | 8.396541 | 2011-09-29 13:55:00.198486528 |
| **3** | 89886 | 10992 | 319038 | 1831 | 9483 | 615 | 0 | 8 | 1454 | 0 | ... | 0 | 0 | 0 | 0 | 520 | 0.947994 | 3 | 49.008962 | 8.397075 | 2011-09-29 13:55:00.302340864 |
| **4** | 91569 | 14370 | 311801 | 1000 | 18342 | 1474 | 0 | 81 | 1430 | 0 | ... | 0 | 0 | 0 | 0 | 602 | 0.946398 | 4 | 49.009505 | 8.395251 | 2011-09-29 13:55:00.406079232 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1054** | 124321 | 13098 | 244613 | 11616 | 51167 | 1354 | 50 | 997 | 53 | 0 | ... | 0 | 949 | 18 | 0 | 0 | 0.942456 | 1054 | 49.009215 | 8.396286 | 2011-09-29 13:56:49.458599424 |
| **1055** | 126360 | 10678 | 241321 | 14036 | 50878 | 1703 | 81 | 993 | 4 | 0 | ... | 0 | 1152 | 4 | 0 | 0 | 0.942314 | 1055 | 49.009353 | 8.395764 | 2011-09-29 13:56:49.562463744 |
| **1056** | 122516 | 14267 | 241194 | 14602 | 50789 | 1880 | 102 | 1000 | 0 | 0 | ... | 0 | 313 | 0 | 0 | 0 | 0.941350 | 1056 | 49.008706 | 8.397888 | 2011-09-29 13:56:49.666327808 |
| **1057** | 122223 | 14577 | 241842 | 14740 | 49996 | 1962 | 34 | 917 | 0 | 0 | ... | 0 | 714 | 0 | 0 | 0 | 0.941474 | 1057 | 49.009215 | 8.396288 | 2011-09-29 13:56:49.770316544 |
| **1058** | 121671 | 14120 | 241772 | 14341 | 51080 | 2141 | 166 | 864 | 0 | 0 | ... | 0 | 561 | 0 | 0 | 0 | 0.939658 | 1058 | 49.009079 | 8.396812 | 2011-09-29 13:56:49.874179584 |

1059 rows × 24 columns

打印所有位置时刻，每一图像包含对象的面积频数，即各对象像素占整体图像像素的比例。如果要查看单个对象，可以点击图例对应项。因为’2011\_09\_29\_drive\_0071\_sync’部分数据位于城市的街巷内，因此可以明显的观察到’building’对象所占的数量较大，次之则为’road’，其它对象相对较小。’vegetation’在部分区域有较高的比例。

import plotly.express as px  
labels=list(trainID\_label\_mapping.values())  
fig = px.line(id\_info\_df, x="trainID", y=labels,  
 hover\_data=['confidence','lat','lon','trainID'],  
 title='id\_info\_df'  
 )  
fig.show()



可以打印感兴趣的对象，观察在实际空间地理位置上的分布情况。例如’sky’和’vegetation’的分布，通过量化的方式能够明确变化方式的具体位置。

fig = px.density\_mapbox(id\_info\_df, lat='lat', lon='lon', z='sky', radius=10,  
 center=dict(lat=49.008645, lon=8.398104), zoom=18,  
 mapbox\_style="stamen-terrain",  
 hover\_data=['confidence','lat','lon','trainID'],  
 title=r'sky Kernel Density')  
fig.show()



import plotly.express as px  
fig = px.density\_mapbox(id\_info\_df, lat='lat', lon='lon', z='vegetation', radius=10,  
 center=dict(lat=49.008645, lon=8.398104), zoom=18,  
 mapbox\_style="stamen-terrain",  
 hover\_data=['confidence','lat','lon','trainID'],  
 title=r'vegetation Kernel Density')  
fig.show()



虽然对象的像素数量可以比较不同对象所占的比例，但是计算各自对象所占的百分比，则更容易得知对象在整个图像视野中比例的变化情况。

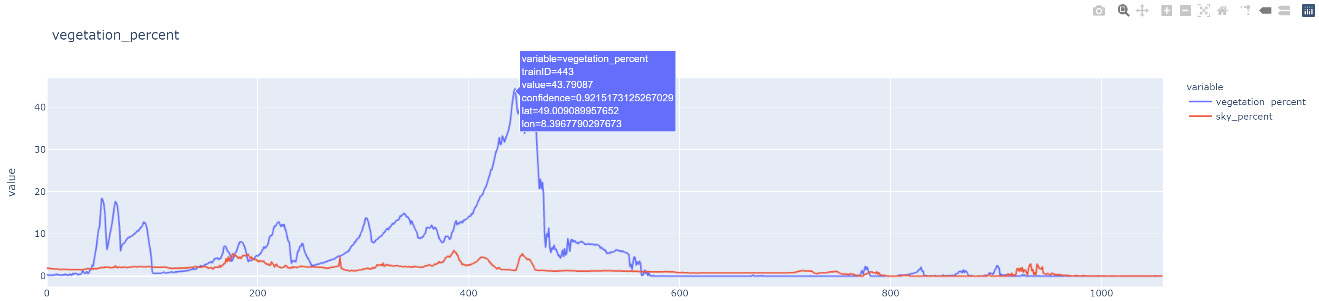
labels=list(trainID\_label\_mapping.values())  
sum\_syntax='pixels='+''.join("%s+"%''.join(map(str,x)) for x in labels)[:-1]   
id\_info\_df\_int=id\_info\_df[labels].astype(int)  
id\_info\_df\_int=id\_info\_df\_int.eval(sum\_syntax)  
id\_info\_df\_int['vegetation\_percent']=id\_info\_df\_int.apply(lambda row:row.vegetation/row.pixels\*100,axis=1)  
id\_info\_df\_int['sky\_percent']=id\_info\_df\_int.apply(lambda row:row.sky/row.pixels\*100,axis=1)  
id\_info\_df\_int

|  | **road** | **sidewalk** | **building** | **wall** | **fence** | **pole** | **traffic\_light** | **traffic\_sign** | **vegetation** | **terrain** | **...** | **rider** | **car** | **truck** | **bus** | **train** | **motorcycle** | **bicycle** | **pixels** | **vegetation\_percent** | **sky\_percent** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 84650 | 10430 | 319765 | 1696 | 0 | 1588 | 0 | 260 | 2589 | 1 | ... | 249 | 25563 | 0 | 0 | 0 | 0 | 0 | 463012 | 0.559165 | 1.908590 |
| **1** | 86454 | 9873 | 322688 | 2318 | 413 | 1229 | 0 | 84 | 1693 | 0 | ... | 1212 | 21695 | 0 | 0 | 0 | 4 | 652 | 463012 | 0.365649 | 1.878137 |
| **2** | 87182 | 12023 | 325361 | 2264 | 65 | 595 | 0 | 175 | 1250 | 0 | ... | 889 | 17241 | 0 | 0 | 0 | 8 | 451 | 463012 | 0.269971 | 1.838829 |
| **3** | 89886 | 10992 | 319038 | 1831 | 9483 | 615 | 0 | 8 | 1454 | 0 | ... | 1553 | 12757 | 0 | 0 | 0 | 0 | 520 | 463012 | 0.314031 | 1.809240 |
| **4** | 91569 | 14370 | 311801 | 1000 | 18342 | 1474 | 0 | 81 | 1430 | 0 | ... | 1771 | 6855 | 0 | 0 | 0 | 0 | 602 | 463012 | 0.308847 | 1.780947 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1054** | 124321 | 13098 | 244613 | 11616 | 51167 | 1354 | 50 | 997 | 53 | 0 | ... | 0 | 13358 | 0 | 949 | 18 | 0 | 0 | 463012 | 0.011447 | 0.048163 |
| **1055** | 126360 | 10678 | 241321 | 14036 | 50878 | 1703 | 81 | 993 | 4 | 0 | ... | 0 | 14228 | 0 | 1152 | 4 | 0 | 0 | 463012 | 0.000864 | 0.057882 |
| **1056** | 122516 | 14267 | 241194 | 14602 | 50789 | 1880 | 102 | 1000 | 0 | 0 | ... | 0 | 14339 | 0 | 313 | 0 | 0 | 0 | 463012 | 0.000000 | 0.031965 |
| **1057** | 122223 | 14577 | 241842 | 14740 | 49996 | 1962 | 34 | 917 | 0 | 0 | ... | 0 | 14158 | 0 | 714 | 0 | 0 | 0 | 463012 | 0.000000 | 0.029589 |
| **1058** | 121671 | 14120 | 241772 | 14341 | 51080 | 2141 | 166 | 864 | 0 | 0 | ... | 0 | 14716 | 0 | 561 | 0 | 0 | 0 | 463012 | 0.000000 | 0.021814 |

1059 rows × 22 columns

打印植被百分比的分布，可以得知该街道植被在前半部分开始逐渐增加，在中心广场部分则有相对较多的树木栽植，但是到了后半段，则迅速减少。同时也加入了天空的百分比。

import plotly.express as px  
labels=list(trainID\_label\_mapping.values())  
extracted\_labels=['vegetation\_percent','sky\_percent']  
hover\_data=['confidence','lat','lon','trainID']  
id\_info\_df\_int[hover\_data]=id\_info\_df[hover\_data]  
fig = px.line(id\_info\_df\_int, x="trainID", y=extracted\_labels,  
 hover\_data=hover\_data,  
 title='vegetation\_percent'  
 )  
fig.show()



将“KITTI动态街景视觉感知”部分的代码加入到util\_A模块中，此处调用计算视觉感知消失的距离，获得开始点和消失点的索引，计算每一位置下与感知距离消失位置间均衡度的变化。这一信息的比较可以粗略的得知当前位置到视觉感知消失位置城市空间场景混杂程度的变化。 当差值绝对值较大时，说明场景在混杂丰富和简单冷清间互相变换，即视觉消失的位置场景与当前场景有很大不同；如果差值绝对值较小，则说明城市空间对象的混合程度基本保持不变，即视觉消失的位置场景与当前场景类似。

对于需要花时间计算的内容，通常都将其保存到本地磁盘中，需要时直接调用，避免重复计算。

import util\_A  
import glob  
  
drive\_29\_0071\_img\_fp\_list=glob.glob("G:/data/2011\_09\_29\_drive\_0071\_sync/image\_03/data/\*.png")  
drive\_29\_0071\_img\_fp\_list.sort()  
dsv\_vp=util\_A.dynamicStreetView\_visualPerception(drive\_29\_0071\_img\_fp\_list[:]) #[:200] #pip install opencv-python and pip install opencv-contrib-python   
matches\_num=dsv\_vp.sequence\_statistics()

计算关键点和描述子...  
  
  
100%|██████████| 1059/1059 [01:16<00:00, 13.87it/s]  
  
  
计算序列图像匹配数...  
  
  
100%|██████████| 1058/1058 [31:31<00:00, 1.79s/it]

import util\_A  
  
KITTI\_info\_fp=r'G:\data\2011\_09\_29\_drive\_0071\_sync\oxts\data'  
timestamps\_fp=r'G:\data\2011\_09\_29\_drive\_0071\_sync\image\_03\timestamps.txt'  
drive\_29\_0071\_info=util\_A.KITTI\_info(KITTI\_info\_fp,timestamps\_fp)  
drive\_29\_0071\_info\_coordi=drive\_29\_0071\_info[['lat','lon','timestamps\_']]  
  
coordi\_df=drive\_29\_0071\_info\_coordi  
vanishing\_gpd=vanishing\_position\_length(matches\_num,coordi\_df,epsg="EPSG:3857",threshold=0)  
vanishing\_gpd.to\_pickle('./results/drive\_29\_0071\_vanishing\_gpd.pkl')

C:\Users\richi\anaconda3\envs\USDA\_database\lib\site-packages\pandas\core\dtypes\cast.py:91: ShapelyDeprecationWarning: The array interface is deprecated and will no longer work in Shapely 2.0. Convert the '.coords' to a numpy array instead.  
 values = construct\_1d\_object\_array\_from\_listlike(values)  
C:\Users\richi\anaconda3\envs\USDA\_database\lib\site-packages\pyproj\crs\crs.py:131: FutureWarning: '+init=<authority>:<code>' syntax is deprecated. '<authority>:<code>' is the preferred initialization method. When making the change, be mindful of axis order changes: https://pyproj4.github.io/pyproj/stable/gotchas.html#axis-order-changes-in-proj-6  
 in\_crs\_string = \_prepare\_from\_proj\_string(in\_crs\_string)

import pandas as pd  
vanishing\_gpd\_=pd.read\_pickle('./results/drive\_29\_0071\_vanishing\_gpd.pkl')  
vanishing\_gpd\_

|  | **start\_idx** | **end\_idx** | **geometry** | **length** |
| --- | --- | --- | --- | --- |
| **0** | 0 | 91 | LINESTRING (934871.288 6276329.185, 934817.770... | 11310.685843 |
| **1** | 1 | 81 | LINESTRING (934817.770 6276350.782, 934698.648... | 9900.262438 |
| **2** | 2 | 81 | LINESTRING (934698.648 6276416.118, 934758.123... | 9764.399529 |
| **3** | 3 | 81 | LINESTRING (934758.123 6276382.241, 934555.096... | 9695.952996 |
| **4** | 4 | 82 | LINESTRING (934555.096 6276474.385, 934582.633... | 9474.584042 |
| **...** | ... | ... | ... | ... |
| **1053** | 1053 | 1053 | GEOMETRYCOLLECTION EMPTY | 0.000000 |
| **1054** | 1054 | 1054 | GEOMETRYCOLLECTION EMPTY | 0.000000 |
| **1055** | 1055 | 1055 | GEOMETRYCOLLECTION EMPTY | 0.000000 |
| **1056** | 1056 | 1056 | GEOMETRYCOLLECTION EMPTY | 0.000000 |
| **1057** | 1057 | 1057 | GEOMETRYCOLLECTION EMPTY | 0.000000 |

1058 rows × 4 columns

id\_info\_df=id\_info\_df\_int.join(vanishing\_gpd\_)  
id\_info\_df

|  | **road** | **sidewalk** | **building** | **wall** | **fence** | **pole** | **traffic\_light** | **traffic\_sign** | **vegetation** | **terrain** | **...** | **vegetation\_percent** | **sky\_percent** | **confidence** | **lat** | **lon** | **trainID** | **start\_idx** | **end\_idx** | **geometry** | **length** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 84650 | 10430 | 319765 | 1696 | 0 | 1588 | 0 | 260 | 2589 | 1 | ... | 0.559165 | 1.908590 | 0.957152 | 49.008650 | 8.398092 | 0 | 0.0 | 91.0 | LINESTRING (934871.288 6276329.185, 934817.770... | 11310.685843 |
| **1** | 86454 | 9873 | 322688 | 2318 | 413 | 1229 | 0 | 84 | 1693 | 0 | ... | 0.365649 | 1.878137 | 0.957426 | 49.008777 | 8.397611 | 1 | 1.0 | 81.0 | LINESTRING (934817.770 6276350.782, 934698.648... | 9900.262438 |
| **2** | 87182 | 12023 | 325361 | 2264 | 65 | 595 | 0 | 175 | 1250 | 0 | ... | 0.269971 | 1.838829 | 0.957314 | 49.009162 | 8.396541 | 2 | 2.0 | 81.0 | LINESTRING (934698.648 6276416.118, 934758.123... | 9764.399529 |
| **3** | 89886 | 10992 | 319038 | 1831 | 9483 | 615 | 0 | 8 | 1454 | 0 | ... | 0.314031 | 1.809240 | 0.947994 | 49.008962 | 8.397075 | 3 | 3.0 | 81.0 | LINESTRING (934758.123 6276382.241, 934555.096... | 9695.952996 |
| **4** | 91569 | 14370 | 311801 | 1000 | 18342 | 1474 | 0 | 81 | 1430 | 0 | ... | 0.308847 | 1.780947 | 0.946398 | 49.009505 | 8.395251 | 4 | 4.0 | 82.0 | LINESTRING (934555.096 6276474.385, 934582.633... | 9474.584042 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1054** | 124321 | 13098 | 244613 | 11616 | 51167 | 1354 | 50 | 997 | 53 | 0 | ... | 0.011447 | 0.048163 | 0.942456 | 49.009215 | 8.396286 | 1054 | 1054.0 | 1054.0 | GEOMETRYCOLLECTION EMPTY | 0.000000 |
| **1055** | 126360 | 10678 | 241321 | 14036 | 50878 | 1703 | 81 | 993 | 4 | 0 | ... | 0.000864 | 0.057882 | 0.942314 | 49.009353 | 8.395764 | 1055 | 1055.0 | 1055.0 | GEOMETRYCOLLECTION EMPTY | 0.000000 |
| **1056** | 122516 | 14267 | 241194 | 14602 | 50789 | 1880 | 102 | 1000 | 0 | 0 | ... | 0.000000 | 0.031965 | 0.941350 | 49.008706 | 8.397888 | 1056 | 1056.0 | 1056.0 | GEOMETRYCOLLECTION EMPTY | 0.000000 |
| **1057** | 122223 | 14577 | 241842 | 14740 | 49996 | 1962 | 34 | 917 | 0 | 0 | ... | 0.000000 | 0.029589 | 0.941474 | 49.009215 | 8.396288 | 1057 | 1057.0 | 1057.0 | GEOMETRYCOLLECTION EMPTY | 0.000000 |
| **1058** | 121671 | 14120 | 241772 | 14341 | 51080 | 2141 | 166 | 864 | 0 | 0 | ... | 0.000000 | 0.021814 | 0.939658 | 49.009079 | 8.396812 | 1058 | NaN | NaN | None | NaN |

1059 rows × 30 columns

id\_info\_df.to\_pickle('./results/DUC\_info\_drive\_29\_0071\_vanishing\_gpd.pkl')

id\_info\_df\_=pd.read\_pickle('./results/DUC\_info\_drive\_29\_0071\_vanishing\_gpd.pkl')

计算信息熵和均衡度。计算的内容是图像中各个对象的频数或百分比。

def entroy\_df\_row(row,labels,id\_info\_df\_):  
 '''  
 function - 计算DataFrame每行的信息熵，用于df.apply(lambda)  
 '''  
 import numpy as np  
 import math  
   
 labels\_percent=row[labels].to\_numpy()\*1.000/id\_info\_df\_.iloc[[0]][["pixels"]].to\_numpy()  
 labels\_percent=labels\_percent[labels\_percent!=0]  
 entropy=-np.sum(labels\_percent\*np.log(labels\_percent.astype(np.float)))  
 max\_entropy=math.log(len(labels))  
 frank\_e=entropy/max\_entropy  
   
 return frank\_e  
id\_info\_df\_['equilibrium']=id\_info\_df\_.apply(lambda row :entroy\_df\_row(row,labels,id\_info\_df\_), axis=1 )   
id\_info\_df\_

C:\Users\richi\AppData\Local\Temp/ipykernel\_27732/1638474821.py:9: DeprecationWarning:  
  
`np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.  
Deprecated in NumPy 1.20; for more details and guidance: <https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations>

|  | **road** | **sidewalk** | **building** | **wall** | **fence** | **pole** | **traffic\_light** | **traffic\_sign** | **vegetation** | **terrain** | **...** | **sky\_percent** | **confidence** | **lat** | **lon** | **trainID** | **start\_idx** | **end\_idx** | **geometry** | **length** | **equilibrium** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 84650 | 10430 | 319765 | 1696 | 0 | 1588 | 0 | 260 | 2589 | 1 | ... | 1.908590 | 0.957152 | 49.008650 | 8.398092 | 0 | 0.0 | 91.0 | LINESTRING (934871.288 6276329.185, 934817.770... | 11310.685843 | 0.349989 |
| **1** | 86454 | 9873 | 322688 | 2318 | 413 | 1229 | 0 | 84 | 1693 | 0 | ... | 1.878137 | 0.957426 | 49.008777 | 8.397611 | 1 | 1.0 | 81.0 | LINESTRING (934817.770 6276350.782, 934698.648... | 9900.262438 | 0.345377 |
| **2** | 87182 | 12023 | 325361 | 2264 | 65 | 595 | 0 | 175 | 1250 | 0 | ... | 1.838829 | 0.957314 | 49.009162 | 8.396541 | 2 | 2.0 | 81.0 | LINESTRING (934698.648 6276416.118, 934758.123... | 9764.399529 | 0.336291 |
| **3** | 89886 | 10992 | 319038 | 1831 | 9483 | 615 | 0 | 8 | 1454 | 0 | ... | 1.809240 | 0.947994 | 49.008962 | 8.397075 | 3 | 3.0 | 81.0 | LINESTRING (934758.123 6276382.241, 934555.096... | 9695.952996 | 0.356748 |
| **4** | 91569 | 14370 | 311801 | 1000 | 18342 | 1474 | 0 | 81 | 1430 | 0 | ... | 1.780947 | 0.946398 | 49.009505 | 8.395251 | 4 | 4.0 | 82.0 | LINESTRING (934555.096 6276474.385, 934582.633... | 9474.584042 | 0.370141 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **1054** | 124321 | 13098 | 244613 | 11616 | 51167 | 1354 | 50 | 997 | 53 | 0 | ... | 0.048163 | 0.942456 | 49.009215 | 8.396286 | 1054 | 1054.0 | 1054.0 | GEOMETRYCOLLECTION EMPTY | 0.000000 | 0.439346 |
| **1055** | 126360 | 10678 | 241321 | 14036 | 50878 | 1703 | 81 | 993 | 4 | 0 | ... | 0.057882 | 0.942314 | 49.009353 | 8.395764 | 1055 | 1055.0 | 1055.0 | GEOMETRYCOLLECTION EMPTY | 0.000000 | 0.444203 |
| **1056** | 122516 | 14267 | 241194 | 14602 | 50789 | 1880 | 102 | 1000 | 0 | 0 | ... | 0.031965 | 0.941350 | 49.008706 | 8.397888 | 1056 | 1056.0 | 1056.0 | GEOMETRYCOLLECTION EMPTY | 0.000000 | 0.449985 |
| **1057** | 122223 | 14577 | 241842 | 14740 | 49996 | 1962 | 34 | 917 | 0 | 0 | ... | 0.029589 | 0.941474 | 49.009215 | 8.396288 | 1057 | 1057.0 | 1057.0 | GEOMETRYCOLLECTION EMPTY | 0.000000 | 0.450238 |
| **1058** | 121671 | 14120 | 241772 | 14341 | 51080 | 2141 | 166 | 864 | 0 | 0 | ... | 0.021814 | 0.939658 | 49.009079 | 8.396812 | 1058 | NaN | NaN | None | NaN | 0.450034 |

1059 rows × 31 columns

打印均衡度的曲线分布。

import plotly.express as px  
labels=list(trainID\_label\_mapping.values())  
fig = px.line(id\_info\_df\_, x="trainID", y='equilibrium',  
 hover\_data=['confidence','lat','lon','trainID'],  
 title='equilibrium'  
 )  
fig.show()



打印均衡度的空间核密度。

import plotly.express as px  
fig = px.density\_mapbox(id\_info\_df\_, lat='lat', lon='lon', z='equilibrium', radius=10,  
 center=dict(lat=49.008645, lon=8.398104), zoom=18,  
 mapbox\_style="stamen-terrain",  
 hover\_data=['confidence','lat','lon','trainID'],  
 title=r'equilibrium Kernel Density')  
fig.show()



计算视觉感知消失距离对应图像之间的均衡度变化。

id\_info\_df\_.dropna(inplace=True)  
id\_info\_df\_["VP\_equilibrium"]=id\_info\_df\_.apply(lambda row: id\_info\_df\_.iloc[[int(row.end\_idx)]].equilibrium.values[0]-row.equilibrium,axis=1)  
  
import plotly.express as px  
fig = px.density\_mapbox(id\_info\_df\_, lat='lat', lon='lon', z='VP\_equilibrium', radius=10,  
 center=dict(lat=49.008645, lon=8.398104), zoom=18,  
 mapbox\_style="stamen-terrain",  
 hover\_data=['confidence','lat','lon','trainID'],  
 title=r'VP\_equilibrium Kernel Density')  
fig.show()



### 附： 知识点

**知识点-01：**super().\_\_init\_\_()——继承父类的init方法

通过下述实例理解继承父类的方法，对于子类Child\_robin虽然继承了父类，可以调用父类的方法（函数），但是因为子类自身的\_\_init\_\_初始化，覆盖了父类的属性，因此无法调用父类属性。对于子类Child\_sparrow，增加了super().\_\_init\_\_()方法，从而可以调用父类属性。

class Parent:  
 def \_\_init\_\_(self,name="bird"):  
 self.name=name  
   
class Child\_robin(Parent):  
 def \_\_init\_\_(self,species="robin"):  
 self.species=species  
   
class Child\_sparrow(Parent):  
 def \_\_init\_\_(self,species="sparrow"):  
 self.species=species  
 super(Child\_sparrow,self).\_\_init\_\_()  
   
p=Parent()  
print("获取Parent类属性：name=%s"%p.name)  
print("\_"\*50)  
c\_r=Child\_robin()  
print("获取子类Child\_robin的属性：species=%s"%c\_r.species)  
try:  
 print("获取父类Parent属性：name=%s"%c\_r.name)  
except AttributeError as error:  
 print(error)  
print("\_"\*50)  
c\_s=Child\_sparrow()  
print("获取子类Child\_sparrow的属性：species=%s"%c\_s.species)  
print("获取父类Parent属性：name=%s"%c\_s.name)

获取Parent类属性：name=bird  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
获取子类Child\_robin的属性：species=robin  
'Child\_robin' object has no attribute 'name'  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
获取子类Child\_sparrow的属性：species=sparrow  
获取父类Parent属性：name=bird

**知识点-02：**\_\_getattr\_\_,\_\_setattr\_\_,\_\_delattr\_\_

类Class\_A的实例（instance）C\_A，通过C\_A.attri\_a访问实例属性attri\_a（对象变量），并返回属性对应值；通过实例的C\_A.\_\_dict\_\_可以查看所有实例的属性（即实例的属性存储在\_\_dict\_\_中）。如果预提取实例中不存在的属性，则会调用\_\_getattri\_\_。如果类的变量（属性）定义在初始化函数外部，例如attri\_c（类变量），则实例的C\_A.\_\_dict\_\_并不包含该属性，但是在类自身的Class\_A.\_\_dict\_\_对象中包含该属性。

在实例初始化、重新赋值，及增加新的属性时均会自动调用\_\_setattr\_\_方法，并用self.\_\_dict\_\_[name]=value语句，把属性键值对保存在\_\_dict\_\_对象中。其中\_\_setattr\_\_(self,name,value)的参数name和value为固定参数，代表属性的键值对。

如果要删除实例的属性键值对，则可以执行del C\_A.attri\_d，调用\_\_delattr\_\_，用del self.\_\_dict\_\_[name]方法删除属性键值对。

class Class\_A:  
 attri\_c="attri\_C"  
 def \_\_init\_\_(self,attri\_a,attri\_b):  
 self.attri\_a=attri\_a  
 self.attri\_b=attri\_b  
   
 def \_\_getattr\_\_(self,attri):  
 return('invoke \_\_getattr\_\_',attri)  
   
 def \_\_setattr\_\_(self,name,value):  
 print("invoke \_\_setattr\_\_",)  
 self.\_\_dict\_\_[name]=value  
   
 def \_\_delattr\_\_(self,name):  
 print("invoke \_\_delattr\_\_",)  
 print("deleting `{}`".format(str(name)))  
 try:  
 del self.\_\_dict\_\_[name]  
 print ("`{}` deleted".format(str(name)))  
 except KeyError as k:  
 return None  
   
   
C\_A=Class\_A("attri\_A","attri\_B")  
print("实例C\_A包含的属性及其值：",C\_A.\_\_dict\_\_)  
print("实例C\_A已有属性attri\_a，则直接返回该属性对应值，不会调用\_\_getattr\_\_，attri\_a=",C\_A.attri\_a)  
print("实例C\_A没有属性attri\_none,则调用\_\_getattr\_\_：",C\_A.attri\_none)  
print(Class\_A.\_\_dict\_\_)  
print("用类的实例C\_A提取属性attri\_c=%s"%C\_A.attri\_c,";" "用类自身Class\_A直接提取属性attri\_c=%s"%Class\_A.attri\_c)  
print("\_"\*50)  
C\_A.attri\_b="attri\_B\_assignment"  
print("对已有属性重新赋值：",C\_A.\_\_dict\_\_)  
C\_A.attri\_d="attri\_D"  
print("增加新的属性，并赋值：",C\_A.\_\_dict\_\_)

invoke \_\_setattr\_\_  
invoke \_\_setattr\_\_  
实例C\_A包含的属性及其值： {'attri\_a': 'attri\_A', 'attri\_b': 'attri\_B'}  
实例C\_A已有属性attri\_a，则直接返回该属性对应值，不会调用\_\_getattr\_\_，attri\_a= attri\_A  
实例C\_A没有属性attri\_none,则调用\_\_getattr\_\_： ('invoke \_\_getattr\_\_', 'attri\_none')  
{'\_\_module\_\_': '\_\_main\_\_', 'attri\_c': 'attri\_C', '\_\_init\_\_': <function Class\_A.\_\_init\_\_ at 0x0000020817835820>, '\_\_getattr\_\_': <function Class\_A.\_\_getattr\_\_ at 0x0000020817835790>, '\_\_setattr\_\_': <function Class\_A.\_\_setattr\_\_ at 0x0000020817835700>, '\_\_delattr\_\_': <function Class\_A.\_\_delattr\_\_ at 0x0000020817835670>, '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'Class\_A' objects>, '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'Class\_A' objects>, '\_\_doc\_\_': None}  
用类的实例C\_A提取属性attri\_c=attri\_C ;用类自身Class\_A直接提取属性attri\_c=attri\_C  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
invoke \_\_setattr\_\_  
对已有属性重新赋值： {'attri\_a': 'attri\_A', 'attri\_b': 'attri\_B\_assignment'}  
invoke \_\_setattr\_\_  
增加新的属性，并赋值： {'attri\_a': 'attri\_A', 'attri\_b': 'attri\_B\_assignment', 'attri\_d': 'attri\_D'}

print("\_"\*50)  
del C\_A.attri\_d  
print("删除属性attri\_d：",C\_A.\_\_dict\_\_)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
invoke \_\_delattr\_\_  
deleting `attri\_d`  
`attri\_d` deleted  
删除属性attri\_d： {'attri\_a': 'attri\_A', 'attri\_b': 'attri\_B\_assignment'}

**知识点-03：** mutable（可变）与immutable（不可变）

Python的数据类型分为mutable（可变）与immutable（不可变），mutable就是创建后可以修改，而immutable是创建后不可修改。

对于mutable，下述代码定义了变量a，并将变量b指向了变量a，因此a和b指向同一对象；但是当变量b执行运算后，则变量b指向新的对象（地址）。同样定义列表lst\_a，并将列表lst\_b指向列表lst\_a，则lst\_a和lst\_b指向同一对象。即使二者分别追加新的值，仍然指向同一对象。但是，重新定义变量lst\_b为新的列表，则lst\_b指向新的对象。

a=0  
b=a  
print("a,b 是否指向同一个对象：id\_a={};id\_b={}".format(id(a),id(b)),id(a)==id(b))  
  
b+=1  
print("b执行运算后，a,b 是否指向同一个对象：",id(a)==id(b))

a,b 是否指向同一个对象：id\_a=140716884631264;id\_b=140716884631264 True  
b执行运算后，a,b 是否指向同一个对象： False

lst\_a=[0]  
lst\_b=lst\_a  
print("列表lst\_a和lst\_b是否指向同一个对象：",id(lst\_a)==id(lst\_b))  
lst\_b.append(99)  
print("lit\_b追加值后,列表lst\_a和lst\_b是否指向同一个对象：",id(lst\_a)==id(lst\_b))  
lst\_a.append(79)  
print("lit\_a追加值后,列表lst\_a和lst\_b是否指向同一个对象：",id(lst\_a)==id(lst\_b))  
lst\_b=[0]  
print("lit\_b定义新的列表,列表lst\_a和lst\_b是否指向同一个对象：",id(lst\_a)==id(lst\_b))

列表lst\_a和lst\_b是否指向同一个对象： True  
lit\_b追加值后,列表lst\_a和lst\_b是否指向同一个对象： True  
lit\_a追加值后,列表lst\_a和lst\_b是否指向同一个对象： True  
lit\_b定义新的列表,列表lst\_a和lst\_b是否指向同一个对象： False

对于immutable，因为自定义的python类型一般都是mutable，如果实现immutable数据类型，通常需要重写对象(object)的\_\_setattr\_\_和\_\_delattr\_\_方法。例如下述重新定义了\_\_setattr\_\_，并不会将待增加或修改的属性写入\_\_dict\_\_中，而是直接引起TypeError异常。为保证不能删除类实例对象，令\_\_delattr\_\_ = \_\_setattr\_\_。因此待类immutable实例化为cls对象，修改删除和增加属性值都会引发异常。

class immutable:  
 def \_\_setattr\_\_(self, \*args):   
 print("invoke \_\_setattr\_\_")  
 raise TypeError("cannot modify the value of immutable instance")  
 \_\_delattr\_\_ = \_\_setattr\_\_  
 def \_\_init\_\_(self,name,value):  
 super(immutable,self).\_\_setattr\_\_(name,value)   
cls=immutable("attri\_e","attri\_E")  
print("实例初始化属性值，并读取 attri\_e=%s"%cls.attri\_e)  
cls.attri\_e="attri\_new"

实例初始化属性值，并读取 attri\_e=attri\_E  
invoke \_\_setattr\_\_  
  
  
  
---------------------------------------------------------------------------  
  
TypeError Traceback (most recent call last)  
  
~\AppData\Local\Temp/ipykernel\_21220/902156601.py in <module>  
 8 cls=immutable("attri\_e","attri\_E")  
 9 print("实例初始化属性值，并读取 attri\_e=%s"%cls.attri\_e)  
---> 10 cls.attri\_e="attri\_new"  
  
  
~\AppData\Local\Temp/ipykernel\_21220/902156601.py in \_\_setattr\_\_(self, \*args)  
 2 def \_\_setattr\_\_(self, \*args):  
 3 print("invoke \_\_setattr\_\_")  
----> 4 raise TypeError("cannot modify the value of immutable instance")  
 5 \_\_delattr\_\_ = \_\_setattr\_\_  
 6 def \_\_init\_\_(self,name,value):  
  
  
TypeError: cannot modify the value of immutable instance

del cls.attri\_e

invoke \_\_setattr\_\_  
  
  
  
---------------------------------------------------------------------------  
  
TypeError Traceback (most recent call last)  
  
~\AppData\Local\Temp/ipykernel\_21220/3598793624.py in <module>  
----> 1 del cls.attri\_e  
  
  
~\AppData\Local\Temp/ipykernel\_21220/902156601.py in \_\_setattr\_\_(self, \*args)  
 2 def \_\_setattr\_\_(self, \*args):  
 3 print("invoke \_\_setattr\_\_")  
----> 4 raise TypeError("cannot modify the value of immutable instance")  
 5 \_\_delattr\_\_ = \_\_setattr\_\_  
 6 def \_\_init\_\_(self,name,value):  
  
  
TypeError: cannot modify the value of immutable instance

cls.attri\_f="attri\_F"

invoke \_\_setattr\_\_  
  
  
  
---------------------------------------------------------------------------  
  
TypeError Traceback (most recent call last)  
  
~\AppData\Local\Temp/ipykernel\_21220/3100895739.py in <module>  
----> 1 cls.attri\_f="attri\_F"  
  
  
~\AppData\Local\Temp/ipykernel\_21220/902156601.py in \_\_setattr\_\_(self, \*args)  
 2 def \_\_setattr\_\_(self, \*args):  
 3 print("invoke \_\_setattr\_\_")  
----> 4 raise TypeError("cannot modify the value of immutable instance")  
 5 \_\_delattr\_\_ = \_\_setattr\_\_  
 6 def \_\_init\_\_(self,name,value):  
  
  
TypeError: cannot modify the value of immutable instance

**知识点-04：** \_variable、\_\_variable和\_\_variable\_\_

Python中成员函数和变量都是公开的public，在Python中没有public和private方法修饰成员函数或变量。虽然没有支持私有化（priviate），但是可以应用下划线的方法限制成员函数和成员变量的访问权限（尽力避免定义以下划线开头的变量）。\_variable单下划线开始的成员变量叫做包含变量，只有类的实例和子类的实例能访问这些变量，并需要通过类的接口访问，不能用from module imort \*的方法导入。\_\_variable双下划线开始的成员变量为私有成员，只有类对象自己能访问，子类对象不能访问。\_\_variable\_\_前后双下划线，为Python特殊方法专用的标识，例如\_\_init\_\_()类的构造函数。

class private:  
 def \_\_init\_\_(self):  
 self.attri='attri public'  
 self.\_attri='attri\_singleUnderscore'  
 self.\_\_attri='attri\_\_doubleUnderscore'  
   
 def func(self):  
 return self.attri+' func'  
 def \_func(self):  
 return self.\_attri+' \_func'  
 def \_\_func(self):  
 return self.\_\_attri+' \_\_func'  
 def invoke\_\_func(self):  
 return self.\_\_func()  
   
class private\_Child(private):  
 def \_\_init\_\_(self):  
 self.attri\_Child='attri child'  
 super(private\_Child,self).\_\_init\_\_()  
   
p=private()  
print("类属性——公有成员：attri=%s"%p.attri)  
print("类属性——包含变量(单下划线):\_attri=%s"%p.\_attri)  
try:  
 print("类属性——私有变量（双下划线）:\_\_attri=%s"%p.\_\_attri)  
except AttributeError as error:  
 print(error)  
print("\_"\*50)   
print("类方法——公有成员：func=",p.func())  
print("类方法——单下划线：\_func=",p.\_func())  
try:  
 print("类方法——双下划线：\_\_func=",p.\_\_func())  
except:  
 print("没有类方法：\_\_func")  
print("\_"\*50)  
p\_child=private\_Child()  
print("子类调用父类单下划线方法：",p\_child.\_func())  
try:  
 print("子类调用父类双下划线方法：",p\_child.\_\_func())  
except:  
 print("子类没有父类方法：\_\_func()")

类属性——公有成员：attri=attri public  
类属性——包含变量(单下划线):\_attri=attri\_singleUnderscore  
'private' object has no attribute '\_\_attri'  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
类方法——公有成员：func= attri public func  
类方法——单下划线：\_func= attri\_singleUnderscore \_func  
没有类方法：\_\_func  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
子类调用父类单下划线方法： attri\_singleUnderscore \_func  
子类没有父类方法：\_\_func()

**知识点-05：**[collections](https://docs.python.org/3/library/collections.html).namedtuple⑭

collections.namedtuple(typename, field\_names, \*, verbose=False, rename=False, module=None)，其中参数typename为创建的一个元组子类类名，用于实例化各种元组对象；field\_names类似于字典的键（key），通过键提取对应的值（value）；rename默认为False，如果为True，则不能包含有“非Python标识符，Python中的关键字以及重复的name”；如果有，则会默认重命名。

from collections import namedtuple  
# 01-实例化nametuple对象  
Point = namedtuple('Point', ['x', 'y'])  
# 02-使用关键字参数或位置参数初始化nametuple  
p = Point(11, y=22)  
print("02-使用关键字参数或位置参数初始化nametuple:p={}".format(p))  
# 03-使用键提取元组元素  
print("03-使用键提取元组元素:p[0]={},p[1]={}".format(p[0],p[1]))  
# 04-拆包  
x,y=p  
print("04-拆包：x,y=p——>x={},y={}".format(x,y))  
# 05-instance.key的方式提取值  
print("05-instance.key的方式提取值:p.x={},p.y={}".format(p.x,p.y))  
# 06-用已有序列或可迭代对象实例化一个nametuple  
lst=[99,77]  
print("06-用已有序列或可迭代对象实例化一个nametuple:Point.\_make(lst)={}".format(Point.\_make(lst)))  
# 07-将nametuple对象转换为有序字典OrderDict  
print("07-将nametuple对象转换为有序字典OrderDict:p.\_asdict()={}".format(p.\_asdict()))  
# 08-有序字典转换为nametuple对象  
dic={'x': 11, 'y': 22}  
print("08-有序字典转换为nametuple对象:Point(\*\*dic)={}".format(Point(\*\*dic)))  
# 09-替换值  
print("09-替换值:p.\_replace(x=33)——>{}".format(p.\_replace(x=33)))  
# 10-获取所有nametuple对象字段名  
print("10-获取所有nametuple对象字段名:p.\_fields={}".format(p.\_fields))

02-使用关键字参数或位置参数初始化nametuple:p=Point(x=11, y=22)  
03-使用键提取元组元素:p[0]=11,p[1]=22  
04-拆包：x,y=p——>x=11,y=22  
05-instance.key的方式提取值:p.x=11,p.y=22  
06-用已有序列或可迭代对象实例化一个nametuple:Point.\_make(lst)=Point(x=99, y=77)  
07-将nametuple对象转换为有序字典OrderDict:p.\_asdict()={'x': 11, 'y': 22}  
08-有序字典转换为nametuple对象:Point(\*\*dic)=Point(x=11, y=22)  
09-替换值:p.\_replace(x=33)——>Point(x=33, y=22)  
10-获取所有nametuple对象字段名:p.\_fields=('x', 'y')

**知识点-06：** [importlib](https://docs.python.org/3/library/importlib.html)⑮ 与getattr

Python标准库importlib，可以导入自定义的对象（.py文件/模块），并支持传入字符串导入模块。首先定义了“importlib\_func\_A.py”文件，将其置于“datasets”文件夹（包）下，使用importlib库，读取模块，并应用模块的基本操作，读取模块中的变量值、类的属性和方法。同时可以应用importlib.util.find\_spec查看是否存在模块等。读取模块的属性使用getattr方法。

# datasets/importlib\_func\_A.py  
# -\*- coding: utf-8 -\*-  
  
attri\_f="attri\_F"  
  
def func\_A():  
 print("importlib\_func\_A/func\_A")  
  
class cls\_A:  
 attri\_g="attri\_G"  
   
 def func\_B():  
 print("importlib\_func\_A/cls\_A/func\_B")  
   
if \_\_name\_\_=="\_\_main\_\_":  
 func\_A()

importlib\_func\_A/func\_A

args.impoftlib\_module="importlib\_func\_A"  
  
def dynamic\_import(package,module):  
 import importlib  
 '''  
 function - 应用importlib调入自定义模块  
 '''   
 return importlib.import\_module('{}.{}'.format(package,module))  
  
module=dynamic\_import("dataset",args.impoftlib\_module)  
print("imported module:",module)  
print("调入模块中的变量值：",getattr(module,"attri\_f"))  
cls\_A=getattr(module,'cls\_A')  
print("调入模块中的类：",cls\_A)  
print("调入模块中类的属性：",cls\_A.attri\_g)  
print("\_"\*50)  
print("调入模块中类的方法：")  
cls\_A.func\_B()

imported module: <module 'dataset.importlib\_func\_A' from 'C:\\Users\\richi\\omen\_richiebao\\omen\_github\\USDA\_CH\_final\\USDA\\notebook\\dataset\\importlib\_func\_A.py'>  
调入模块中的变量值： attri\_F  
调入模块中的类： <class 'dataset.importlib\_func\_A.cls\_A'>  
调入模块中类的属性： attri\_G  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
调入模块中类的方法：  
importlib\_func\_A/cls\_A/func\_B

def check\_module(package,module):   
 '''  
 function - 应用importlib查看是否存在模块  
 '''  
 import importlib  
  
 module\_spec=importlib.util.find\_spec('{}.{}'.format(package,module))  
 if module\_spec is None:  
 print("Module:{} not found.".format('{}.{}'.format(package,module)))  
 return None  
 else:  
 print("Module:{} can be imported.".format('{}.{}'.format(package,module)))  
 return module\_spec  
check\_module('dataset',args.impoftlib\_module)

Module:dataset.importlib\_func\_A can be imported.  
  
ModuleSpec(name='dataset.importlib\_func\_A', loader=<\_frozen\_importlib\_external.SourceFileLoader object at 0x0000020817914CA0>, origin='C:\\Users\\richi\\omen\_richiebao\\omen\_github\\USDA\_CH\_final\\USDA\\notebook\\dataset\\importlib\_func\_A.py')

注释（Notes）：

① Cityscapes数据集，（<https://www.cityscapes-dataset.com>）。

② Cityscapes Dataset Overview, <https://www.cityscapes-dataset.com/dataset-overview/#features>.

③ Cityscape数据下载注册页，（<https://www.cityscapes-dataset.com/login>）。

④ argparse，用来解析命令行参数的 Python 库，使编写用户友好的命令行界面变得容易。可以从所定义的参数中解析这些参数，并会自动生成帮助和用法信息；当提供的参数值无效时，引发错误提示，（<https://docs.python.org/3/library/argparse.html>）。

⑤ cityscapesScripts，该库包含用于检查、准备和评估 Cityscapes 数据集的模块，（<https://github.com/mcordts/cityscapesScripts>）。

⑥ 开放神经网络交换——ONNX，帮助创建和部署深度学习模型，（<https://onnx.ai/supported-tools.html>）。

⑦ torchvision.models，预定义深度学习模型，（<https://pytorch.org/vision/0.8/models.html>）。

⑧ ONNX Model Zoo，为预训练的深度学习模型库，每个模型都有基于Python编写训练和运行交互解释的Jupyter格式文件，（<https://github.com/onnx/models>）。

⑨ Netron，网络可视化工具，（<https://github.com/lutzroeder/Netron>）。

⑩ DUC，基于 CNN 的语义分割模型，（<https://github.com/onnx/models/tree/master/vision/object_detection_segmentation/duc>）。

⑪ Inference demo for DUC models，（<https://github.com/onnx/models/blob/master/vision/object_detection_segmentation/duc/dependencies/duc-inference.ipynb>）。

⑫ MXNet，开源深度学习框架，（<https://mxnet.apache.org/>）。

⑬ KITTI数据集，（<https://www.cvlibs.net/datasets/kitti>）。

⑭ collections， 专门的数据类型，为常规的dict、list、set和tuple数据结构提供了可替代方案，（<https://docs.python.org/3/library/collections.html>）。

⑮ importlib，（<https://docs.python.org/3/library/importlib.html>）。

参考文献（References）:

[1] Cityscapes Dataset Overview, <https://www.cityscapes-dataset.com/dataset-overview/>.

[2] Tao, A., Sapra, K. & Catanzaro, B. Hierarchical Multi-Scale Attention for Semantic Segmentation. (2020).https://arxiv.org/abs/2005.10821; https://github.com/NVIDIA/semantic-segmentation