**实验（实习）报告**

1. 实验目的

掌握图像分割任务难点，了解如何使用深度学习解决相关问题。掌握不同图像分割神经网络架构的设计原理与核心思想，熟悉使用MindSpore深度学习框架实现深度学习实验的一般流程。

1. 实验任务
   1. 基于U-Net的医学图像分割实验，使用MindSpore，实现基于U-Net模型的医学图像分割实验。
   2. 基于DeepLabv3的语义分割实验，使用MindSpore实现基于微调DeepLabv3模型的语义分割实验。

（详见使用MindSpore实现基于微调DeepLabv3模型的语义分割实验。）

1. 实验步骤

1 基于U-Net的医学图像分割实验

* 1. 下载数据集

import moxing as mox

mox.file.copy\_parallel(src\_url="obs://ascend-professional-construction-dataset/ComputerVision/Unet",dst\_url="./")

* 1. 导入实验环境

import os

import argparse

import ast

import numpy as np

import cv2

import mindspore

import mindspore.nn as nn

import mindspore.ops.operations as F

from mindspore import Model, context

from mindspore.nn.loss.loss import \_Loss

from mindspore.communication.management import init, get\_group\_size

from mindspore.train.callback import CheckpointConfig, ModelCheckpoint

from mindspore.context import ParallelMode

from mindspore.train.serialization import load\_checkpoint, load\_param\_into\_net

from mindspore.common.initializer import TruncatedNormal

from mindspore.nn import CentralCrop

from PIL import Image, ImageSequence

import mindspore.dataset as ds

import mindspore.dataset.vision.c\_transforms as c\_vision

from mindspore.dataset.vision.utils import Inter

from mindspore.communication.management import get\_rank, get\_group\_size

from collections import deque

import time

from mindspore.train.callback import Callback

from mindspore.common.tensor import Tensor

from scipy.special import softmax

from matplotlib import pyplot as plt

device\_id = 2

context.set\_context(mode=context.GRAPH\_MODE, device\_target="Ascend", save\_graphs=False)

mindspore.set\_seed(1)

* 1. 查看数据集

步骤 1数据集说明

训练和测试数据集为两组30节果蝇一龄幼虫腹神经索（VNC）的连续透射电子显微镜（ssTEM）数据集。微立方体的尺寸约为2 x 2 x 1.5微米，分辨率为4x4x50纳米/像素。数据集大小为22.5 MB，共三个文件：train-volume.tif，train-labels.tif，test-volume.tif。第一个文件为训练集图像，该TIF文件共有30个通道，每个通道为一张灰度图像。第二个文件为训练集标签，该TIF文件共有30个通道，每个通道为一张灰度图像（像素值仅为0或255）。第三个文件为测试集图像，该TIF文件同样有30个通道，每个通道为一张灰度图像。

步骤 2展示数据集图像和标签

#打印图像和标签的形状，并展示第一张图像和标签

image= np.array([np.array(p) for p in ImageSequence.Iterator(Image.open("./data/train-volume.tif"))])

label= np.array([np.array(p) for p in ImageSequence.Iterator(Image.open("./data/train-labels.tif"))])

print(image.shape)

print(label.shape)

plt.figure(figsize=(10,10))

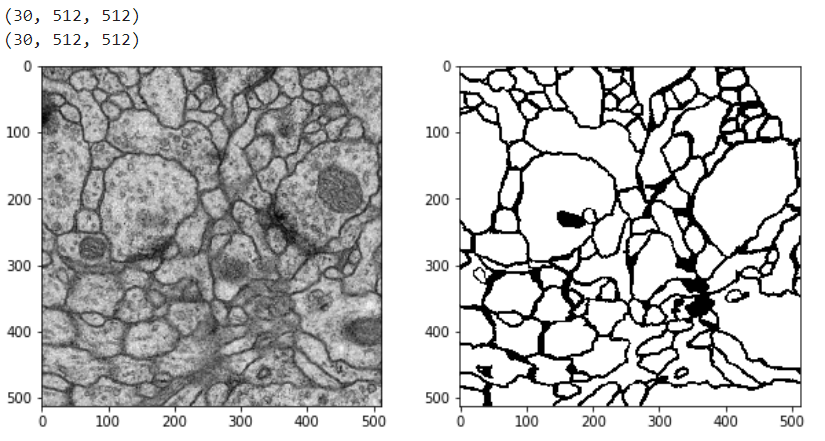
plt.subplot(2,2,1)

plt.imshow(image[0],cmap='gray')

plt.subplot(2,2,2)

plt.imshow(label[0],cmap='gray')

plt.show()



* 1. 使用大津阈值法进行图像分割
     1. OTSU算法介绍：

大津阈值法（OTSU）,又称作最大类间方差法，是一种图像二值化分割阈值的算法，由日本学者大津于1979年提出。大津阈值法是具有统计意义上的最佳分割阈值。其核心思想就是使类间方差最大，按照大津阈值法求得的阈值进行图像二值分割以区分前后背景，前景与背景图像的类间方差最大。该算法要求被分割的物体颜色纹理比较紧凑，类内方差小，对于一些文本图像的处理（比如车牌、指纹）效果很好。

大津阈值法是基于统计直方图的图像分割算法，首先我们绘制观察图像直方图。

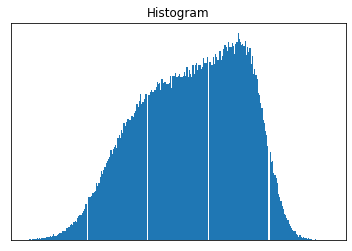
* + 1. 请用OpenCV和matplotlib绘制一张训练集中的直方图。（初级）

plt.hist(image[0].ravel(), 256)

plt.title("Histogram")

plt.xticks([])

plt.yticks([])



* + 1. 请用OpenCV中的python接口实现基于大津阈值法的图像二值化分割。（初级）

# 二值化处理，thesh=0代表其从0开始扫描

ret1, th1 = cv2.threshold(src=image[0], thresh=0,

                          maxval=255, type=cv2.THRESH\_OTSU)

plt.figure(figsize=(10,10))

plt.subplot(121)

plt.imshow(image[0],cmap='gray')

plt.title("source image")

plt.xticks([])

plt.yticks([])

plt.subplot(122)

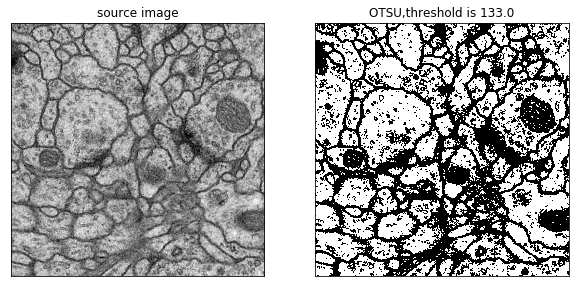
plt.imshow(th1, "gray")

plt.title("OTSU,threshold is " + str(ret1))

plt.xticks([])

plt.yticks([])

plt.show()



* 1. 基于神经网络的图像分割算法
     1. 定义Unet网络结构

U-Net得名于它的网络结构图，类似一个英文字母“U”。左半边是一个从上到下，一步一步从原始图像抽取特征（即原始图像本质信息）的过程；右半边是一个从下到上，一步一步从图像特征还原目标信息的过程。U-Net网络主要分为encoder和decoder两个部分。

* + 1. 编码

第一部分网络与普通卷积网络相同，通过叠加2个3x3卷积和最大池化的模块，获取多尺度特征图以抓住图像中的上下文信息（也即像素间的关系）。网络中特征图的尺寸共压缩了4次。

卷积层：原始U-Net网络中卷积层统一为3x3的卷积核，padding为0 ，striding为1。没有padding所以每次卷积之后feature map的尺寸变小。因此在copy and crop时要注意feature map的尺寸。且我们会发现输入的图像尺寸为572x572，输出尺寸为388x388，输入的尺寸是要大于输出的尺寸的。

池化层：两次卷积之后是一个窗口尺寸为2，步长为2的max pooling，输出大小变为原来的二分之一，池化的主要作用为减小特征图尺寸。

* + 1. 解码

第二部分与前面基本对称，共4个上采样模块，每个模块有2个3x3卷积和转置卷积，3x3的卷积核同样采用padding为0 ，striding为1，转置卷积考虑到棋盘效应（https://distill.pub/2016/deconv-checkerboard/），采用的是窗口大小为2，步长为2的转置卷积。每上采样一次，通过copy and crop，即skip-connection操作与encoder中对应的特征图进行拼接。网络最后输出时利用1\*1卷积调整特征图输出数量。该过程除了卷积比较关键的步骤就是转置卷积与skip-connection。

上采样的目的是恢复图像分辨率，上采样采用的方式有简单像素重复、转置卷积和差值（最邻近差值和双线性差值法等）。

* + 1. 超参数配置

cfg\_unet = {

    'name': 'Unet',

    'lr': 0.0001,

    'epochs': 400,

    'distribute\_epochs': 1600,

    'batchsize': 16,

    'cross\_valid\_ind': 1,

    'num\_classes': 2,

    'num\_channels': 1,

    'keep\_checkpoint\_max': 10,

    'weight\_decay': 0.0005,

    'loss\_scale': 1024.0,

    'FixedLossScaleManager': 1024.0,

    'resume': False,

    'resume\_ckpt': './',

}

* + 1. 各个模块搭建

class DoubleConv(nn.Cell):

    def \_\_init\_\_(self, in\_channels, out\_channels, mid\_channels=None):

        super().\_\_init\_\_()

        init\_value\_0 = TruncatedNormal(0.06)

        init\_value\_1 = TruncatedNormal(0.06)

        if not mid\_channels:

            mid\_channels = out\_channels

        self.double\_conv = nn.SequentialCell(

            [nn.Conv2d(in\_channels, mid\_channels, kernel\_size=3, has\_bias=True,

                       weight\_init=init\_value\_0, pad\_mode="valid"),

             nn.ReLU(),

             nn.Conv2d(mid\_channels, out\_channels, kernel\_size=3, has\_bias=True,

                       weight\_init=init\_value\_1, pad\_mode="valid"),

             nn.ReLU()]

        )

    def construct(self, x):

        return self.double\_conv(x)

class Down(nn.Cell):

    """Downscaling with maxpool then double conv"""

    def \_\_init\_\_(self, in\_channels, out\_channels):

        super().\_\_init\_\_()

        self.maxpool\_conv = nn.SequentialCell(

            [nn.MaxPool2d(kernel\_size=2, stride=2),

             DoubleConv(in\_channels, out\_channels)]

        )

    def construct(self, x):

        return self.maxpool\_conv(x)

class Up1(nn.Cell):

    """Upscaling then double conv"""

    def \_\_init\_\_(self, in\_channels, out\_channels, bilinear=True):

        super().\_\_init\_\_()

        self.concat = F.Concat(axis=1)

        self.factor = 56.0 / 64.0

        self.center\_crop = CentralCrop(central\_fraction=self.factor)

        self.print\_fn = F.Print()

        self.conv = DoubleConv(in\_channels, out\_channels, in\_channels // 2)

        self.up = nn.Conv2dTranspose(in\_channels, in\_channels // 2, kernel\_size=2, stride=2)

        self.relu = nn.ReLU()

    def construct(self, x1, x2):

        x1 = self.up(x1)

        x1 = self.relu(x1)

        x2 = self.center\_crop(x2)

        x = self.concat((x1, x2))

        return self.conv(x)

class Up2(nn.Cell):

    """Upscaling then double conv"""

    def \_\_init\_\_(self, in\_channels, out\_channels, bilinear=True):

        super().\_\_init\_\_()

        self.concat = F.Concat(axis=1)

        self.factor = 104.0 / 136.0

        self.center\_crop = CentralCrop(central\_fraction=self.factor)

        self.conv = DoubleConv(in\_channels, out\_channels, in\_channels // 2)

        self.up = nn.Conv2dTranspose(in\_channels, in\_channels // 2, kernel\_size=2, stride=2)

        self.relu = nn.ReLU()

    def construct(self, x1, x2):

        x1 = self.up(x1)

        x1 = self.relu(x1)

        x2 = self.center\_crop(x2)

        x = self.concat((x1, x2))

        return self.conv(x)

class Up3(nn.Cell):

    """Upscaling then double conv"""

    def \_\_init\_\_(self, in\_channels, out\_channels, bilinear=True):

        super().\_\_init\_\_()

        self.concat = F.Concat(axis=1)

        self.factor = 200 / 280

        self.center\_crop = CentralCrop(central\_fraction=self.factor)

        self.print\_fn = F.Print()

        self.conv = DoubleConv(in\_channels, out\_channels, in\_channels // 2)

        self.up = nn.Conv2dTranspose(in\_channels, in\_channels // 2, kernel\_size=2, stride=2)

        self.relu = nn.ReLU()

    def construct(self, x1, x2):

        x1 = self.up(x1)

        x1 = self.relu(x1)

        x2 = self.center\_crop(x2)

        x = self.concat((x1, x2))

        return self.conv(x)

class Up4(nn.Cell):

    """Upscaling then double conv"""

    def \_\_init\_\_(self, in\_channels, out\_channels, bilinear=True):

        super().\_\_init\_\_()

        self.concat = F.Concat(axis=1)

        self.factor = 392 / 568

        self.center\_crop = CentralCrop(central\_fraction=self.factor)

        self.conv = DoubleConv(in\_channels, out\_channels, in\_channels // 2)

        self.up = nn.Conv2dTranspose(in\_channels, in\_channels // 2, kernel\_size=2, stride=2)

        self.relu = nn.ReLU()

    def construct(self, x1, x2):

        x1 = self.up(x1)

        x1 = self.relu(x1)

        x2 = self.center\_crop(x2)

        x = self.concat((x1, x2))

        return self.conv(x)

class OutConv(nn.Cell):

    def \_\_init\_\_(self, in\_channels, out\_channels):

        super(OutConv, self).\_\_init\_\_()

        init\_value = TruncatedNormal(0.06)

        self.conv = nn.Conv2d(in\_channels, out\_channels, kernel\_size=1, has\_bias=True, weight\_init=init\_value)

    def construct(self, x):

        x = self.conv(x)

        return x

* + 1. 损失函数

请重新构建一个类，将MindSpore的nn.SoftmaxCrossEntropyWithLogits损失函数用于Unet，计算输出特征图各个位置平均的损失值。

class CrossEntropyWithLogits(\_Loss):

    #重写损失函数。

    def \_\_init\_\_(self):

        super(CrossEntropyWithLogits, self).\_\_init\_\_()

        self.transpose\_fn = F.Transpose()

        self.reshape\_fn = F.Reshape()

        self.softmax\_cross\_entropy\_loss = nn.SoftmaxCrossEntropyWithLogits()

        self.cast = F.Cast()

    def construct(self, logits, label):

        # NCHW->NHWC

        logits = self.transpose\_fn(logits, (0, 2, 3, 1))

        logits = self.cast(logits, mindspore.float32)

        label = self.transpose\_fn(label, (0, 2, 3, 1))

        loss = self.reduce\_mean(self.softmax\_cross\_entropy\_loss(self.reshape\_fn(logits, (-1, 2)),

                                                                self.reshape\_fn(label, (-1, 2))))

        return self.get\_loss(loss)

* + 1. 数据预处理

def \_load\_multipage\_tiff(path):

    """Load tiff images containing many images in the channel dimension"""

    return np.array([np.array(p) for p in ImageSequence.Iterator(Image.open(path))])

def \_get\_val\_train\_indices(length, fold, ratio=0.8):

    assert 0 < ratio <= 1, "Train/total data ratio must be in range (0.0, 1.0]"

    np.random.seed(0)

    indices = np.arange(0, length, 1, dtype=np.int)

    np.random.shuffle(indices)

    if fold is not None:

        indices = deque(indices)

        indices.rotate(fold \* round((1.0 - ratio) \* length))

        indices = np.array(indices)

        train\_indices = indices[:round(ratio \* len(indices))]

        val\_indices = indices[round(ratio \* len(indices)):]

    else:

        train\_indices = indices

        val\_indices = []

    return train\_indices, val\_indices

def data\_post\_process(img, mask):

    img = np.expand\_dims(img, axis=0)

    mask = (mask > 0.5).astype(np.int)

    mask = (np.arange(mask.max() + 1) == mask[..., None]).astype(int)

    mask = mask.transpose(2, 0, 1).astype(np.float32)

    return img, mask

def train\_data\_augmentation(img, mask):

    h\_flip = np.random.random()

    if h\_flip > 0.5:

        img = np.flipud(img)

        mask = np.flipud(mask)

    v\_flip = np.random.random()

    if v\_flip > 0.5:

        img = np.fliplr(img)

        mask = np.fliplr(mask)

    left = int(np.random.uniform()\*0.3\*572)

    right = int((1-np.random.uniform()\*0.3)\*572)

    top = int(np.random.uniform()\*0.3\*572)

    bottom = int((1-np.random.uniform()\*0.3)\*572)

    img = img[top:bottom, left:right]

    mask = mask[top:bottom, left:right]

    #adjust brightness

    brightness = np.random.uniform(-0.2, 0.2)

    img = np.float32(img+brightness\*np.ones(img.shape))

    img = np.clip(img, -1.0, 1.0)

    return img, mask

def create\_dataset(data\_dir, repeat=400, train\_batch\_size=16, augment=False, cross\_val\_ind=1, run\_distribute=False):

    images = \_load\_multipage\_tiff(os.path.join(data\_dir, 'train-volume.tif'))

    masks = \_load\_multipage\_tiff(os.path.join(data\_dir, 'train-labels.tif'))

    train\_indices, val\_indices = \_get\_val\_train\_indices(len(images), cross\_val\_ind)

    train\_images = images[train\_indices]

    train\_masks = masks[train\_indices]

    train\_images = np.repeat(train\_images, repeat, axis=0)

    train\_masks = np.repeat(train\_masks, repeat, axis=0)

    val\_images = images[val\_indices]

    val\_masks = masks[val\_indices]

    train\_image\_data = {"image": train\_images}

    train\_mask\_data = {"mask": train\_masks}

    valid\_image\_data = {"image": val\_images}

    valid\_mask\_data = {"mask": val\_masks}

    ds\_train\_images = ds.NumpySlicesDataset(data=train\_image\_data, sampler=None, shuffle=False)

    ds\_train\_masks = ds.NumpySlicesDataset(data=train\_mask\_data, sampler=None, shuffle=False)

    if run\_distribute:

        rank\_id = get\_rank()

        rank\_size = get\_group\_size()

        ds\_train\_images = ds.NumpySlicesDataset(data=train\_image\_data,

                                                sampler=None,

                                                shuffle=False,

                                                num\_shards=rank\_size,

                                                shard\_id=rank\_id)

        ds\_train\_masks = ds.NumpySlicesDataset(data=train\_mask\_data,

                                               sampler=None,

                                               shuffle=False,

                                               num\_shards=rank\_size,

                                               shard\_id=rank\_id)

    ds\_valid\_images = ds.NumpySlicesDataset(data=valid\_image\_data, sampler=None, shuffle=False)

ds\_valid\_masks = ds.NumpySlicesDataset(data=valid\_mask\_data, sampler=None, shuffle=False)

    c\_resize\_op = c\_vision.Resize(size=(388, 388), interpolation=Inter.BILINEAR)

    c\_pad = c\_vision.Pad(padding=92)

    c\_rescale\_image = c\_vision.Rescale(1.0/127.5, -1)

c\_rescale\_mask = c\_vision.Rescale(1.0/255.0, 0)

    c\_trans\_normalize\_img = [c\_rescale\_image, c\_resize\_op, c\_pad]

    c\_trans\_normalize\_mask = [c\_rescale\_mask, c\_resize\_op, c\_pad]

c\_center\_crop = c\_vision.CenterCrop(size=388)

    train\_image\_ds = ds\_train\_images.map(input\_columns="image", operations=c\_trans\_normalize\_img)

    train\_mask\_ds = ds\_train\_masks.map(input\_columns="mask", operations=c\_trans\_normalize\_mask)

    train\_ds = ds.zip((train\_image\_ds, train\_mask\_ds))

    train\_ds = train\_ds.project(columns=["image", "mask"])

    if augment:

        augment\_process = train\_data\_augmentation

        c\_resize\_op = c\_vision.Resize(size=(572, 572), interpolation=Inter.BILINEAR)

        train\_ds = train\_ds.map(input\_columns=["image", "mask"], operations=augment\_process)

        train\_ds = train\_ds.map(input\_columns="image", operations=c\_resize\_op)

        train\_ds = train\_ds.map(input\_columns="mask", operations=c\_resize\_op)

    train\_ds = train\_ds.map(input\_columns="mask", operations=c\_center\_crop)

    post\_process = data\_post\_process

    train\_ds = train\_ds.map(input\_columns=["image", "mask"], operations=post\_process)

    train\_ds = train\_ds.shuffle(repeat\*24)

train\_ds = train\_ds.batch(batch\_size=train\_batch\_size, drop\_remainder=True)

    valid\_image\_ds = ds\_valid\_images.map(input\_columns="image", operations=c\_trans\_normalize\_img)

    valid\_mask\_ds = ds\_valid\_masks.map(input\_columns="mask", operations=c\_trans\_normalize\_mask)

    valid\_ds = ds.zip((valid\_image\_ds, valid\_mask\_ds))

    valid\_ds = valid\_ds.project(columns=["image", "mask"])

    valid\_ds = valid\_ds.map(input\_columns="mask", operations=c\_center\_crop)

    post\_process = data\_post\_process

    valid\_ds = valid\_ds.map(input\_columns=["image", "mask"], operations=post\_process)

valid\_ds = valid\_ds.batch(batch\_size=1, drop\_remainder=True)

    return train\_ds, valid\_ds

* 1. 模型训练
     1. 定义类用于打印损失和速度

class StepLossTimeMonitor(Callback):

    def \_\_init\_\_(self, batch\_size, per\_print\_times=1):

        super(StepLossTimeMonitor, self).\_\_init\_\_()

        if not isinstance(per\_print\_times, int) or per\_print\_times < 0:

            raise ValueError("print\_step must be int and >= 0.")

        self.\_per\_print\_times = per\_print\_times

        self.batch\_size = batch\_size

    def step\_begin(self, run\_context):

        self.step\_time = time.time()

def step\_end(self, run\_context):

        step\_seconds = time.time() - self.step\_time

        step\_fps = self.batch\_size\*1.0/step\_seconds

        cb\_params = run\_context.original\_args()

        loss = cb\_params.net\_outputs

        if isinstance(loss, (tuple, list)):

            if isinstance(loss[0], Tensor) and isinstance(loss[0].asnumpy(), np.ndarray):

                loss = loss[0]

        if isinstance(loss, Tensor) and isinstance(loss.asnumpy(), np.ndarray):

            loss = np.mean(loss.asnumpy())

        cur\_step\_in\_epoch = (cb\_params.cur\_step\_num - 1) % cb\_params.batch\_num + 1

        if isinstance(loss, float) and (np.isnan(loss) or np.isinf(loss)):

            raise ValueError("epoch: {} step: {}. Invalid loss, terminating training.".format(

                cb\_params.cur\_epoch\_num, cur\_step\_in\_epoch))

        if self.\_per\_print\_times != 0 and cb\_params.cur\_step\_num % self.\_per\_print\_times == 0:

            # TEST

            print("step: %s, loss is %s, fps is %s" % (cur\_step\_in\_epoch, loss, step\_fps), flush=True)

* + 1. 定义类用于训练模型

def train\_net(data\_dir, cross\_valid\_ind=1, epochs=400, batch\_size=16, lr=0.0001, run\_distribute=False, cfg=None):

    if run\_distribute:

        init()

        group\_size = get\_group\_size()

        parallel\_mode = ParallelMode.DATA\_PARALLEL

        context.set\_auto\_parallel\_context(parallel\_mode=parallel\_mode,

                                          device\_num=group\_size,

                                          gradients\_mean=False)

    net = UNet(n\_channels=cfg['num\_channels'], n\_classes=cfg['num\_classes'])

    if cfg['resume']:

        param\_dict = load\_checkpoint(cfg['resume\_ckpt'])

        load\_param\_into\_net(net, param\_dict)

    criterion = CrossEntropyWithLogits()

    train\_dataset, \_ = create\_dataset(data\_dir, epochs, batch\_size, True, cross\_valid\_ind, run\_distribute)

    train\_data\_size = train\_dataset.get\_dataset\_size()

    print("dataset length is:", train\_data\_size)

    ckpt\_config = CheckpointConfig(save\_checkpoint\_steps=train\_data\_size,

                                   keep\_checkpoint\_max=cfg['keep\_checkpoint\_max'])

    ckpoint\_cb = ModelCheckpoint(prefix='ckpt\_unet\_medical\_adam',

                                 directory='./ckpt\_{}/'.format(device\_id),

                                 config=ckpt\_config)

    optimizer = nn.Adam(params=net.trainable\_params(), learning\_rate=lr, weight\_decay=cfg['weight\_decay'],

                        loss\_scale=cfg['loss\_scale'])

    loss\_scale\_manager = mindspore.train.loss\_scale\_manager.FixedLossScaleManager(cfg['FixedLossScaleManager'], False)

    model = Model(net, loss\_fn=criterion, loss\_scale\_manager=loss\_scale\_manager, optimizer=optimizer, amp\_level="O3")

    print("============== Starting Training ==============")

    model.train(2, train\_dataset, callbacks=[StepLossTimeMonitor(batch\_size=batch\_size), ckpoint\_cb],

                dataset\_sink\_mode=False)

    print("============== End Training ==============")

* + 1. 训练模型

data\_url = './data'

run\_distribute = False

epoch\_size = cfg\_unet['epochs'] if not run\_distribute else cfg\_unet['distribute\_epochs']

train\_net(data\_dir=data\_url, cross\_valid\_ind=cfg\_unet['cross\_valid\_ind'], epochs=epoch\_size,

          batch\_size=cfg\_unet['batchsize'], lr=cfg\_unet['lr'], run\_distribute=run\_distribute,

          cfg=cfg\_unet)

dataset length is: 600

============== Starting Training ==============

step: 1, loss is 0.7007539, fps is 0.1921919720257687

step: 2, loss is 0.68959564, fps is 66.86380814315534

step: 3, loss is 0.68191934, fps is 68.38402083653894

step: 4, loss is 0.66419667, fps is 69.69344542353167

step: 5, loss is 0.62513983, fps is 68.18475118901141

step: 6, loss is 0.5507719, fps is 69.44614340234948

step: 7, loss is 0.5840579, fps is 68.46529849375834

step: 8, loss is 0.57061434, fps is 69.46914776444477

step: 9, loss is 0.544724, fps is 68.18191091396396

step: 10, loss is 0.5460727, fps is 69.47432584641886

step: 11, loss is 0.5698999, fps is 68.40019120985077

step: 12, loss is 0.56544375, fps is 69.69359017896737

step: 13, loss is 0.55058885, fps is 68.25923691980955

step: 14, loss is 0.542965, fps is 69.3874794372395

step: 15, loss is 0.5487885, fps is 68.26708333587648

step: 16, loss is 0.5379101, fps is 69.28532387901139

step: 17, loss is 0.5241088, fps is 67.34876028413208

step: 18, loss is 0.5739319, fps is 68.97781797811709

step: 19, loss is 0.5424314, fps is 68.40855819718838

step: 20, loss is 0.5409745, fps is 68.76633907061914

step: 21, loss is 0.55142844, fps is 68.65876700640253

step: 22, loss is 0.53180593, fps is 68.89128830224396

step: 23, loss is 0.5441859, fps is 67.89838643749519

step: 24, loss is 0.5403717, fps is 68.08423880512218

step: 25, loss is 0.54632163, fps is 67.7257097645354

step: 26, loss is 0.5353315, fps is 68.66712711563511

step: 27, loss is 0.5468826, fps is 68.04958141345966

step: 28, loss is 0.5361666, fps is 68.55685329851799

step: 29, loss is 0.5301911, fps is 68.21850622221048

step: 30, loss is 0.53177065, fps is 68.62057527327013

step: 31, loss is 0.52787715, fps is 68.17075992738894

step: 32, loss is 0.5366892, fps is 68.3280412398425

step: 33, loss is 0.53281623, fps is 68.17062142875282

step: 34, loss is 0.53748494, fps is 69.42351335889906

step: 35, loss is 0.5436444, fps is 68.43157797059361

step: 36, loss is 0.5256405, fps is 69.71943866123114

step: 37, loss is 0.5388684, fps is 69.48101534488988

step: 38, loss is 0.5095582, fps is 68.93693360534658

step: 39, loss is 0.5292056, fps is 68.71718982792252

step: 40, loss is 0.5214486, fps is 68.98171763375649

step: 41, loss is 0.514818, fps is 68.20484769765359

step: 42, loss is 0.5138229, fps is 68.66466804864622

step: 43, loss is 0.5085483, fps is 68.2977919624829

step: 44, loss is 0.5306792, fps is 68.80173058094412

step: 45, loss is 0.52113897, fps is 68.56581908984

step: 46, loss is 0.504831, fps is 69.05205796314074

step: 47, loss is 0.5106175, fps is 68.28396271435128

step: 48, loss is 0.49835697, fps is 68.88421692275004

step: 49, loss is 0.5094215, fps is 68.59637642272683

step: 50, loss is 0.48664993, fps is 69.08212584256708

step: 51, loss is 0.49854907, fps is 68.59960194997491

step: 52, loss is 0.5046706, fps is 69.08546833451892

step: 53, loss is 0.49254832, fps is 68.37565985004173

step: 54, loss is 0.48819235, fps is 68.67155388009546

step: 55, loss is 0.48756364, fps is 68.36075449710549

step: 56, loss is 0.4801605, fps is 69.08539721430115

step: 57, loss is 0.4710208, fps is 68.66649476730147

step: 58, loss is 0.4701817, fps is 69.08632178855378

step: 59, loss is 0.45800185, fps is 68.4336714455137

step: 60, loss is 0.4654809, fps is 69.02528192743326

step: 61, loss is 0.45771974, fps is 68.64928531314159

step: 62, loss is 0.4637226, fps is 68.74091070188413

step: 63, loss is 0.44668278, fps is 69.44175992466927

step: 64, loss is 0.43674305, fps is 68.49702470262336

step: 65, loss is 0.43379036, fps is 68.2121269411461

step: 66, loss is 0.45250782, fps is 69.08482825783047

step: 67, loss is 0.42090103, fps is 68.44323327594788

step: 68, loss is 0.4143166, fps is 68.75224900803916

step: 69, loss is 0.46226743, fps is 68.38555390586866

step: 70, loss is 0.42448235, fps is 69.72450923961492

step: 71, loss is 0.41091245, fps is 68.55125086954612

step: 72, loss is 0.4190496, fps is 68.67366206378541

step: 73, loss is 0.4177092, fps is 69.4638266511265

step: 74, loss is 0.41465545, fps is 68.71500861128006

step: 75, loss is 0.40030777, fps is 69.46109450429338

step: 76, loss is 0.40752137, fps is 68.4562891391254

step: 77, loss is 0.399377, fps is 67.83702664399667

step: 78, loss is 0.40581402, fps is 68.4613173237303

step: 79, loss is 0.40000334, fps is 68.05565426209425

step: 80, loss is 0.37630162, fps is 68.47263344022905

step: 81, loss is 0.38288435, fps is 68.07968024022561

step: 82, loss is 0.39028332, fps is 68.66579217170116

step: 83, loss is 0.3799576, fps is 68.25819549575756

step: 84, loss is 0.36951226, fps is 68.76190007787204

step: 85, loss is 0.3661428, fps is 68.32588465751974

step: 86, loss is 0.37919763, fps is 68.73626378128571

step: 87, loss is 0.37322092, fps is 68.22585774296788

step: 88, loss is 0.35616004, fps is 68.32414557738494

step: 89, loss is 0.3604926, fps is 68.05275572186223

step: 90, loss is 0.35474628, fps is 68.76182962229024

step: 91, loss is 0.3604562, fps is 68.09024875531028

step: 92, loss is 0.35675442, fps is 69.38181216657706

step: 93, loss is 0.36966935, fps is 69.78664744927617

step: 94, loss is 0.36687288, fps is 68.6697268922611

step: 95, loss is 0.3675928, fps is 69.6285831676717

step: 96, loss is 0.32752863, fps is 68.58270635966929

step: 97, loss is 0.38295054, fps is 69.64202294251089

step: 98, loss is 0.32807994, fps is 69.81147565711385

step: 99, loss is 0.38122687, fps is 68.00668829892237

step: 100, loss is 0.34681472, fps is 69.61016609841433

step: 101, loss is 0.34355706, fps is 68.08928156525815

step: 102, loss is 0.3411967, fps is 68.81894598887762

step: 103, loss is 0.3476376, fps is 66.3123204072679

step: 104, loss is 0.3369664, fps is 68.52220295821617

step: 105, loss is 0.34419966, fps is 68.09943843142348

step: 106, loss is 0.32950956, fps is 69.84744252658224

step: 107, loss is 0.3398701, fps is 69.90033383156349

step: 108, loss is 0.3235427, fps is 68.65504425159006

step: 109, loss is 0.3284481, fps is 69.59175876937371

step: 110, loss is 0.32152706, fps is 69.92036174579856

step: 111, loss is 0.32570276, fps is 69.87478759204345

step: 112, loss is 0.33673638, fps is 68.71634546956965

step: 113, loss is 0.33720025, fps is 68.3434891311543

step: 114, loss is 0.32316503, fps is 69.73820348603394

step: 115, loss is 0.30764234, fps is 69.64354066240836

step: 116, loss is 0.30919722, fps is 68.58705214789136

step: 117, loss is 0.32366976, fps is 69.74653859664804

step: 118, loss is 0.31892183, fps is 69.8624942092579

step: 119, loss is 0.3166284, fps is 69.85689451648967

step: 120, loss is 0.31269357, fps is 68.72556417157206

step: 121, loss is 0.29750156, fps is 68.1848897450669

step: 122, loss is 0.32792112, fps is 69.91671945324221

step: 123, loss is 0.29964212, fps is 68.23029716419063

step: 124, loss is 0.2971104, fps is 69.7798264157143

step: 125, loss is 0.30321223, fps is 69.48029598216735

step: 126, loss is 0.30806878, fps is 68.77247005052213

step: 127, loss is 0.3075993, fps is 69.6784664871117

step: 128, loss is 0.3305972, fps is 68.54501924318318

step: 129, loss is 0.3460342, fps is 68.87813669734469

step: 130, loss is 0.34790528, fps is 69.62056512812745

step: 131, loss is 0.3063764, fps is 68.79185677602547

step: 132, loss is 0.30924982, fps is 68.68476730095307

step: 133, loss is 0.3011819, fps is 68.22932599143128

step: 134, loss is 0.31252465, fps is 68.75774344556817

step: 135, loss is 0.29032284, fps is 68.18627533659351

step: 136, loss is 0.30081496, fps is 68.43841712957628

step: 137, loss is 0.2953038, fps is 68.00772206250006

step: 138, loss is 0.33097842, fps is 68.59329141961182

step: 139, loss is 0.29886186, fps is 69.35047955051117

step: 140, loss is 0.31080696, fps is 69.58266697634295

step: 141, loss is 0.29082665, fps is 68.09384141608365

step: 142, loss is 0.31422755, fps is 68.65139212976366

step: 143, loss is 0.29641926, fps is 69.52355615228103

step: 144, loss is 0.28913966, fps is 68.61447134471713

step: 145, loss is 0.281744, fps is 68.31308709938801

step: 146, loss is 0.30015543, fps is 68.56953217437861

step: 147, loss is 0.29751298, fps is 68.21670326095796

step: 148, loss is 0.28373688, fps is 68.8815301791607

step: 149, loss is 0.28043514, fps is 68.36576865675443

step: 150, loss is 0.31237477, fps is 68.86004377319615

step: 151, loss is 0.30090162, fps is 68.39628731615193

step: 152, loss is 0.28625405, fps is 69.07032303350549

step: 153, loss is 0.3046996, fps is 68.26951400764395

step: 154, loss is 0.2970054, fps is 68.79559438308446

step: 155, loss is 0.2777652, fps is 68.23563910860031

step: 156, loss is 0.27823102, fps is 68.74767098084129

step: 157, loss is 0.28177622, fps is 69.51426615164539

step: 158, loss is 0.28350708, fps is 68.57962258597891

step: 159, loss is 0.26845008, fps is 68.2770849230331

step: 160, loss is 0.29733947, fps is 68.80398784862662

step: 161, loss is 0.2798301, fps is 68.10164985153568

step: 162, loss is 0.26647225, fps is 68.85700565968817

step: 163, loss is 0.28019533, fps is 68.23008905341476

step: 164, loss is 0.28307685, fps is 68.58501937194231

step: 165, loss is 0.27435327, fps is 69.3453915688795

step: 166, loss is 0.2714198, fps is 68.76908730573506

step: 167, loss is 0.26547363, fps is 68.53157959233691

step: 168, loss is 0.2721162, fps is 69.03650118611881

step: 169, loss is 0.28836715, fps is 69.23385882903972

step: 170, loss is 0.28224367, fps is 68.83871199368942

step: 171, loss is 0.29041576, fps is 69.30299688126071

step: 172, loss is 0.27760494, fps is 68.75809568305372

step: 173, loss is 0.28057054, fps is 68.39517199980024

step: 174, loss is 0.2770548, fps is 69.03898695839587

step: 175, loss is 0.26765722, fps is 68.55741359178073

step: 176, loss is 0.30340618, fps is 69.02166129959569

step: 177, loss is 0.2699868, fps is 68.59034690275256

step: 178, loss is 0.25743002, fps is 68.63117202672478

step: 179, loss is 0.28036544, fps is 68.79453652855929

step: 180, loss is 0.28048912, fps is 68.91647420184006

step: 181, loss is 0.2820357, fps is 65.45386307221895

step: 182, loss is 0.27211457, fps is 69.40556350183937

step: 183, loss is 0.2636395, fps is 69.79688150034114

step: 184, loss is 0.2668308, fps is 69.53515413316866

step: 185, loss is 0.25991994, fps is 69.95009704163722

step: 186, loss is 0.25460175, fps is 69.2604395547689

step: 187, loss is 0.25435415, fps is 69.4061377535032

step: 188, loss is 0.24968274, fps is 68.67267822862219

step: 189, loss is 0.26162693, fps is 70.28158561962681

step: 190, loss is 0.24143745, fps is 69.2629414800289

step: 191, loss is 0.25600907, fps is 69.60309075728739

step: 192, loss is 0.24994458, fps is 69.14497611155701

step: 193, loss is 0.2737018, fps is 69.87420555921135

step: 194, loss is 0.25921658, fps is 69.16407447296413

step: 195, loss is 0.26516646, fps is 69.7566883774305

step: 196, loss is 0.25548905, fps is 69.37836921371918

step: 197, loss is 0.26101825, fps is 70.05992829984612

step: 198, loss is 0.25596684, fps is 69.55526166321873

step: 199, loss is 0.23997699, fps is 69.55691979208234

step: 200, loss is 0.25377712, fps is 69.2447887539029

step: 201, loss is 0.2507464, fps is 69.55828961033846

step: 202, loss is 0.25735408, fps is 69.12033486490355

step: 203, loss is 0.26473993, fps is 69.63660305426683

step: 204, loss is 0.25160676, fps is 69.47044221257426

step: 205, loss is 0.2811937, fps is 69.98423633401953

step: 206, loss is 0.27065453, fps is 69.70647582148774

step: 207, loss is 0.24876483, fps is 69.87158653145367

step: 208, loss is 0.26931316, fps is 69.74740846212208

step: 209, loss is 0.26404554, fps is 69.99211936512704

step: 210, loss is 0.25480923, fps is 69.12417945192075

step: 211, loss is 0.26715097, fps is 68.44393132441402

step: 212, loss is 0.25367665, fps is 68.54985040537132

step: 213, loss is 0.27850217, fps is 68.7234528000852

step: 214, loss is 0.26090306, fps is 69.315738481783

step: 215, loss is 0.2727887, fps is 69.54099060543446

step: 216, loss is 0.28970158, fps is 69.35506654003201

step: 217, loss is 0.23895378, fps is 69.74392913040496

step: 218, loss is 0.2734999, fps is 69.41432187234624

step: 219, loss is 0.26248643, fps is 69.36596307241325

step: 220, loss is 0.25531316, fps is 70.0345576456469

step: 221, loss is 0.28421512, fps is 68.60121482735427

step: 222, loss is 0.2834223, fps is 69.38310336199963

step: 223, loss is 0.2846061, fps is 69.71654151897577

step: 224, loss is 0.2485065, fps is 69.1894602594623

step: 225, loss is 0.2488975, fps is 69.75632583439271

step: 226, loss is 0.24140848, fps is 69.15153107646778

step: 227, loss is 0.27181935, fps is 67.48746869940365

step: 228, loss is 0.2497071, fps is 69.40606597152555

step: 229, loss is 0.26128933, fps is 69.5255729145083

step: 230, loss is 0.2494817, fps is 69.13493232676826

step: 231, loss is 0.23571712, fps is 69.45800312364479

step: 232, loss is 0.25235772, fps is 69.2694473317238

step: 233, loss is 0.23074122, fps is 69.49828452008505

step: 234, loss is 0.24094343, fps is 69.34524825626453

step: 235, loss is 0.26398915, fps is 69.69663018202854

step: 236, loss is 0.24631707, fps is 69.25572213031772

step: 237, loss is 0.25745198, fps is 69.7907842574451

step: 238, loss is 0.24153605, fps is 69.37614582159557

step: 239, loss is 0.25430235, fps is 69.76401255378448

step: 240, loss is 0.25412738, fps is 69.36732537971268

step: 241, loss is 0.24404597, fps is 70.19755648535565

step: 242, loss is 0.22180301, fps is 68.90614292593722

step: 243, loss is 0.24721181, fps is 69.54538662024011

step: 244, loss is 0.22971208, fps is 69.20701301353122

step: 245, loss is 0.24409366, fps is 70.03243816357667

step: 246, loss is 0.25885063, fps is 69.71168934507715

step: 247, loss is 0.25345582, fps is 69.88701252176521

step: 248, loss is 0.23521331, fps is 69.32970235555551

step: 249, loss is 0.25430793, fps is 69.58136834268912

step: 250, loss is 0.23147053, fps is 69.58100761972106

step: 251, loss is 0.24603692, fps is 69.71704850148922

step: 252, loss is 0.23483035, fps is 69.63082276990312

step: 253, loss is 0.23285666, fps is 69.74994569369163

step: 254, loss is 0.25151756, fps is 70.37275224932363

step: 255, loss is 0.23893133, fps is 70.23377536742274

step: 256, loss is 0.2572706, fps is 69.52384425400719

step: 257, loss is 0.24823555, fps is 69.73878325534221

step: 258, loss is 0.23911576, fps is 69.62627147131694

step: 259, loss is 0.24634857, fps is 55.0098357054387

step: 260, loss is 0.22519675, fps is 70.41750068729316

step: 261, loss is 0.22311819, fps is 69.49267109662762

step: 262, loss is 0.23384644, fps is 69.80886132964675

step: 263, loss is 0.25012, fps is 70.0463998714072

step: 264, loss is 0.23349673, fps is 69.86569443678094

step: 265, loss is 0.24200699, fps is 69.93048931330244

step: 266, loss is 0.21950263, fps is 69.74516135402062

step: 267, loss is 0.23404677, fps is 69.48101534488988

step: 268, loss is 0.24251431, fps is 69.7485683165065

step: 269, loss is 0.22745936, fps is 69.93150951870342

step: 270, loss is 0.22866929, fps is 69.86009423100546

step: 271, loss is 0.23522742, fps is 69.95593045777034

step: 272, loss is 0.23876423, fps is 69.5280940318378

step: 273, loss is 0.23795812, fps is 69.47936083289677

step: 274, loss is 0.22674067, fps is 69.51779461914252

step: 275, loss is 0.22596212, fps is 69.64108343477173

step: 276, loss is 0.23412278, fps is 69.36624986950325

step: 277, loss is 0.2259152, fps is 69.86467615076701

step: 278, loss is 0.24141882, fps is 69.37958855377656

step: 279, loss is 0.24399221, fps is 69.39350641158873

step: 280, loss is 0.22011068, fps is 69.47058604312197

step: 281, loss is 0.25322783, fps is 69.59803782272994

step: 282, loss is 0.24748327, fps is 69.6504796530183

step: 283, loss is 0.25947985, fps is 69.64043302118615

step: 284, loss is 0.2506118, fps is 69.68808989510784

step: 285, loss is 0.22568126, fps is 69.69858460976025

step: 286, loss is 0.22425537, fps is 69.73117455013228

step: 287, loss is 0.22696681, fps is 69.49159169607502

step: 288, loss is 0.21557534, fps is 69.59089278196144

step: 289, loss is 0.25262544, fps is 69.59168660293284

step: 290, loss is 0.22840334, fps is 69.67542806832482

step: 291, loss is 0.23886207, fps is 69.44722139147227

step: 292, loss is 0.2252227, fps is 69.66089073196555

step: 293, loss is 0.22479525, fps is 69.72805908334779

step: 294, loss is 0.2420184, fps is 69.58057475709633

step: 295, loss is 0.23827648, fps is 69.56557218159296

step: 296, loss is 0.23078007, fps is 69.68157752375454

step: 297, loss is 0.21606442, fps is 69.48972081454805

step: 298, loss is 0.22073746, fps is 69.59681079219955

step: 299, loss is 0.21284749, fps is 69.50555453135001

step: 300, loss is 0.22341394, fps is 69.12802446664627

step: 301, loss is 0.2127525, fps is 69.47216821845542

step: 302, loss is 0.2349858, fps is 69.62648818632948

step: 303, loss is 0.2272016, fps is 69.11755848468182

step: 304, loss is 0.2261703, fps is 69.53810827445999

step: 305, loss is 0.21933392, fps is 69.44851502259624

step: 306, loss is 0.22024947, fps is 69.68389289411628

step: 307, loss is 0.2245857, fps is 69.39156905565483

step: 308, loss is 0.23561159, fps is 69.73204403241125

step: 309, loss is 0.23626637, fps is 69.86671275247859

step: 310, loss is 0.2301722, fps is 69.45958472587789

step: 311, loss is 0.23697223, fps is 69.49871635945249

step: 312, loss is 0.23033293, fps is 69.51837072948392

step: 313, loss is 0.22237359, fps is 69.40541994040817

step: 314, loss is 0.21869783, fps is 69.43838286847148

step: 315, loss is 0.22113843, fps is 69.83646688721392

step: 316, loss is 0.20956244, fps is 69.60374047353334

step: 317, loss is 0.2135254, fps is 69.77910085231557

step: 318, loss is 0.21436204, fps is 69.71183417630921

step: 319, loss is 0.23254451, fps is 69.51153003568339

step: 320, loss is 0.22465603, fps is 69.35793371662913

step: 321, loss is 0.23486236, fps is 69.55677560382543

step: 322, loss is 0.219012, fps is 69.4444905506914

step: 323, loss is 0.22958955, fps is 69.38037756135888

step: 324, loss is 0.2249033, fps is 69.43349749617184

step: 325, loss is 0.24607475, fps is 69.67984109695432

step: 326, loss is 0.22471307, fps is 69.52823810120772

step: 327, loss is 0.23740156, fps is 69.73088472752437

step: 328, loss is 0.22503948, fps is 69.46066313233459

step: 329, loss is 0.22881621, fps is 70.05985515915893

step: 330, loss is 0.23024607, fps is 69.32741046220089

step: 331, loss is 0.214774, fps is 69.92932340072129

step: 332, loss is 0.2388385, fps is 69.49360660423078

step: 333, loss is 0.21740033, fps is 69.83145267696969

step: 334, loss is 0.23701367, fps is 69.96110842956493

step: 335, loss is 0.21518888, fps is 70.04464521489176

step: 336, loss is 0.2523286, fps is 69.4974928285161

step: 337, loss is 0.22690956, fps is 69.7299428206868

step: 338, loss is 0.22898136, fps is 69.46181346946464

step: 339, loss is 0.21678422, fps is 70.18081880053711

step: 340, loss is 0.23191498, fps is 69.85631278170443

step: 341, loss is 0.2173424, fps is 70.12054124653885

step: 342, loss is 0.23001377, fps is 69.57675137113412

step: 343, loss is 0.20947611, fps is 69.94667036330985

step: 344, loss is 0.21965185, fps is 69.5684567898035

step: 345, loss is 0.22762215, fps is 70.05473569052215

step: 346, loss is 0.20769, fps is 69.62843868215236

step: 347, loss is 0.21008265, fps is 69.51549027326027

step: 348, loss is 0.21659587, fps is 69.66848410605651

step: 349, loss is 0.22511294, fps is 69.85304070405896

step: 350, loss is 0.21021062, fps is 69.80450455229601

step: 351, loss is 0.20428008, fps is 69.86220329402065

step: 352, loss is 0.22758019, fps is 69.79557485860722

step: 353, loss is 0.22984758, fps is 69.82193492539612

step: 354, loss is 0.20826653, fps is 69.47080179006026

step: 355, loss is 0.21398115, fps is 69.80160033616836

step: 356, loss is 0.23337995, fps is 69.5944290329104

step: 357, loss is 0.22860609, fps is 69.76720376506009

step: 358, loss is 0.19866084, fps is 70.15066707504084

step: 359, loss is 0.21744826, fps is 69.5802861850052

step: 360, loss is 0.21586116, fps is 69.6655188772772

step: 361, loss is 0.21478952, fps is 69.82462288238759

step: 362, loss is 0.22678757, fps is 70.20350469025095

step: 363, loss is 0.21311966, fps is 69.79513932223415

step: 364, loss is 0.19861242, fps is 69.5454586906869

step: 365, loss is 0.20867485, fps is 69.83159800626424

step: 366, loss is 0.21672273, fps is 69.96942395974631

step: 367, loss is 0.22775054, fps is 69.5978934639513

step: 368, loss is 0.21117651, fps is 69.84591590663324

step: 369, loss is 0.2139209, fps is 69.65380507606416

step: 370, loss is 0.22431228, fps is 69.53486593770043

step: 371, loss is 0.21015684, fps is 63.80206668574455

step: 372, loss is 0.21995084, fps is 69.47360662220667

step: 373, loss is 0.20520553, fps is 69.56946645919528

step: 374, loss is 0.20771787, fps is 69.74008777153213

step: 375, loss is 0.2108471, fps is 69.46648710588669

step: 376, loss is 0.21343245, fps is 70.14604756758112

step: 377, loss is 0.22420324, fps is 69.4938944577914

step: 378, loss is 0.20854285, fps is 69.58872790771949

step: 379, loss is 0.20234816, fps is 69.46864438097742

step: 380, loss is 0.20734796, fps is 69.58512008328374

step: 381, loss is 0.20349792, fps is 69.26079696159681

step: 382, loss is 0.21893036, fps is 69.66190308620638

step: 383, loss is 0.21837577, fps is 69.54617940337053

step: 384, loss is 0.20584172, fps is 69.61197126266283

step: 385, loss is 0.22067168, fps is 69.50166740543429

step: 386, loss is 0.2181633, fps is 67.19171138013171

step: 387, loss is 0.22962412, fps is 69.51498621796233

step: 388, loss is 0.22751422, fps is 69.63472435658646

step: 389, loss is 0.20854059, fps is 69.53724362175907

step: 390, loss is 0.22075412, fps is 69.51124203600642

step: 391, loss is 0.22067967, fps is 69.4834612895711

step: 392, loss is 0.21140264, fps is 69.64845563792568

step: 393, loss is 0.21037705, fps is 69.44765259649206

step: 394, loss is 0.2193297, fps is 69.57530869317299

step: 395, loss is 0.19408482, fps is 69.61146580723594

step: 396, loss is 0.22784191, fps is 69.96541183437381

step: 397, loss is 0.21387403, fps is 69.39809909484154

step: 398, loss is 0.23013511, fps is 68.22946472870424

step: 399, loss is 0.22418685, fps is 69.327123986186

step: 400, loss is 0.23340783, fps is 69.10773616071245

step: 401, loss is 0.24358991, fps is 69.2005902412214

step: 402, loss is 0.22403836, fps is 69.25779485888137

step: 403, loss is 0.22191021, fps is 69.3367938603198

step: 404, loss is 0.21492681, fps is 69.32977397966458

step: 405, loss is 0.22997317, fps is 69.55273857532106

step: 406, loss is 0.21726422, fps is 69.26723091532142

step: 407, loss is 0.21562304, fps is 69.25079096453301

step: 408, loss is 0.22411862, fps is 69.23443024288687

step: 409, loss is 0.22254379, fps is 69.41755297690594

step: 410, loss is 0.22865896, fps is 69.16329037736938

step: 411, loss is 0.22267954, fps is 69.07238467927783

step: 412, loss is 0.22087143, fps is 69.32024927202693

step: 413, loss is 0.22082673, fps is 68.97562018467814

step: 414, loss is 0.19826598, fps is 69.32411613096498

step: 415, loss is 0.19936135, fps is 69.12773963607147

step: 416, loss is 0.20948355, fps is 69.31530891332909

step: 417, loss is 0.22418618, fps is 69.18432455224273

step: 418, loss is 0.20664242, fps is 69.14832468835844

step: 419, loss is 0.20941381, fps is 68.7373198444345

step: 420, loss is 0.21385944, fps is 69.1585861771174

step: 421, loss is 0.21407515, fps is 69.70864802821217

step: 422, loss is 0.21566164, fps is 69.25150558272965

step: 423, loss is 0.20863245, fps is 64.73259516412031

step: 424, loss is 0.24526754, fps is 69.35442145796603

step: 425, loss is 0.20470376, fps is 68.9949088277144

step: 426, loss is 0.22089778, fps is 69.31087035105296

step: 427, loss is 0.20311937, fps is 68.86597946607695

step: 428, loss is 0.22548902, fps is 69.76031401508537

step: 429, loss is 0.20480748, fps is 68.89425871458856

step: 430, loss is 0.20753133, fps is 69.3159532680066

step: 431, loss is 0.22131932, fps is 69.24135938638179

step: 432, loss is 0.19739185, fps is 69.88104505742818

step: 433, loss is 0.21816556, fps is 69.35040788361418

step: 434, loss is 0.2182081, fps is 69.52074728557858

step: 435, loss is 0.21737976, fps is 69.38726420751806

step: 436, loss is 0.20827052, fps is 69.26508613126632

step: 437, loss is 0.22310239, fps is 69.29691406149159

step: 438, loss is 0.2161264, fps is 69.21600689799568

step: 439, loss is 0.20644736, fps is 69.15231490545624

step: 440, loss is 0.2187406, fps is 69.27924416107825

step: 441, loss is 0.2028882, fps is 69.44161621345872

step: 442, loss is 0.2106876, fps is 69.39329114447615

step: 443, loss is 0.218744, fps is 69.11948057030973

step: 444, loss is 0.22195594, fps is 69.33442986421132

step: 445, loss is 0.2214871, fps is 69.45541520262053

step: 446, loss is 0.21913154, fps is 69.30414200205716

step: 447, loss is 0.2109756, fps is 69.50382686615393

step: 448, loss is 0.20402494, fps is 69.81568804103912

step: 449, loss is 0.2011174, fps is 69.2623695954621

step: 450, loss is 0.21532293, fps is 69.3442450845454

step: 451, loss is 0.21225457, fps is 69.08461490157

step: 452, loss is 0.20129518, fps is 69.18489514872788

step: 453, loss is 0.21130596, fps is 69.49353464121323

step: 454, loss is 0.20162947, fps is 69.5105220472547

step: 455, loss is 0.2077716, fps is 69.2788150448343

step: 456, loss is 0.19809827, fps is 68.98639780259543

step: 457, loss is 0.20488171, fps is 69.2028023948638

step: 458, loss is 0.2017256, fps is 69.13030319574683

step: 459, loss is 0.1861913, fps is 69.22271814396217

step: 460, loss is 0.19048929, fps is 69.17947486209744

step: 461, loss is 0.20919673, fps is 68.9921425142104

step: 462, loss is 0.20208427, fps is 69.28868606355958

step: 463, loss is 0.2067127, fps is 69.23843040393379

step: 464, loss is 0.21590246, fps is 69.2004475264881

step: 465, loss is 0.19577575, fps is 69.13671292685767

step: 466, loss is 0.19844203, fps is 69.18147171098954

step: 467, loss is 0.21351536, fps is 68.11893151652707

step: 468, loss is 0.19418944, fps is 69.18938892497397

step: 469, loss is 0.19152124, fps is 68.95067441497275

step: 470, loss is 0.20942813, fps is 69.42056894589842

step: 471, loss is 0.20022927, fps is 69.06285948336279

step: 472, loss is 0.21610117, fps is 69.31566688667094

step: 473, loss is 0.20212144, fps is 69.06705308918455

step: 474, loss is 0.21296544, fps is 69.28725530684729

step: 475, loss is 0.20234883, fps is 69.20694164284447

step: 476, loss is 0.21488012, fps is 69.22328937392659

step: 477, loss is 0.20746854, fps is 69.3397311709634

step: 478, loss is 0.20636775, fps is 69.38726420751806

step: 479, loss is 0.2060452, fps is 69.38231429241972

step: 480, loss is 0.22500701, fps is 67.77358734467659

step: 481, loss is 0.19489737, fps is 69.24643211289289

step: 482, loss is 0.21357559, fps is 69.14761219804723

step: 483, loss is 0.19541714, fps is 69.46900393985256

step: 484, loss is 0.2187878, fps is 69.08034805308705

step: 485, loss is 0.20511669, fps is 69.0625751895368

step: 486, loss is 0.21046847, fps is 69.10887483780611

step: 487, loss is 0.2033378, fps is 69.37750852889486

step: 488, loss is 0.20285091, fps is 68.91378494140021

step: 489, loss is 0.18619871, fps is 69.11570768844159

step: 490, loss is 0.19712809, fps is 69.13137140172898

step: 491, loss is 0.20708433, fps is 68.8085028283634

step: 492, loss is 0.19104168, fps is 68.59483388648287

step: 493, loss is 0.19316173, fps is 68.32358909043892

step: 494, loss is 0.19459848, fps is 68.92319827129353

step: 495, loss is 0.19840819, fps is 69.1171313690594

step: 496, loss is 0.2010721, fps is 68.97810157456925

step: 497, loss is 0.2021948, fps is 69.64093889736945

step: 498, loss is 0.194739, fps is 69.67144958726935

step: 499, loss is 0.20176807, fps is 69.55482912086431

step: 500, loss is 0.196168, fps is 69.6146430775774

step: 501, loss is 0.18476434, fps is 70.31214468768138

step: 502, loss is 0.20071723, fps is 69.56946645919528

step: 503, loss is 0.20928307, fps is 69.7027109838417

step: 504, loss is 0.208898, fps is 69.4971329746747

step: 505, loss is 0.20975377, fps is 69.65257607838252

step: 506, loss is 0.19459172, fps is 69.62836643961757

step: 507, loss is 0.20238544, fps is 69.5249246567466

step: 508, loss is 0.20922865, fps is 69.697860734857

step: 509, loss is 0.21439682, fps is 70.50368703439307

step: 510, loss is 0.19555907, fps is 70.4981322038314

step: 511, loss is 0.20609331, fps is 69.42351335889906

step: 512, loss is 0.19225715, fps is 69.40879379106059

step: 513, loss is 0.1945791, fps is 70.30882976928969

step: 514, loss is 0.21065545, fps is 70.29527868763165

step: 515, loss is 0.21179272, fps is 69.7135721980352

step: 516, loss is 0.20298366, fps is 69.72516122463186

step: 517, loss is 0.22925383, fps is 69.6392044953381

step: 518, loss is 0.2076314, fps is 69.60114168132488

step: 519, loss is 0.23742844, fps is 68.93197692573176

step: 520, loss is 0.20288809, fps is 69.54581904516151

step: 521, loss is 0.22536568, fps is 69.6094440589289

step: 522, loss is 0.20968008, fps is 69.51707449464344

step: 523, loss is 0.20672381, fps is 69.70850321021868

step: 524, loss is 0.21202719, fps is 69.60922745000416

step: 525, loss is 0.20440724, fps is 70.03017265218594

step: 526, loss is 0.2050942, fps is 69.58959384125299

step: 527, loss is 0.20735344, fps is 70.50442774460807

step: 528, loss is 0.21188328, fps is 70.07009634148761

step: 529, loss is 0.21136582, fps is 66.42205990062752

step: 530, loss is 0.1968829, fps is 68.80575144206419

step: 531, loss is 0.19340229, fps is 70.11035856032612

step: 532, loss is 0.21483232, fps is 69.1516023329143

step: 533, loss is 0.19891068, fps is 70.45439074450272

step: 534, loss is 0.20002453, fps is 70.50650181601179

step: 535, loss is 0.18682948, fps is 69.63407406177626

step: 536, loss is 0.19449075, fps is 69.1783338573424

step: 537, loss is 0.20686117, fps is 69.33034697786474

step: 538, loss is 0.20880572, fps is 69.85674908188503

step: 539, loss is 0.21526258, fps is 68.83666427325879

step: 540, loss is 0.20281184, fps is 69.6235265312559

step: 541, loss is 0.21271922, fps is 69.53854060887406

step: 542, loss is 0.19321874, fps is 70.27761121193494

step: 543, loss is 0.20204459, fps is 70.31398644408053

step: 544, loss is 0.2008477, fps is 69.42696079394916

step: 545, loss is 0.19623226, fps is 68.97243008856287

step: 546, loss is 0.19990146, fps is 70.19432578416892

step: 547, loss is 0.20833524, fps is 69.37564378503214

step: 548, loss is 0.19036397, fps is 69.43594009639042

step: 549, loss is 0.20691182, fps is 69.32984560392163

step: 550, loss is 0.19288217, fps is 70.43361223178938

step: 551, loss is 0.20073958, fps is 69.32762532076579

step: 552, loss is 0.18963175, fps is 69.48669883410301

step: 553, loss is 0.20562048, fps is 70.02111207219465

step: 554, loss is 0.20459175, fps is 70.39814702142927

step: 555, loss is 0.20303145, fps is 70.77359133280392

step: 556, loss is 0.21329305, fps is 69.62034844998475

step: 557, loss is 0.19387585, fps is 69.76292470757498

step: 558, loss is 0.19707024, fps is 59.83373945357881

step: 559, loss is 0.20256007, fps is 65.89494728118599

step: 560, loss is 0.204864, fps is 68.50401680627662

step: 561, loss is 0.19252382, fps is 66.45218352243134

step: 562, loss is 0.19239804, fps is 68.93459679614593

step: 563, loss is 0.19341904, fps is 69.4528274744347

step: 564, loss is 0.19762348, fps is 67.96137144338583

step: 565, loss is 0.20563163, fps is 62.54204395071853

step: 566, loss is 0.19447844, fps is 64.03743643621006

step: 567, loss is 0.18857725, fps is 65.06302232598183

step: 568, loss is 0.19925721, fps is 64.55748753512435

step: 569, loss is 0.19751391, fps is 69.45807501309282

step: 570, loss is 0.18666798, fps is 66.5495813185985

step: 571, loss is 0.19478597, fps is 65.55623024292653

step: 572, loss is 0.18499298, fps is 69.90958166048225

step: 573, loss is 0.18392548, fps is 65.10436568492679

step: 574, loss is 0.20971885, fps is 69.15288497406337

step: 575, loss is 0.19992614, fps is 68.64500185143133

step: 576, loss is 0.21209913, fps is 57.766267750451696

step: 577, loss is 0.19812879, fps is 68.18391986450362

step: 578, loss is 0.19206414, fps is 59.37823362912352

step: 579, loss is 0.19456972, fps is 57.92072554467204

step: 580, loss is 0.19644356, fps is 68.25035112252422

step: 581, loss is 0.19687083, fps is 68.97973229942501

step: 582, loss is 0.18875787, fps is 67.30404573262462

step: 583, loss is 0.20005877, fps is 69.20922557783618

step: 584, loss is 0.20136878, fps is 68.96052300315779

step: 585, loss is 0.20056972, fps is 68.32567596322914

step: 586, loss is 0.1956827, fps is 68.93983713445397

step: 587, loss is 0.19559959, fps is 68.65511448851022

step: 588, loss is 0.19571038, fps is 69.26215514118428

step: 589, loss is 0.1966508, fps is 66.30884775187364

step: 590, loss is 0.19261095, fps is 68.09598337500405

step: 591, loss is 0.1838328, fps is 67.27988947905575

step: 592, loss is 0.19177707, fps is 67.80323694960086

step: 593, loss is 0.20014365, fps is 70.06936472917226

step: 594, loss is 0.19058493, fps is 70.04376791959912

step: 595, loss is 0.19541696, fps is 66.07420797728372

step: 596, loss is 0.17765197, fps is 68.12494759335489

step: 597, loss is 0.19217896, fps is 66.55459731357257

step: 598, loss is 0.1973227, fps is 67.43775794550021

step: 599, loss is 0.19972578, fps is 68.56911180499927

step: 600, loss is 0.1975179, fps is 60.47456258605899

step: 1, loss is 0.19367504, fps is 64.17239918566324

step: 2, loss is 0.2143362, fps is 70.50390924582338

step: 3, loss is 0.20217693, fps is 70.04544942154902

step: 4, loss is 0.20252435, fps is 70.88610076538427

step: 5, loss is 0.19385016, fps is 70.00131847957608

step: 6, loss is 0.20342863, fps is 70.88018605967724

step: 7, loss is 0.21228078, fps is 70.03031880978793

step: 8, loss is 0.21602696, fps is 70.9476272686234

step: 9, loss is 0.21892227, fps is 70.70640593391774

step: 10, loss is 0.20446385, fps is 70.841427999886

step: 11, loss is 0.21790363, fps is 70.37090741313591

step: 12, loss is 0.1974517, fps is 70.9252825275235

step: 13, loss is 0.20571586, fps is 70.75508583851625

step: 14, loss is 0.20557365, fps is 70.91456504785836

step: 15, loss is 0.21484296, fps is 69.59219177116215

step: 16, loss is 0.1934391, fps is 70.74434515307118

step: 17, loss is 0.2069011, fps is 69.86700370527306

step: 18, loss is 0.19609192, fps is 70.45601804947648

step: 19, loss is 0.2018737, fps is 70.01139647506317

step: 20, loss is 0.19845156, fps is 69.92349442092552

step: 21, loss is 0.2004079, fps is 69.82338784942198

step: 22, loss is 0.19950165, fps is 70.39814702142927

step: 23, loss is 0.21018444, fps is 69.48936104118798

step: 24, loss is 0.2060488, fps is 70.0006613197815

step: 25, loss is 0.19453917, fps is 69.87857104181332

step: 26, loss is 0.20011501, fps is 70.33226189490541

step: 27, loss is 0.17904034, fps is 69.99635357779104

step: 28, loss is 0.2053832, fps is 70.16049471774846

step: 29, loss is 0.19335465, fps is 69.74443651132651

step: 30, loss is 0.19761968, fps is 70.22414763749454

step: 31, loss is 0.1893575, fps is 70.65801615127873

step: 32, loss is 0.19285603, fps is 69.9037559712422

step: 33, loss is 0.18815912, fps is 69.78628459476266

step: 34, loss is 0.188476, fps is 70.45712761869565

step: 35, loss is 0.20498799, fps is 70.6741635274703

step: 36, loss is 0.1988478, fps is 70.51605893565288

step: 37, loss is 0.19280083, fps is 69.68903067158476

step: 38, loss is 0.202113, fps is 70.11951551829345

step: 39, loss is 0.2013886, fps is 69.53673925094681

step: 40, loss is 0.18801898, fps is 70.58845110896534

step: 41, loss is 0.19712295, fps is 69.99971211074974

step: 42, loss is 0.19319129, fps is 69.51009006116287

step: 43, loss is 0.19470957, fps is 69.07259796000288

step: 44, loss is 0.18934856, fps is 70.6338460890753

step: 45, loss is 0.1943907, fps is 70.5029463397415

step: 46, loss is 0.18509507, fps is 70.62068965517241

step: 47, loss is 0.18375082, fps is 70.0243999148549

step: 48, loss is 0.19111353, fps is 69.76662352311298

step: 49, loss is 0.19282402, fps is 69.50793021124099

step: 50, loss is 0.1814244, fps is 70.42466867523615

step: 51, loss is 0.19700468, fps is 69.57963690655166

step: 52, loss is 0.19499761, fps is 70.08626887431882

step: 53, loss is 0.1883395, fps is 69.28661148560403

step: 54, loss is 0.1884946, fps is 70.10603730694658

step: 55, loss is 0.18161215, fps is 69.76067659957982

step: 56, loss is 0.19006227, fps is 69.94193189326874

step: 57, loss is 0.1947105, fps is 69.74755344181058

step: 58, loss is 0.19240859, fps is 70.06153743360171

step: 59, loss is 0.1925525, fps is 69.47101753833859

step: 60, loss is 0.19085784, fps is 69.86831302283694

step: 61, loss is 0.19440547, fps is 69.4992921536239

step: 62, loss is 0.19686688, fps is 70.4772538715362

step: 63, loss is 0.2010585, fps is 69.4621729576316

step: 64, loss is 0.187335, fps is 70.14450786695836

step: 65, loss is 0.18125981, fps is 69.53695540897043

step: 66, loss is 0.18176013, fps is 69.95921218192831

step: 67, loss is 0.19028473, fps is 69.61652069184079

step: 68, loss is 0.20674986, fps is 70.51250249545565

step: 69, loss is 0.19886324, fps is 67.04865430843103

step: 70, loss is 0.18498139, fps is 70.27488826616737

step: 71, loss is 0.19232151, fps is 69.93763137627624

step: 72, loss is 0.18949556, fps is 70.20020000753166

step: 73, loss is 0.20282555, fps is 69.65886611175294

step: 74, loss is 0.19781369, fps is 70.27245987092937

step: 75, loss is 0.18387535, fps is 69.39228658227638

step: 76, loss is 0.20801777, fps is 70.2419352812865

step: 77, loss is 0.1958344, fps is 69.56679811085264

step: 78, loss is 0.21111386, fps is 70.00372817105774

step: 79, loss is 0.18691091, fps is 69.55965948254864

step: 80, loss is 0.20766144, fps is 70.32069125225551

step: 81, loss is 0.18901998, fps is 69.7390006713181

step: 82, loss is 0.2076544, fps is 70.09249106726006

step: 83, loss is 0.2012309, fps is 69.77612620012601

step: 84, loss is 0.18014535, fps is 69.89705762881337

step: 85, loss is 0.18434487, fps is 69.65409425829226

step: 86, loss is 0.21195996, fps is 69.91708366542166

step: 87, loss is 0.19553809, fps is 70.55231062465766

step: 88, loss is 0.2069487, fps is 70.22547037525376

step: 89, loss is 0.19226992, fps is 70.399180917733

step: 90, loss is 0.18874653, fps is 70.2082052194109

step: 91, loss is 0.19202235, fps is 69.74037767064651

step: 92, loss is 0.20621419, fps is 70.20673623643259

step: 93, loss is 0.21013154, fps is 69.432563606209

step: 94, loss is 0.20522445, fps is 70.01446426708398

step: 95, loss is 0.18368092, fps is 69.63017254796169

step: 96, loss is 0.183544, fps is 70.25142080695764

step: 97, loss is 0.20324336, fps is 70.03492308655632

step: 98, loss is 0.19381273, fps is 70.4228950165855

step: 99, loss is 0.19083424, fps is 69.44391566420903

step: 100, loss is 0.19381082, fps is 69.9380686928311

step: 101, loss is 0.189211, fps is 69.8904333571825

step: 102, loss is 0.195552, fps is 70.63726607785493

step: 103, loss is 0.19375142, fps is 69.65293754378682

step: 104, loss is 0.19332835, fps is 70.37887779918346

step: 105, loss is 0.19743277, fps is 69.58981032800367

step: 106, loss is 0.17964754, fps is 70.8941133965838

step: 107, loss is 0.1909426, fps is 70.0006613197815

step: 108, loss is 0.1895783, fps is 70.67341924653762

step: 109, loss is 0.19628145, fps is 69.56838467168267

step: 110, loss is 0.1958672, fps is 70.47717985690086

step: 111, loss is 0.18848017, fps is 69.93850601485502

step: 112, loss is 0.19158675, fps is 70.14546100699272

step: 113, loss is 0.19713588, fps is 69.70582418586872

step: 114, loss is 0.18736508, fps is 69.96526594757573

step: 115, loss is 0.1904966, fps is 69.88024462144176

step: 116, loss is 0.18816775, fps is 70.31015569913743

step: 117, loss is 0.18873669, fps is 68.88025758429532

step: 118, loss is 0.18329899, fps is 70.88385455836003

step: 119, loss is 0.20174116, fps is 69.9519198796694

step: 120, loss is 0.18152097, fps is 70.94425216028958

step: 121, loss is 0.19333382, fps is 70.02242717220842

step: 122, loss is 0.18083358, fps is 70.19006758693111

step: 123, loss is 0.18503076, fps is 69.82389638711665

step: 124, loss is 0.17699735, fps is 70.33963371621968

step: 125, loss is 0.17667834, fps is 69.99898196747303

step: 126, loss is 0.17859212, fps is 70.66992133612746

step: 127, loss is 0.19326396, fps is 69.73653670296783

step: 128, loss is 0.19263038, fps is 70.41011253638075

step: 129, loss is 0.18470664, fps is 69.87682478331675

step: 130, loss is 0.18284889, fps is 70.87180234553625

step: 131, loss is 0.17798835, fps is 69.78367215364938

step: 132, loss is 0.18523093, fps is 70.26657354870527

step: 133, loss is 0.18279408, fps is 69.80886132964675

step: 134, loss is 0.1898699, fps is 70.63027775819009

step: 135, loss is 0.17548463, fps is 69.596449909101

step: 136, loss is 0.18269825, fps is 69.89858648521127

step: 137, loss is 0.18545943, fps is 69.52197163546707

step: 138, loss is 0.18816164, fps is 69.79855116930933

step: 139, loss is 0.18494451, fps is 69.41209617528843

step: 140, loss is 0.190962, fps is 69.89480086778921

step: 141, loss is 0.19535688, fps is 69.63421857068444

step: 142, loss is 0.1740748, fps is 70.16328216237466

step: 143, loss is 0.18425235, fps is 69.46375474977202

step: 144, loss is 0.19127503, fps is 70.35644724854562

step: 145, loss is 0.1974196, fps is 69.86954964559682

step: 146, loss is 0.18913695, fps is 70.03704271904036

step: 147, loss is 0.19107737, fps is 69.59955362606226

step: 148, loss is 0.17230915, fps is 69.4843965492249

step: 149, loss is 0.18663658, fps is 69.39931912851759

step: 150, loss is 0.21107845, fps is 70.38205169618267

step: 151, loss is 0.19351865, fps is 69.11549414140748

step: 152, loss is 0.17988534, fps is 70.00029623633819

step: 153, loss is 0.19454509, fps is 69.48597935368973

step: 154, loss is 0.1959014, fps is 69.24200236691763

step: 155, loss is 0.19286439, fps is 69.59962580882025

step: 156, loss is 0.20851992, fps is 69.20965383242957

step: 157, loss is 0.18562543, fps is 69.78418011299176

step: 158, loss is 0.19834842, fps is 69.42186158426452

step: 159, loss is 0.1946049, fps is 69.78432524559071

step: 160, loss is 0.18318366, fps is 69.43306646691346

step: 161, loss is 0.20995928, fps is 69.92604448001067

step: 162, loss is 0.18941757, fps is 69.48626714406711

step: 163, loss is 0.19541311, fps is 69.56124571778035

step: 164, loss is 0.19670305, fps is 69.25314926055455

step: 165, loss is 0.19540808, fps is 69.7724989187211

step: 166, loss is 0.18603447, fps is 69.29147620911078

step: 167, loss is 0.18621065, fps is 69.8066102601123

step: 168, loss is 0.2015312, fps is 69.31538050770166

step: 169, loss is 0.18155982, fps is 69.62670490269112

step: 170, loss is 0.20074546, fps is 69.44434682817847

step: 171, loss is 0.18557453, fps is 69.60078075331079

step: 172, loss is 0.17579952, fps is 69.19274180493936

step: 173, loss is 0.17937721, fps is 69.79463120333556

step: 174, loss is 0.18973541, fps is 69.58750120543644

step: 175, loss is 0.18689162, fps is 69.80123732614403

step: 176, loss is 0.18672206, fps is 69.54942279531024

step: 177, loss is 0.18473741, fps is 69.92429584802487

step: 178, loss is 0.18171592, fps is 69.25529330541472

step: 179, loss is 0.19384865, fps is 69.59312996018886

step: 180, loss is 0.18180601, fps is 69.28603921009893

step: 181, loss is 0.18526846, fps is 69.82280667255556

step: 182, loss is 0.19280072, fps is 69.50353893030471

step: 183, loss is 0.1838668, fps is 69.81837551706631

step: 184, loss is 0.1710403, fps is 69.44815567578307

step: 185, loss is 0.18000108, fps is 69.68056459760376

step: 186, loss is 0.18203431, fps is 69.52607712334742

step: 187, loss is 0.18609104, fps is 69.59168660293284

step: 188, loss is 0.19362879, fps is 69.23257318236918

step: 189, loss is 0.18569179, fps is 69.69149128400257

step: 190, loss is 0.18405257, fps is 69.2959838420219

step: 191, loss is 0.21112037, fps is 70.18111237534302

step: 192, loss is 0.19680752, fps is 69.27087735306657

step: 193, loss is 0.1934241, fps is 69.6544557394541

step: 194, loss is 0.20197909, fps is 69.7840349809965

step: 195, loss is 0.17747658, fps is 70.49405921769282

step: 196, loss is 0.20013875, fps is 69.36460081862157

step: 197, loss is 0.1986418, fps is 69.32841314689794

step: 198, loss is 0.19902292, fps is 69.54971111145657

step: 199, loss is 0.21214479, fps is 69.3122304952123

step: 200, loss is 0.19649132, fps is 70.22767504892265

step: 201, loss is 0.19936162, fps is 69.40104160203191

step: 202, loss is 0.21339023, fps is 69.13721151131908

step: 203, loss is 0.2032195, fps is 69.23485880946215

step: 204, loss is 0.22196229, fps is 70.46348982351805

step: 205, loss is 0.20277089, fps is 69.19003094066424

step: 206, loss is 0.19822967, fps is 69.72740704413546

step: 207, loss is 0.18430994, fps is 69.85013244791857

step: 208, loss is 0.212405, fps is 69.61327103889529

step: 209, loss is 0.19544895, fps is 69.87456932859513

step: 210, loss is 0.19939312, fps is 69.34338524617267

step: 211, loss is 0.20246226, fps is 69.80218116006095

step: 212, loss is 0.18935537, fps is 69.62005954789248

step: 213, loss is 0.20179173, fps is 69.5477650238566

step: 214, loss is 0.19758444, fps is 69.44786820101002

step: 215, loss is 0.1891662, fps is 69.49972400551367

step: 216, loss is 0.19606839, fps is 69.39931912851759

step: 217, loss is 0.18441465, fps is 69.69279403151235

step: 218, loss is 0.18844661, fps is 69.51736254265275

step: 219, loss is 0.18971425, fps is 69.81234714311721

step: 220, loss is 0.19136325, fps is 69.43364117378052

step: 221, loss is 0.1884115, fps is 70.03791984587431

step: 222, loss is 0.18995462, fps is 69.35664345782465

step: 223, loss is 0.18350026, fps is 69.93435167643118

step: 224, loss is 0.19757994, fps is 70.31752288684984

step: 225, loss is 0.18657717, fps is 69.87151378348523

step: 226, loss is 0.1902223, fps is 69.8661308541594

step: 227, loss is 0.1943684, fps is 69.79252621813991

step: 228, loss is 0.19377755, fps is 69.89560163727829

step: 229, loss is 0.19984749, fps is 69.62663266375401

step: 230, loss is 0.1827083, fps is 69.47626782617698

step: 231, loss is 0.18651795, fps is 69.65474492708495

step: 232, loss is 0.18115115, fps is 70.31199735133754

step: 233, loss is 0.17792748, fps is 70.52954703100367

step: 234, loss is 0.20597878, fps is 69.56340888241606

step: 235, loss is 0.18199413, fps is 70.25053832440575

step: 236, loss is 0.19773506, fps is 69.8020359531816

step: 237, loss is 0.19212726, fps is 69.45440884096834

step: 238, loss is 0.18634205, fps is 69.54574697396784

step: 239, loss is 0.20146497, fps is 70.18801209040612

step: 240, loss is 0.18580662, fps is 69.58793415424498

step: 241, loss is 0.19667995, fps is 69.37019356999468

step: 242, loss is 0.17715812, fps is 69.32440258212203

step: 243, loss is 0.20005392, fps is 69.74371168369845

step: 244, loss is 0.18810242, fps is 69.40671201466967

step: 245, loss is 0.18233062, fps is 69.70343495949737

step: 246, loss is 0.18338563, fps is 70.2617916379619

step: 247, loss is 0.17885973, fps is 70.51050215548717

step: 248, loss is 0.182959, fps is 69.02450097557514

step: 249, loss is 0.1815097, fps is 70.61637957341134

step: 250, loss is 0.17963871, fps is 69.15031973934494

step: 251, loss is 0.1823033, fps is 70.7239915521813

step: 252, loss is 0.18389815, fps is 69.33693713799224

step: 253, loss is 0.18518767, fps is 69.81612383390743

step: 254, loss is 0.18479988, fps is 70.1529403893758

step: 255, loss is 0.18846427, fps is 70.63964539581923

step: 256, loss is 0.17882857, fps is 70.37784449914058

step: 257, loss is 0.18367632, fps is 69.26029659307055

step: 258, loss is 0.18732074, fps is 69.53335295068446

step: 259, loss is 0.17900395, fps is 70.37533518249553

step: 260, loss is 0.18667436, fps is 69.28847144628581

step: 261, loss is 0.19725609, fps is 70.2155510564946

step: 262, loss is 0.19192404, fps is 69.11485350822265

step: 263, loss is 0.18053809, fps is 69.40642488289859

step: 264, loss is 0.1947874, fps is 70.28003996322043

step: 265, loss is 0.20053503, fps is 69.04985543631481

step: 266, loss is 0.17599052, fps is 69.4174811713144

step: 267, loss is 0.1918388, fps is 69.49684509428472

step: 268, loss is 0.20449333, fps is 69.5416391628334

step: 269, loss is 0.19026785, fps is 68.79897973619947

step: 270, loss is 0.19431962, fps is 69.27166388997273

step: 271, loss is 0.19710907, fps is 69.20508605665611

step: 272, loss is 0.1891786, fps is 69.18182830327946

step: 273, loss is 0.19002509, fps is 69.54862993823322

step: 274, loss is 0.19750054, fps is 69.15929889359738

step: 275, loss is 0.17991371, fps is 69.40362547262694

step: 276, loss is 0.18944578, fps is 69.45347438840373

step: 277, loss is 0.18248662, fps is 69.30965342521071

step: 278, loss is 0.18647996, fps is 68.58473899850891

step: 279, loss is 0.20536037, fps is 69.3064323571815

step: 280, loss is 0.18986696, fps is 69.64252884179328

step: 281, loss is 0.18925105, fps is 69.32554841042328

step: 282, loss is 0.19662753, fps is 69.77191858869324

step: 283, loss is 0.19035797, fps is 69.35241461272155

step: 284, loss is 0.19136418, fps is 69.43040857140197

step: 285, loss is 0.1909382, fps is 69.03635914739527

step: 286, loss is 0.1784548, fps is 69.6628431557771

step: 287, loss is 0.1853003, fps is 69.37542862872921

step: 288, loss is 0.18769792, fps is 69.4162604989863

step: 289, loss is 0.1889007, fps is 69.24986198292612

step: 290, loss is 0.1784301, fps is 69.32468903564636

step: 291, loss is 0.18484859, fps is 69.29662783745633

step: 292, loss is 0.18853351, fps is 70.23414289033408

step: 293, loss is 0.18666239, fps is 70.59758485056012

step: 294, loss is 0.1814546, fps is 70.17436077614512

step: 295, loss is 0.18761739, fps is 70.35327566035284

step: 296, loss is 0.19014795, fps is 70.19190295342321

step: 297, loss is 0.18259278, fps is 69.60460678072985

step: 298, loss is 0.1818887, fps is 70.06248832011092

step: 299, loss is 0.19829768, fps is 69.83813845065146

step: 300, loss is 0.18218285, fps is 69.6560463011401

step: 301, loss is 0.19027321, fps is 70.34892441162627

step: 302, loss is 0.18003532, fps is 69.71444124139722

step: 303, loss is 0.18290615, fps is 69.71002382916133

step: 304, loss is 0.17882168, fps is 70.2292918975939

step: 305, loss is 0.19078223, fps is 70.51287294123827

step: 306, loss is 0.18354687, fps is 70.35534084809447

step: 307, loss is 0.18240924, fps is 69.64216748441301

step: 308, loss is 0.18403871, fps is 70.32444945115402

step: 309, loss is 0.19147244, fps is 70.30661999733897

step: 310, loss is 0.18739817, fps is 70.25075894296518

step: 311, loss is 0.18430272, fps is 69.57617429277022

step: 312, loss is 0.18228805, fps is 69.04026542527198

step: 313, loss is 0.19350831, fps is 70.59565394394541

step: 314, loss is 0.17973559, fps is 70.33963371621968

step: 315, loss is 0.17615716, fps is 69.01364047716989

step: 316, loss is 0.19074447, fps is 69.10389340253107

step: 317, loss is 0.19617413, fps is 70.42126924130318

step: 318, loss is 0.19086441, fps is 70.3944547825193

step: 319, loss is 0.18149643, fps is 70.42917712644933

step: 320, loss is 0.18122227, fps is 69.28339255885213

step: 321, loss is 0.18625672, fps is 69.30900918765285

step: 322, loss is 0.1848432, fps is 70.1778831875583

step: 323, loss is 0.18907222, fps is 69.3684009232738

step: 324, loss is 0.18412183, fps is 70.48369373996317

step: 325, loss is 0.1869779, fps is 70.48280541228046

step: 326, loss is 0.18025704, fps is 70.42806843738555

step: 327, loss is 0.19132885, fps is 70.35290688958568

step: 328, loss is 0.18308155, fps is 70.25936414774922

step: 329, loss is 0.18300784, fps is 69.40362547262694

step: 330, loss is 0.17818263, fps is 69.47533278533405

step: 331, loss is 0.18123122, fps is 69.70929971662913

step: 332, loss is 0.18244785, fps is 69.00696970881054

step: 333, loss is 0.19282264, fps is 70.47458944259968

step: 334, loss is 0.18224072, fps is 70.36205354568473

step: 335, loss is 0.17357248, fps is 70.53599646000212

step: 336, loss is 0.18127173, fps is 70.34486864752337

step: 337, loss is 0.17413387, fps is 70.6556355956294

step: 338, loss is 0.19465071, fps is 69.09023372269958

step: 339, loss is 0.17860252, fps is 69.76103918784344

step: 340, loss is 0.18769757, fps is 70.24031785040684

step: 341, loss is 0.18948242, fps is 70.59624805780764

step: 342, loss is 0.17753048, fps is 70.44766041784327

step: 343, loss is 0.18675937, fps is 70.39777778010891

step: 344, loss is 0.18667135, fps is 70.31207101943228

step: 345, loss is 0.18270096, fps is 70.46674534938909

step: 346, loss is 0.18307929, fps is 70.4488436788125

step: 347, loss is 0.17752334, fps is 69.59341863882467

step: 348, loss is 0.1877034, fps is 69.74276943030489

step: 349, loss is 0.18806332, fps is 70.50598328672078

step: 350, loss is 0.17126973, fps is 70.40989091580572

step: 351, loss is 0.19062021, fps is 70.54348520779723

step: 352, loss is 0.1745254, fps is 69.1849664739503

step: 353, loss is 0.16427618, fps is 69.62865541065624

step: 354, loss is 0.18127082, fps is 70.20739727116562

step: 355, loss is 0.18832701, fps is 69.67448765906994

step: 356, loss is 0.17684582, fps is 69.4028359361992

step: 357, loss is 0.18362449, fps is 69.6602399483482

step: 358, loss is 0.18214448, fps is 69.43213258854531

step: 359, loss is 0.18431036, fps is 69.58692394873853

step: 360, loss is 0.18787657, fps is 70.37371160236869

step: 361, loss is 0.1765191, fps is 69.8515138393898

step: 362, loss is 0.18580595, fps is 69.45304311108535

step: 363, loss is 0.19074209, fps is 69.76437517672693

step: 364, loss is 0.19109797, fps is 69.32067890171709

step: 365, loss is 0.18440883, fps is 69.5649952886566

step: 366, loss is 0.1843313, fps is 69.38862735163806

step: 367, loss is 0.18238525, fps is 69.80181814399533

step: 368, loss is 0.17919447, fps is 70.41528407922405

step: 369, loss is 0.20019308, fps is 70.51146526797122

step: 370, loss is 0.18030854, fps is 69.31903201682444

step: 371, loss is 0.18875042, fps is 69.8663490648932

step: 372, loss is 0.19146183, fps is 70.28541325192055

step: 373, loss is 0.18320927, fps is 69.68519535756452

step: 374, loss is 0.18371247, fps is 69.66588047702886

step: 375, loss is 0.20722918, fps is 70.44307565699698

step: 376, loss is 0.18154223, fps is 70.39534088453541

step: 377, loss is 0.19040044, fps is 70.611104154983

step: 378, loss is 0.18797459, fps is 69.52888642076101

step: 379, loss is 0.19695185, fps is 69.27245044474058

step: 380, loss is 0.18660448, fps is 69.34926123340642

step: 381, loss is 0.17082225, fps is 69.5962333810383

step: 382, loss is 0.18680243, fps is 69.54762087355198

step: 383, loss is 0.19044124, fps is 69.83951936776005

step: 384, loss is 0.17603703, fps is 70.16665673380237

step: 385, loss is 0.18087484, fps is 70.60218974646536

step: 386, loss is 0.18096687, fps is 69.35155457174096

step: 387, loss is 0.18787175, fps is 70.40442471655733

step: 388, loss is 0.17357549, fps is 69.55281066100645

step: 389, loss is 0.18379335, fps is 69.55230606434655

step: 390, loss is 0.17764941, fps is 69.43112690147359

step: 391, loss is 0.17714255, fps is 69.57588575717801

step: 392, loss is 0.18492624, fps is 70.48487821170609

step: 393, loss is 0.17936653, fps is 70.59788192249145

step: 394, loss is 0.18462728, fps is 70.07163277708052

step: 395, loss is 0.1740482, fps is 69.56434629553998

step: 396, loss is 0.18625116, fps is 69.40046743469358

step: 397, loss is 0.18788105, fps is 69.82767432762022

step: 398, loss is 0.18959564, fps is 69.99284936524558

step: 399, loss is 0.1789565, fps is 69.30142240195299

step: 400, loss is 0.16744089, fps is 69.41482446885537

step: 401, loss is 0.18176527, fps is 70.43494287231653

step: 402, loss is 0.17829676, fps is 70.28865234336132

step: 403, loss is 0.1706797, fps is 69.51837072948392

step: 404, loss is 0.16896811, fps is 69.23107332160033

step: 405, loss is 0.180145, fps is 70.28136480740632

step: 406, loss is 0.1809548, fps is 70.51013173461432

step: 407, loss is 0.16953304, fps is 69.40577884509973

step: 408, loss is 0.17895913, fps is 70.34169810312136

step: 409, loss is 0.16903289, fps is 70.48857994250321

step: 410, loss is 0.1929925, fps is 70.6170483447172

step: 411, loss is 0.16995345, fps is 69.70408655044622

step: 412, loss is 0.17673942, fps is 69.71074795673726

step: 413, loss is 0.17982236, fps is 70.4054587972643

step: 414, loss is 0.17882597, fps is 69.63530240661437

step: 415, loss is 0.1776817, fps is 70.45838517272624

step: 416, loss is 0.19075735, fps is 69.42667349461782

step: 417, loss is 0.17726746, fps is 69.54077442232305

step: 418, loss is 0.18812174, fps is 69.41417827468041

step: 419, loss is 0.18050829, fps is 69.61413758334959

step: 420, loss is 0.16705942, fps is 69.31645444103819

step: 421, loss is 0.16946855, fps is 69.50721029109421

step: 422, loss is 0.17178112, fps is 69.36660836920075

step: 423, loss is 0.18103254, fps is 69.72327774562781

step: 424, loss is 0.16929485, fps is 69.3591523384693

step: 425, loss is 0.17456415, fps is 69.34259707972932

step: 426, loss is 0.18445322, fps is 69.42193339891857

step: 427, loss is 0.17449851, fps is 69.41008599094376

step: 428, loss is 0.17216836, fps is 70.34059216649756

step: 429, loss is 0.17523609, fps is 69.37291857046571

step: 430, loss is 0.1720696, fps is 70.4197174349202

step: 431, loss is 0.17059775, fps is 70.59565394394541

step: 432, loss is 0.18311557, fps is 70.42008690642712

step: 433, loss is 0.17733765, fps is 70.4013226566852

step: 434, loss is 0.18517366, fps is 70.50976131763342

step: 435, loss is 0.17439452, fps is 70.73040204553551

step: 436, loss is 0.17872009, fps is 70.2636307573754

step: 437, loss is 0.17677598, fps is 69.25700863689043

step: 438, loss is 0.18619573, fps is 69.02251317785606

step: 439, loss is 0.19339608, fps is 69.75436816710565

step: 440, loss is 0.18103583, fps is 69.6160151703448

step: 441, loss is 0.18457231, fps is 69.66183077421337

step: 442, loss is 0.18501367, fps is 69.3600125679039

step: 443, loss is 0.19528261, fps is 69.70010479600093

step: 444, loss is 0.17537686, fps is 69.73784113511142

step: 445, loss is 0.18086517, fps is 69.8611851097696

step: 446, loss is 0.18660696, fps is 69.23921620416989

step: 447, loss is 0.17328294, fps is 69.41188079282514

step: 448, loss is 0.18202436, fps is 69.88119059321399

step: 449, loss is 0.16802907, fps is 69.617603976912

step: 450, loss is 0.18673122, fps is 69.39156905565483

step: 451, loss is 0.17757207, fps is 69.83821112913783

step: 452, loss is 0.18896224, fps is 69.30557345626993

step: 453, loss is 0.18680067, fps is 69.5762464270422

step: 454, loss is 0.18033099, fps is 69.38999054931828

step: 455, loss is 0.17123643, fps is 69.48669883410301

step: 456, loss is 0.18025276, fps is 69.35492318742475

step: 457, loss is 0.17811154, fps is 69.71067554330268

step: 458, loss is 0.17330022, fps is 69.32569164162422

step: 459, loss is 0.17820607, fps is 69.7250163380188

step: 460, loss is 0.1816561, fps is 69.85195007962695

step: 461, loss is 0.17510836, fps is 69.6753557282501

step: 462, loss is 0.16184261, fps is 69.66060149330112

step: 463, loss is 0.18588942, fps is 69.19217107901815

step: 464, loss is 0.18410933, fps is 69.58598592703044

step: 465, loss is 0.16359787, fps is 69.03245331152566

step: 466, loss is 0.18572089, fps is 69.62034844998475

step: 467, loss is 0.18354133, fps is 69.39623324395397

step: 468, loss is 0.19370733, fps is 69.37951682670437

step: 469, loss is 0.18033782, fps is 69.26644447907883

step: 470, loss is 0.17138931, fps is 69.61399315777572

step: 471, loss is 0.17708299, fps is 70.15484715372023

step: 472, loss is 0.18366168, fps is 69.38152524078723

step: 473, loss is 0.18708016, fps is 69.24343125530736

step: 474, loss is 0.17041416, fps is 69.41101927633831

step: 475, loss is 0.17459325, fps is 69.85216820178886

step: 476, loss is 0.18896292, fps is 69.73595697101196

step: 477, loss is 0.18288092, fps is 69.20530015767649

step: 478, loss is 0.18579835, fps is 69.89472807147276

step: 479, loss is 0.18039143, fps is 69.31702721909137

step: 480, loss is 0.17739305, fps is 69.82760167106284

step: 481, loss is 0.17221609, fps is 69.36517439264392

step: 482, loss is 0.17525038, fps is 69.78396241522525

step: 483, loss is 0.18924488, fps is 69.27745621179791

step: 484, loss is 0.17053685, fps is 70.10596406995462

step: 485, loss is 0.19528861, fps is 70.19351815533625

step: 486, loss is 0.17420055, fps is 69.88781311280444

step: 487, loss is 0.20026511, fps is 69.36890285501643

step: 488, loss is 0.17239158, fps is 70.02228104754522

step: 489, loss is 0.18706237, fps is 69.26129733735291

step: 490, loss is 0.18225953, fps is 69.93770426198893

step: 491, loss is 0.18654183, fps is 69.26644447907883

step: 492, loss is 0.17876285, fps is 70.06812102329796

step: 493, loss is 0.19089577, fps is 69.3341433301822

step: 494, loss is 0.19040309, fps is 69.73479753601623

step: 495, loss is 0.1768135, fps is 68.83256919785262

step: 496, loss is 0.18520208, fps is 69.58035832780358

step: 497, loss is 0.19270387, fps is 69.30893760644126

step: 498, loss is 0.18604839, fps is 69.42803818761924

step: 499, loss is 0.17067409, fps is 69.12937744393122

step: 500, loss is 0.18110414, fps is 70.35851262249518

step: 501, loss is 0.17833704, fps is 69.79731705845768

step: 502, loss is 0.17029409, fps is 69.41453726995896

step: 503, loss is 0.18072274, fps is 69.03415762191766

step: 504, loss is 0.19049469, fps is 69.26387081259217

step: 505, loss is 0.179464, fps is 68.86011443018856

step: 506, loss is 0.17855625, fps is 69.24307402768119

step: 507, loss is 0.18362856, fps is 68.91406801156289

step: 508, loss is 0.18012898, fps is 69.44599967299497

step: 509, loss is 0.18362099, fps is 69.14184156761252

step: 510, loss is 0.18679431, fps is 69.18575106110326

step: 511, loss is 0.18660904, fps is 69.2387161474105

step: 512, loss is 0.177601, fps is 69.19994802955716

step: 513, loss is 0.18354018, fps is 69.05070801067623

step: 514, loss is 0.1743867, fps is 68.98760340348717

step: 515, loss is 0.19495323, fps is 69.2023742250564

step: 516, loss is 0.19025977, fps is 69.14319508581484

step: 517, loss is 0.19108777, fps is 69.83167067113835

step: 518, loss is 0.18433903, fps is 69.33256743535411

step: 519, loss is 0.18003678, fps is 70.03302283547231

step: 520, loss is 0.17292996, fps is 69.34066256532763

step: 521, loss is 0.18066026, fps is 69.12126037454449

step: 522, loss is 0.17587551, fps is 69.1452610841905

step: 523, loss is 0.1758503, fps is 69.24893302624301

step: 524, loss is 0.17668809, fps is 69.01598264852674

step: 525, loss is 0.17399004, fps is 69.28010240951417

step: 526, loss is 0.1853793, fps is 68.87594525136913

step: 527, loss is 0.17734334, fps is 69.14348004376791

step: 528, loss is 0.18028492, fps is 68.96420809025206

step: 529, loss is 0.17285137, fps is 69.08482825783047

step: 530, loss is 0.17351027, fps is 69.31051242725165

step: 531, loss is 0.18749519, fps is 69.26608701396286

step: 532, loss is 0.16375268, fps is 69.21479330097549

step: 533, loss is 0.17358828, fps is 69.09279450131064

step: 534, loss is 0.16861154, fps is 69.098699242074

step: 535, loss is 0.17673618, fps is 68.75464391130491

step: 536, loss is 0.17101428, fps is 69.00491196609232

step: 537, loss is 0.19742298, fps is 68.65188373891587

step: 538, loss is 0.17133681, fps is 69.00065495695483

step: 539, loss is 0.1708824, fps is 68.82402759161053

step: 540, loss is 0.17575327, fps is 68.87750045672786

step: 541, loss is 0.17418636, fps is 68.73795349790024

step: 542, loss is 0.17624871, fps is 68.9593183485654

step: 543, loss is 0.17677875, fps is 68.91654497468602

step: 544, loss is 0.17307973, fps is 69.03749547354907

step: 545, loss is 0.17364094, fps is 69.13991823816431

step: 546, loss is 0.17342387, fps is 69.1169178132277

step: 547, loss is 0.17490347, fps is 69.37012186234692

step: 548, loss is 0.17638564, fps is 68.89305637397315

step: 549, loss is 0.16612378, fps is 69.03635914739527

step: 550, loss is 0.176801, fps is 68.95109947579427

step: 551, loss is 0.16379231, fps is 70.11577919058838

step: 552, loss is 0.19104794, fps is 69.05980344738873

step: 553, loss is 0.18471941, fps is 69.03834774271336

step: 554, loss is 0.17454977, fps is 69.76314227410279

step: 555, loss is 0.16667514, fps is 69.11819916801332

step: 556, loss is 0.18227948, fps is 69.69402444888239

step: 557, loss is 0.1726549, fps is 69.10389340253107

step: 558, loss is 0.17533588, fps is 69.7358845051952

step: 559, loss is 0.16754001, fps is 69.13336547455692

step: 560, loss is 0.16701117, fps is 70.11028531430541

step: 561, loss is 0.16605158, fps is 69.09450179251324

step: 562, loss is 0.17228624, fps is 69.75596329512334

step: 563, loss is 0.17074105, fps is 69.12503386267257

step: 564, loss is 0.16969104, fps is 69.19687984945737

step: 565, loss is 0.17044125, fps is 70.25348001913657

step: 566, loss is 0.16937058, fps is 69.3075060132586

step: 567, loss is 0.16636558, fps is 67.33004319188534

step: 568, loss is 0.17539716, fps is 69.36646496887707

step: 569, loss is 0.18332323, fps is 69.00845987497853

step: 570, loss is 0.17509414, fps is 69.31452138499182

step: 571, loss is 0.16505636, fps is 69.08589505890056

step: 572, loss is 0.17292598, fps is 69.3327106955601

step: 573, loss is 0.17570162, fps is 69.1413429163696

step: 574, loss is 0.1723394, fps is 69.31609645956183

step: 575, loss is 0.17945611, fps is 69.24335980948726

step: 576, loss is 0.18816458, fps is 68.49695478879492

step: 577, loss is 0.17495555, fps is 69.43550903680527

step: 578, loss is 0.17591865, fps is 69.37370742634798

step: 579, loss is 0.16800249, fps is 69.33851323147118

step: 580, loss is 0.17121768, fps is 69.49662918555738

step: 581, loss is 0.2024172, fps is 69.39250184315644

step: 582, loss is 0.2106464, fps is 69.6281497129126

step: 583, loss is 0.18907446, fps is 69.3421671783119

step: 584, loss is 0.20788649, fps is 69.5070663088544

step: 585, loss is 0.19161652, fps is 69.35757530659026

step: 586, loss is 0.19024792, fps is 69.54884617018872

step: 587, loss is 0.19265306, fps is 69.48374905909066

step: 588, loss is 0.18040131, fps is 69.56463473543016

step: 589, loss is 0.19323124, fps is 69.50152344615742

step: 590, loss is 0.19446546, fps is 66.12329626910893

step: 591, loss is 0.18283843, fps is 69.52787792890237

step: 592, loss is 0.18336926, fps is 69.43867026472195

step: 593, loss is 0.19956985, fps is 69.52254781504266

step: 594, loss is 0.18710104, fps is 69.30585975420841

step: 595, loss is 0.19449353, fps is 69.26079696159681

step: 596, loss is 0.18978435, fps is 69.42731992145714

step: 597, loss is 0.19844182, fps is 69.54012588105338

step: 598, loss is 0.18141271, fps is 69.31509413109875

step: 599, loss is 0.1852767, fps is 69.5264372769957

step: 600, loss is 0.1861261, fps is 69.44017913406806

============== End Training ==============

* + 1. 模型验证和测试
       1. 定义类用于评估模型性能

class dice\_coeff(nn.Metric):

    def \_\_init\_\_(self):

        super(dice\_coeff, self).\_\_init\_\_()

        self.clear()

    def clear(self):

        self.\_dice\_coeff\_sum = 0

        self.\_samples\_num = 0

    def update(self, \*inputs):

        if len(inputs) != 2:

            raise ValueError('Mean dice coeffcient need 2 inputs (y\_pred, y), but got {}'.format(len(inputs)))

        y\_pred = self.\_convert\_data(inputs[0])

        y = self.\_convert\_data(inputs[1])

        self.\_samples\_num += y.shape[0]

        y\_pred = y\_pred.transpose(0, 2, 3, 1)

        y = y.transpose(0, 2, 3, 1)

        y\_pred = softmax(y\_pred, axis=3)

        inter = np.dot(y\_pred.flatten(), y.flatten())

        union = np.dot(y\_pred.flatten(), y\_pred.flatten()) + np.dot(y.flatten(), y.flatten())

        single\_dice\_coeff = 2 \* float(inter) / float(union + 1e-6)

        print("single dice coeff is:", single\_dice\_coeff)

        self.\_dice\_coeff\_sum += single\_dice\_coeff

    def eval(self):

        if self.\_samples\_num == 0:

            raise RuntimeError('Total samples num must not be 0.')

        return self.\_dice\_coeff\_sum / float(self.\_samples\_num)

* + - 1. 测试模型效果

def test\_net(data\_dir, ckpt\_path, cross\_valid\_ind=1, cfg=None):

    net = UNet(n\_channels=cfg['num\_channels'], n\_classes=cfg['num\_classes'])

    param\_dict = load\_checkpoint(ckpt\_path)

    load\_param\_into\_net(net, param\_dict)

    criterion = CrossEntropyWithLogits()

    \_, valid\_dataset = create\_dataset(data\_dir, 1, 1, False, cross\_valid\_ind, False)

    model = Model(net, loss\_fn=criterion, metrics={"dice\_coeff": dice\_coeff()})

    print("============== Starting Evaluating ============")

    dice\_score = model.eval(valid\_dataset, dataset\_sink\_mode=False)

    print("Cross valid dice coeff is:", dice\_score)

ckpt\_path = './ckpt\_2/ckpt\_unet\_medical\_adam-2\_600.ckpt'

test\_net(data\_dir=data\_url, ckpt\_path=ckpt\_path, cross\_valid\_ind=cfg\_unet['cross\_valid\_ind'],

         cfg=cfg\_unet)

def test\_net(data\_dir, ckpt\_path, cross\_valid\_ind=1, cfg=None):

    net = UNet(n\_channels=cfg['num\_channels'], n\_classes=cfg['num\_classes'])

    param\_dict = load\_checkpoint(ckpt\_path)

    load\_param\_into\_net(net, param\_dict)

    criterion = CrossEntropyWithLogits()

    \_, valid\_dataset = create\_dataset(data\_dir, 1, 1, False, cross\_valid\_ind, False)

    model = Model(net, loss\_fn=criterion, metrics={"dice\_coeff": dice\_coeff()})

    print("============== Starting Evaluating ============")

    dice\_score = model.eval(valid\_dataset, dataset\_sink\_mode=False)

    print("Cross valid dice coeff is:", dice\_score)

    testimage=np.array([np.array(p) for p in ImageSequence.Iterator(Image.open("./data/test-volume.tif"))])

    testdata=testimage[10]

    image = Image.fromarray(testdata)

    image = image.resize((388, 388))

    testdata = np.asarray(image)

    testdata = np.pad(testdata, ((92, 92),(92, 92) ),  'symmetric')

    testdata = testdata/127.5-1

    testdata = testdata.astype(np.float32)

    testdata = testdata.reshape(1,1,572,572)

    output = model.predict(Tensor(testdata))

    pred = np.argmax(output.asnumpy(), axis=1)

    pred = pred.reshape(388, 388)

    plt.figure(figsize=(10,10))

    plt.subplot(2,2,1)

    plt.imshow(testimage[10],cmap='gray')

    plt.subplot(2,2,2)

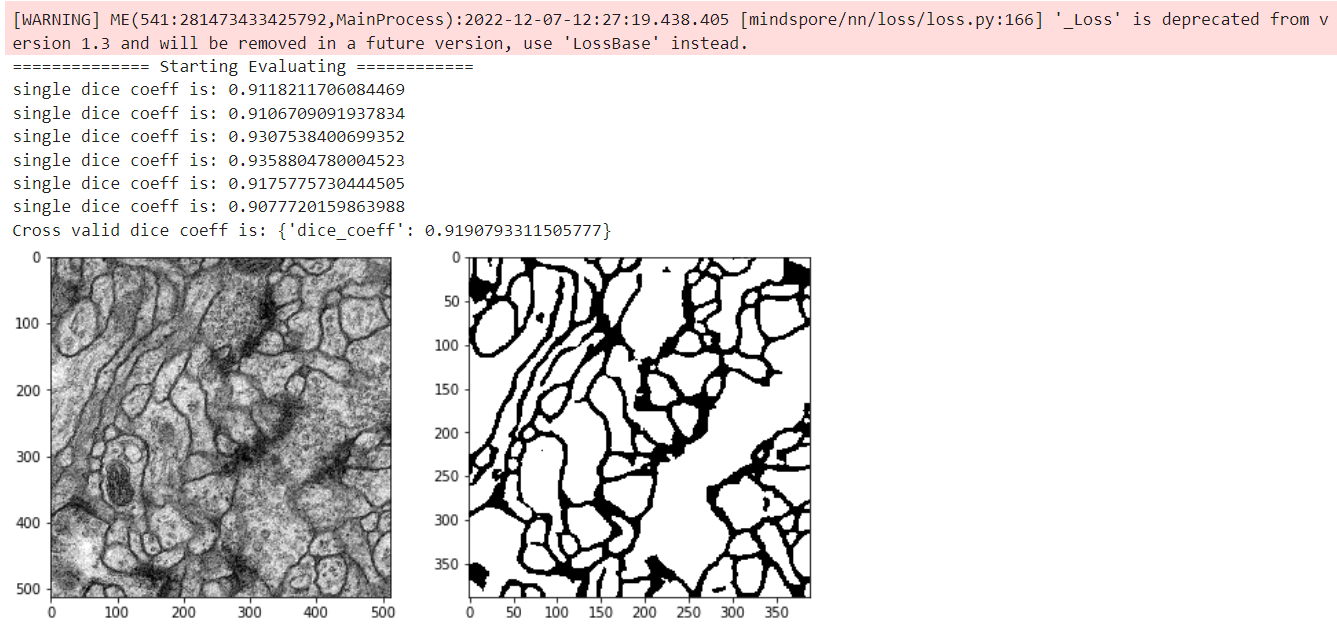
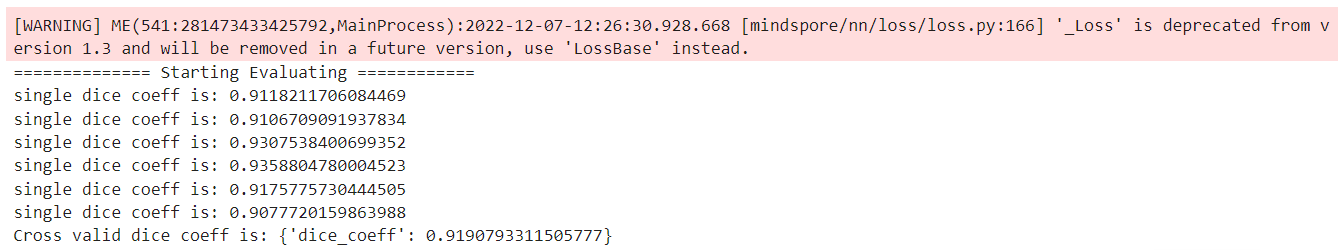
    plt.imshow(pred,cmap='gray')

    plt.show()

ckpt\_path = './ckpt\_2/ckpt\_unet\_medical\_adam-2\_600.ckpt'

test\_net(data\_dir=data\_url, ckpt\_path=ckpt\_path, cross\_valid\_ind=cfg\_unet['cross\_valid\_ind'],

         cfg=cfg\_unet)



2 基于DeepLabv3的语义分割实验

步骤 1准备数据集和预训练模型

import moxing as mox

mox.file.copy\_parallel(src\_url="obs://ascend-professional-construction-dataset/deep-learning/deeplabv3-mindspore/VOC2012", dst\_url="./VOC2012")

import moxing as mox

mox.file.copy\_parallel(src\_url="obs://ascend-professional-construction-dataset/deep-learning/deeplabv3-mindspore/ckpt", dst\_url="./ckpt")

步骤 2构建数据读取相关函数

import os

import numpy as np

import scipy.io

import pickle

from PIL import Image

import shutil

import cv2

from mindspore.mindrecord import FileWriter

import mindspore.dataset as de

cv2.setNumThreads(0)

class SegDataset:

    def \_\_init\_\_(self,

                 image\_mean,

                 image\_std,

                 data\_file='',

                 batch\_size=32,

                 crop\_size=512,

                 max\_scale=2.0,

                 min\_scale=0.5,

                 ignore\_label=255,

                 num\_classes=21,

                 num\_readers=2,

                 num\_parallel\_calls=4,

                 shard\_id=None,

                 shard\_num=None):

        self.data\_file = data\_file

        self.batch\_size = batch\_size

        self.crop\_size = crop\_size

        self.image\_mean = np.array(image\_mean, dtype=np.float32)

        self.image\_std = np.array(image\_std, dtype=np.float32)

        self.max\_scale = max\_scale

        self.min\_scale = min\_scale

        self.ignore\_label = ignore\_label

        self.num\_classes = num\_classes

        self.num\_readers = num\_readers

        self.num\_parallel\_calls = num\_parallel\_calls

        self.shard\_id = shard\_id

        self.shard\_num = shard\_num

        self.voc\_img\_dir = os.path.join(self.data\_file,'JPEGImages')

        self.voc\_anno\_dir = os.path.join(self.data\_file,'SegmentationClass')

        self.voc\_train\_lst = os.path.join(self.data\_file,'ImageSets/Segmentation/train.txt')

        self.voc\_val\_lst = os.path.join(self.data\_file,'ImageSets/Segmentation/val.txt')

        self.voc\_anno\_gray\_dir = os.path.join(self.data\_file,'SegmentationClassGray')

        self.mindrecord\_save =  os.path.join(self.data\_file,'VOC\_mindrecord')

        assert max\_scale > min\_scale

    def preprocess\_(self, image, label):

        # bgr image

        image\_out = cv2.imdecode(np.frombuffer(image, dtype=np.uint8), cv2.IMREAD\_COLOR)

        label\_out = cv2.imdecode(np.frombuffer(label, dtype=np.uint8), cv2.IMREAD\_GRAYSCALE)

        sc = np.random.uniform(self.min\_scale, self.max\_scale)

        new\_h, new\_w = int(sc \* image\_out.shape[0]), int(sc \* image\_out.shape[1])

        image\_out = cv2.resize(image\_out, (new\_w, new\_h), interpolation=cv2.INTER\_CUBIC)

        label\_out = cv2.resize(label\_out, (new\_w, new\_h), interpolation=cv2.INTER\_NEAREST)

        image\_out = (image\_out - self.image\_mean) / self.image\_std

        h\_, w\_ = max(new\_h, self.crop\_size), max(new\_w, self.crop\_size)

        pad\_h, pad\_w = h\_ - new\_h, w\_ - new\_w

        if pad\_h > 0 or pad\_w > 0:

            image\_out = cv2.copyMakeBorder(image\_out, 0, pad\_h, 0, pad\_w, cv2.BORDER\_CONSTANT, value=0)

            label\_out = cv2.copyMakeBorder(label\_out, 0, pad\_h, 0, pad\_w, cv2.BORDER\_CONSTANT, value=self.ignore\_label)

        offset\_h = np.random.randint(0, h\_ - self.crop\_size + 1)

        offset\_w = np.random.randint(0, w\_ - self.crop\_size + 1)

        image\_out = image\_out[offset\_h: offset\_h + self.crop\_size, offset\_w: offset\_w + self.crop\_size, :]

        label\_out = label\_out[offset\_h: offset\_h + self.crop\_size, offset\_w: offset\_w+self.crop\_size]

        if np.random.uniform(0.0, 1.0) > 0.5:

            image\_out = image\_out[:, ::-1, :]

            label\_out = label\_out[:, ::-1]

        image\_out = image\_out.transpose((2, 0, 1))

        image\_out = image\_out.copy()

        label\_out = label\_out.copy()

        return image\_out, label\_out

    def get\_gray\_dataset(self):

        if os.path.exists(self.voc\_anno\_gray\_dir):

            print('the gray file is already exists！')

            return

        os.makedirs(self.voc\_anno\_gray\_dir)

        # convert voc color png to gray png

        print('converting voc color png to gray png ...')

        for ann in os.listdir(self.voc\_anno\_dir):

            ann\_im = Image.open(os.path.join(self.voc\_anno\_dir, ann))

            ann\_im = Image.fromarray(np.array(ann\_im))

            ann\_im.save(os.path.join(self.voc\_anno\_gray\_dir, ann))

        print('converting done')

    def get\_mindrecord\_dataset(self, is\_training,num\_shards=1, shuffle=True):

        datas = []

        if is\_training:

            data\_lst = self.voc\_train\_lst

            self.mindrecord\_save = os.path.join(self.mindrecord\_save,'train')

        else:

            data\_lst = self.voc\_val\_lst

            self.mindrecord\_save = os.path.join(self.mindrecord\_save,'eval')

        if os.path.exists(self.mindrecord\_save):

            #shutil.rmtree(self.mindrecord\_save)

            print('mindrecord file is already exists！')

            self.mindrecord\_save = os.path.join(self.mindrecord\_save,'VOC\_mindrecord')

            return

        with open(data\_lst) as f:

            lines = f.readlines()

        if shuffle:

            np.random.shuffle(lines)

        print('creating mindrecord dataset...')

        os.makedirs(self.mindrecord\_save)

        self.mindrecord\_save = os.path.join(self.mindrecord\_save,'VOC\_mindrecord')

        print('number of samples:', len(lines))

        seg\_schema = {"file\_name": {"type": "string"}, "label": {"type": "bytes"}, "data": {"type": "bytes"}}

        writer = FileWriter(file\_name=self.mindrecord\_save, shard\_num=num\_shards)

        writer.add\_schema(seg\_schema, "seg\_schema")

        cnt = 0

        for l in lines:

            id\_ = l.strip()

            img\_path = os.path.join(self.voc\_img\_dir, id\_ + '.jpg')

            label\_path = os.path.join(self.voc\_anno\_gray\_dir, id\_ + '.png')

            sample\_ = {"file\_name": img\_path.split('/')[-1]}

            with open(img\_path, 'rb') as f:

                sample\_['data'] = f.read()

            with open(label\_path, 'rb') as f:

                sample\_['label'] = f.read()

            datas.append(sample\_)

            cnt += 1

            if cnt % 1000 == 0:

                writer.write\_raw\_data(datas)

                print('number of samples written:', cnt)

                datas = []

        if datas:

            writer.write\_raw\_data(datas)

        writer.commit()

        print('number of samples written:', cnt)

        print('Create Mindrecord Done')

    def get\_dataset(self, repeat=1):

        data\_set = de.MindDataset(dataset\_file=self.mindrecord\_save, columns\_list=["data", "label"],

                                  shuffle=True, num\_parallel\_workers=self.num\_readers,

                                  num\_shards=self.shard\_num, shard\_id=self.shard\_id)

        transforms\_list = self.preprocess\_

        data\_set = data\_set.map(operations=transforms\_list, input\_columns=["data", "label"],

                                output\_columns=["data", "label"],

                                num\_parallel\_workers=self.num\_parallel\_calls)

        data\_set = data\_set.shuffle(buffer\_size=self.batch\_size \* 10)

        data\_set = data\_set.batch(self.batch\_size, drop\_remainder=True)

        data\_set = data\_set.repeat(repeat)

        return data\_set

步骤 3构建网络

import mindspore.nn as nn

from mindspore.ops import operations as P

def conv1x1(in\_planes, out\_planes, stride=1):

    return nn.Conv2d(in\_planes, out\_planes, kernel\_size=1, stride=stride, weight\_init='xavier\_uniform')

def conv3x3(in\_planes, out\_planes, stride=1, dilation=1, padding=1):

    return nn.Conv2d(in\_planes, out\_planes, kernel\_size=3, stride=stride, pad\_mode='pad', padding=padding,

                     dilation=dilation, weight\_init='xavier\_uniform')

class Resnet(nn.Cell):

    def \_\_init\_\_(self, block, block\_num, output\_stride, use\_batch\_statistics=True):

        super(Resnet, self).\_\_init\_\_()

        self.inplanes = 64

        self.conv1 = nn.Conv2d(3, self.inplanes, kernel\_size=7, stride=2, pad\_mode='pad', padding=3,

                               weight\_init='xavier\_uniform')

        self.bn1 = nn.BatchNorm2d(self.inplanes, use\_batch\_statistics=use\_batch\_statistics)

        self.relu = nn.ReLU()

        self.maxpool = nn.MaxPool2d(kernel\_size=3, stride=2, pad\_mode='same')

        self.layer1 = self.\_make\_layer(block, 64, block\_num[0], use\_batch\_statistics=use\_batch\_statistics)

        self.layer2 = self.\_make\_layer(block, 128, block\_num[1], stride=2, use\_batch\_statistics=use\_batch\_statistics)

        if output\_stride == 16:

            self.layer3 = self.\_make\_layer(block, 256, block\_num[2], stride=2,

                                           use\_batch\_statistics=use\_batch\_statistics)

            self.layer4 = self.\_make\_layer(block, 512, block\_num[3], stride=1, base\_dilation=2, grids=[1, 2, 4],

                                           use\_batch\_statistics=use\_batch\_statistics)

        elif output\_stride == 8:

            self.layer3 = self.\_make\_layer(block, 256, block\_num[2], stride=1, base\_dilation=2,

                                           use\_batch\_statistics=use\_batch\_statistics)

            self.layer4 = self.\_make\_layer(block, 512, block\_num[3], stride=1, base\_dilation=4, grids=[1, 2, 4],

                                           use\_batch\_statistics=use\_batch\_statistics)

    def \_make\_layer(self, block, planes, blocks, stride=1, base\_dilation=1, grids=None, use\_batch\_statistics=True):

        if stride != 1 or self.inplanes != planes \* block.expansion:

            downsample = nn.SequentialCell([

                conv1x1(self.inplanes, planes \* block.expansion, stride),

                nn.BatchNorm2d(planes \* block.expansion, use\_batch\_statistics=use\_batch\_statistics)

            ])

        if grids is None:

            grids = [1] \* blocks

        layers = [

            block(self.inplanes, planes, stride, downsample, dilation=base\_dilation \* grids[0],

                  use\_batch\_statistics=use\_batch\_statistics)

        ]

        self.inplanes = planes \* block.expansion

        for i in range(1, blocks):

            layers.append(

                block(self.inplanes, planes, dilation=base\_dilation \* grids[i],

                      use\_batch\_statistics=use\_batch\_statistics))

        return nn.SequentialCell(layers)

    def construct(self, x):

        out = self.conv1(x)

        out = self.bn1(out)

        out = self.relu(out)

        out = self.maxpool(out)

        out = self.layer1(out)

        out = self.layer2(out)

        out = self.layer3(out)

        out = self.layer4(out)

        return out

class Bottleneck(nn.Cell):

    expansion = 4

    def \_\_init\_\_(self, inplanes, planes, stride=1, downsample=None, dilation=1, use\_batch\_statistics=True):

        super(Bottleneck, self).\_\_init\_\_()

        self.conv1 = conv1x1(inplanes, planes)

        self.bn1 = nn.BatchNorm2d(planes, use\_batch\_statistics=use\_batch\_statistics)

        self.conv2 = conv3x3(planes, planes, stride, dilation, dilation)

        self.bn2 = nn.BatchNorm2d(planes, use\_batch\_statistics=use\_batch\_statistics)

        self.conv3 = conv1x1(planes, planes \* self.expansion)

        self.bn3 = nn.BatchNorm2d(planes \* self.expansion, use\_batch\_statistics=use\_batch\_statistics)

        self.relu = nn.ReLU()

        self.downsample = downsample

        self.add = P.TensorAdd()

    def construct(self, x):

        identity = x

        out = self.conv1(x)

        out = self.bn1(out)

        out = self.relu(out)

        out = self.conv2(out)

        out = self.bn2(out)

        out = self.relu(out)

        out = self.conv3(out)

        out = self.bn3(out)

        if self.downsample is not None:

            identity = self.downsample(x)

        out = self.add(out, identity)

        out = self.relu(out)

        return out

class ASPP(nn.Cell):

    def \_\_init\_\_(self, atrous\_rates, phase='train', in\_channels=2048, num\_classes=21,

                 use\_batch\_statistics=True):

        super(ASPP, self).\_\_init\_\_()

        self.phase = phase

        out\_channels = 256

        self.aspp1 = ASPPConv(in\_channels, out\_channels, atrous\_rates[0], use\_batch\_statistics=use\_batch\_statistics)

        self.aspp2 = ASPPConv(in\_channels, out\_channels, atrous\_rates[1], use\_batch\_statistics=use\_batch\_statistics)

        self.aspp3 = ASPPConv(in\_channels, out\_channels, atrous\_rates[2], use\_batch\_statistics=use\_batch\_statistics)

        self.aspp4 = ASPPConv(in\_channels, out\_channels, atrous\_rates[3], use\_batch\_statistics=use\_batch\_statistics)

        self.aspp\_pooling = ASPPPooling(in\_channels, out\_channels)

        self.conv1 = nn.Conv2d(out\_channels \* (len(atrous\_rates) + 1), out\_channels, kernel\_size=1,

                               weight\_init='xavier\_uniform')

        self.bn1 = nn.BatchNorm2d(out\_channels, use\_batch\_statistics=use\_batch\_statistics)

        self.relu = nn.ReLU()

        self.conv2 = nn.Conv2d(out\_channels, num\_classes, kernel\_size=1, weight\_init='xavier\_uniform', has\_bias=True)

        self.concat = P.Concat(axis=1)

        self.drop = nn.Dropout(0.3)

    def construct(self, x):

        x1 = self.aspp1(x)

        x2 = self.aspp2(x)

        x3 = self.aspp3(x)

        x4 = self.aspp4(x)

        x5 = self.aspp\_pooling(x)

        x = self.concat((x1, x2))

        x = self.concat((x, x3))

        x = self.concat((x, x4))

        x = self.concat((x, x5))

        x = self.conv1(x)

        x = self.bn1(x)

        x = self.relu(x)

        if self.phase == 'train':

            x = self.drop(x)

        x = self.conv2(x)

        return x

class ASPPPooling(nn.Cell):

    def \_\_init\_\_(self, in\_channels, out\_channels, use\_batch\_statistics=True):

        super(ASPPPooling, self).\_\_init\_\_()

        self.conv = nn.SequentialCell([

            nn.Conv2d(in\_channels, out\_channels, kernel\_size=1, weight\_init='xavier\_uniform'),

            nn.BatchNorm2d(out\_channels, use\_batch\_statistics=use\_batch\_statistics),

            nn.ReLU()

        ])

        self.shape = P.Shape()

    def construct(self, x):

        size = self.shape(x)

        out = nn.AvgPool2d(size[2])(x)

        out = self.conv(out)

        out = P.ResizeNearestNeighbor((size[2], size[3]), True)(out)

        return out

class ASPPConv(nn.Cell):

    def \_\_init\_\_(self, in\_channels, out\_channels, atrous\_rate=1, use\_batch\_statistics=True):

        super(ASPPConv, self).\_\_init\_\_()

        if atrous\_rate == 1:

            conv = nn.Conv2d(in\_channels, out\_channels, kernel\_size=1, has\_bias=False, weight\_init='xavier\_uniform')

        else:

            conv = nn.Conv2d(in\_channels, out\_channels, kernel\_size=3, pad\_mode='pad', padding=atrous\_rate,

                             dilation=atrous\_rate, weight\_init='xavier\_uniform')

        bn = nn.BatchNorm2d(out\_channels, use\_batch\_statistics=use\_batch\_statistics)

        relu = nn.ReLU()

        self.aspp\_conv = nn.SequentialCell([conv, bn, relu])

    def construct(self, x):

        out = self.aspp\_conv(x)

        return out

class DeepLabV3(nn.Cell):

    def \_\_init\_\_(self, phase='train', num\_classes=21, output\_stride=16, freeze\_bn=False):

        super(DeepLabV3, self).\_\_init\_\_()

        use\_batch\_statistics = not freeze\_bn

        self.resnet = Resnet(Bottleneck, [3, 4, 23, 3], output\_stride=output\_stride,

                             use\_batch\_statistics=use\_batch\_statistics)

        self.aspp = ASPP([1, 6, 12, 18], phase, 2048, num\_classes,

                         use\_batch\_statistics=use\_batch\_statistics)

        self.shape = P.Shape()

    def construct(self, x):

        size = self.shape(x)

        out = self.resnet(x)

        out = self.aspp(out)

        out = P.ResizeBilinear((size[2], size[3]), True)(out)

        return out

步骤 4定义不同的学习率

def cosine\_lr(base\_lr, decay\_steps, total\_steps):

    for i in range(total\_steps):

        step\_ = min(i, decay\_steps)

        yield base\_lr \* 0.5 \* (1 + np.cos(np.pi \* step\_ / decay\_steps))

def poly\_lr(base\_lr, decay\_steps, total\_steps, end\_lr=0.0001, power=0.9):

    for i in range(total\_steps):

        step\_ = min(i, decay\_steps)

        yield (base\_lr - end\_lr) \* ((1.0 - step\_ / decay\_steps) \*\* power) + end\_lr

def exponential\_lr(base\_lr, decay\_steps, decay\_rate, total\_steps, staircase=False):

    for i in range(total\_steps):

        if staircase:

            power\_ = i // decay\_steps

        else:

            power\_ = float(i) / decay\_steps

        yield base\_lr \* (decay\_rate \*\* power\_)

步骤 5定义损失函数

from mindspore import Tensor

import mindspore.common.dtype as mstype

import mindspore.nn as nn

from mindspore.ops import operations as P

class SoftmaxCrossEntropyLoss(nn.Cell):

    def \_\_init\_\_(self, num\_cls=21, ignore\_label=255):

        super(SoftmaxCrossEntropyLoss, self).\_\_init\_\_()

        self.one\_hot = P.OneHot(axis=-1)

        self.on\_value = Tensor(1.0, mstype.float32)

        self.off\_value = Tensor(0.0, mstype.float32)

        self.cast = P.Cast()

        self.ce = nn.SoftmaxCrossEntropyWithLogits()

        self.not\_equal = P.NotEqual()

        self.num\_cls = num\_cls

        self.ignore\_label = ignore\_label

        self.mul = P.Mul()

        self.sum = P.ReduceSum(False)

        self.div = P.RealDiv()

        self.transpose = P.Transpose()

        self.reshape = P.Reshape()

    def construct(self, logits, labels):

        labels\_int = self.cast(labels, mstype.int32)

        labels\_int = self.reshape(labels\_int, (-1,))

        logits\_ = self.transpose(logits, (0, 2, 3, 1))

        logits\_ = self.reshape(logits\_, (-1, self.num\_cls))

        weights = self.not\_equal(labels\_int, self.ignore\_label)

        weights = self.cast(weights, mstype.float32)

        one\_hot\_labels = self.one\_hot(labels\_int, self.num\_cls, self.on\_value, self.off\_value)

        loss = self.ce(logits\_, one\_hot\_labels)

        loss = self.mul(weights, loss)

        loss = self.div(self.sum(loss), self.sum(weights))

        return loss

步骤 6构建训练网络的函数

"""train deeplabv3."""

import os

import sys

sys.path.insert(0,'./deeplabv3/deeplabv3\_2/')     # your code path

from easydict import EasyDict as edict

import shutil

# import moxing as mox

from mindspore import context

from mindspore.train.model import ParallelMode, Model

import mindspore.nn as nn

from mindspore.train.callback import ModelCheckpoint, CheckpointConfig

from mindspore.train.serialization import load\_checkpoint, load\_param\_into\_net

from mindspore.communication.management import init, get\_rank, get\_group\_size

from mindspore.train.callback import LossMonitor, TimeMonitor

from mindspore.train.loss\_scale\_manager import FixedLossScaleManager

from mindspore.common import set\_seed

set\_seed(1)

context.set\_context(mode=context.GRAPH\_MODE, enable\_auto\_mixed\_precision=True, save\_graphs=False,

                    device\_target="Ascend")

class BuildTrainNetwork(nn.Cell):

    def \_\_init\_\_(self, network, criterion):

        super(BuildTrainNetwork, self).\_\_init\_\_()

        self.network = network

        self.criterion = criterion

    def construct(self, input\_data, label):

        output = self.network(input\_data)

        net\_loss = self.criterion(output, label)

        return net\_loss

def train(args):

    # init multicards training

    if args.is\_distributed:

        init()

        args.rank = get\_rank()

        args.group\_size = get\_group\_size()

        parallel\_mode = ParallelMode.DATA\_PARALLEL

        context.set\_auto\_parallel\_context(parallel\_mode=parallel\_mode, gradients\_mean=True, device\_num=args.group\_size)

    # dataset

    dataset = SegDataset(image\_mean=args.image\_mean,

                                        image\_std=args.image\_std,

                                        data\_file=args.data\_file,

                                        batch\_size=args.batch\_size,

                                        crop\_size=args.crop\_size,

                                        max\_scale=args.max\_scale,

                                        min\_scale=args.min\_scale,

                                        ignore\_label=args.ignore\_label,

                                        num\_classes=args.num\_classes,

                                        num\_readers=2,

                                        num\_parallel\_calls=4,

                                        shard\_id=args.rank,

                                        shard\_num=args.group\_size)

    dataset.get\_gray\_dataset()

    dataset.get\_mindrecord\_dataset(is\_training=True)

    dataset = dataset.get\_dataset(repeat=1)

    # network

    if args.model == 'deeplab\_v3\_s16':

        network = DeepLabV3('train', args.num\_classes, 16, args.freeze\_bn)

    elif args.model == 'deeplab\_v3\_s8':

        network = DeepLabV3('train', args.num\_classes, 8, args.freeze\_bn)

    else:

        raise NotImplementedError('model [{:s}] not recognized'.format(args.model))

    # loss

    loss\_ = SoftmaxCrossEntropyLoss(args.num\_classes, args.ignore\_label)

    loss\_.add\_flags\_recursive(fp32=True)

    train\_net = BuildTrainNetwork(network, loss\_)

    # load pretrained model

    param\_dict = load\_checkpoint(args.ckpt\_file)

    load\_param\_into\_net(train\_net, param\_dict)

    # optimizer

    iters\_per\_epoch = dataset.get\_dataset\_size()

    total\_train\_steps = iters\_per\_epoch \* args.train\_epochs

    if args.lr\_type == 'cos':

        lr\_iter = cosine\_lr(args.base\_lr, total\_train\_steps, total\_train\_steps)

    elif args.lr\_type == 'poly':

        lr\_iter = poly\_lr(args.base\_lr, total\_train\_steps, total\_train\_steps, end\_lr=0.0, power=0.9)

    elif args.lr\_type == 'exp':

        lr\_iter = exponential\_lr(args.base\_lr, args.lr\_decay\_step, args.lr\_decay\_rate,

                                                total\_train\_steps, staircase=True)

    else:

        raise ValueError('unknown learning rate type')

    opt = nn.Momentum(params=train\_net.trainable\_params(), learning\_rate=lr\_iter, momentum=0.9, weight\_decay=0.0001,

                      loss\_scale=args.loss\_scale)

    # loss scale

    manager\_loss\_scale = FixedLossScaleManager(args.loss\_scale, drop\_overflow\_update=False)

    model = Model(train\_net, optimizer=opt, amp\_level="O3", loss\_scale\_manager=manager\_loss\_scale)

    # callback for saving ckpts

    time\_cb = TimeMonitor(data\_size=iters\_per\_epoch)

    loss\_cb = LossMonitor()

    cbs = [time\_cb, loss\_cb]

    if args.rank == 0:

        config\_ck = CheckpointConfig(save\_checkpoint\_steps=iters\_per\_epoch,

                                     keep\_checkpoint\_max=args.keep\_checkpoint\_max)

        ckpoint\_cb = ModelCheckpoint(prefix=args.model, directory=args.train\_dir, config=config\_ck)

        cbs.append(ckpoint\_cb)

    model.train(args.train\_epochs, dataset, callbacks=cbs,dataset\_sink\_mode=True)

步骤 7设定相关参数并训练网络

cfg = edict({

    "batch\_size": 16,

    "crop\_size": 513,

    "image\_mean": [103.53, 116.28, 123.675],

    "image\_std": [57.375, 57.120, 58.395],

    "min\_scale": 0.5,

    "max\_scale": 2.0,

    "ignore\_label": 255,

    "num\_classes": 21,

    "train\_epochs" : 3,

    "lr\_type": 'cos',

    "base\_lr": 0.0,

    "lr\_decay\_step": 3\*91,

    "lr\_decay\_rate" :0.1,

    "loss\_scale": 2048,

    "model": 'deeplab\_v3\_s8',

    'rank': 0,

    'group\_size':1,

    'keep\_checkpoint\_max':1,

    'train\_dir': 'model',

    'is\_distributed':False,

    'freeze\_bn':True

})

if os.path.exists(cfg.train\_dir):

    shutil.rmtree(cfg.train\_dir)

data\_path = './VOC2012'

# if not os.path.exists(data\_path):

#     mox.file.copy\_parallel(src\_url="s3://share-course/dataset/voc2012\_raw/", dst\_url=data\_path)

cfg.data\_file = data\_path

ckpt\_path = './ckpt/deeplab\_v3\_s8-300\_11.ckpt'

# if not os.path.exists(ckpt\_path):

#     mox.file.copy\_parallel(src\_url="s3://share-course/checkpoint/deeplabv3/deeplab\_v3\_s8-800\_82.ckpt", dst\_url=ckpt\_path)

cfg.ckpt\_file = ckpt\_path

train(cfg)

# if the model is needed in next time, you can save the mode file to yours obs.

# mox.file.copy\_parallel(src\_url=cfg.train\_dir, dst\_url=os.path.join("s3://yyq-2/DATA/code/deeplabv3/",cfg.train\_dir))

步骤 8验证网络

"""eval deeplabv3."""

import os

import sys

sys.path.insert(0,'./deeplabv3/deeplabv3\_2/')     # your code path

from easydict import EasyDict as edict

from PIL import Image

import PIL

import matplotlib.pyplot as plt

import matplotlib as mpl

import matplotlib.colors as colors

import numpy as np

import cv2

# import moxing as mox

from mindspore import Tensor

import mindspore.common.dtype as mstype

import mindspore.nn as nn

from mindspore import context

from mindspore.train.serialization import load\_checkpoint, load\_param\_into\_net

context.set\_context(mode=context.GRAPH\_MODE, device\_target="Ascend", save\_graphs=False)

def cal\_hist(a, b, n):

    k = (a >= 0) & (a < n)

    return np.bincount(n \* a[k].astype(np.int32) + b[k], minlength=n \*\* 2).reshape(n, n)

def resize\_long(img, long\_size=513):

    h, w, \_ = img.shape

    if h > w:

        new\_h = long\_size

        new\_w = int(1.0 \* long\_size \* w / h)

    else:

        new\_w = long\_size

        new\_h = int(1.0 \* long\_size \* h / w)

    imo = cv2.resize(img, (new\_w, new\_h))

    return imo

class BuildEvalNetwork(nn.Cell):

    def \_\_init\_\_(self, network):

        super(BuildEvalNetwork, self).\_\_init\_\_()

        self.network = network

        self.softmax = nn.Softmax(axis=1)

    def construct(self, input\_data):

        output = self.network(input\_data)

        output = self.softmax(output)

        return output

def pre\_process(args, img\_, crop\_size=513):

    # resize

    img\_ = resize\_long(img\_, crop\_size)

    resize\_h, resize\_w, \_ = img\_.shape

    # mean, std

    image\_mean = np.array(args.image\_mean)

    image\_std = np.array(args.image\_std)

    img\_ = (img\_ - image\_mean) / image\_std

    # pad to crop\_size

    pad\_h = crop\_size - img\_.shape[0]

    pad\_w = crop\_size - img\_.shape[1]

    if pad\_h > 0 or pad\_w > 0:

        img\_ = cv2.copyMakeBorder(img\_, 0, pad\_h, 0, pad\_w, cv2.BORDER\_CONSTANT, value=0)

    # hwc to chw

    img\_ = img\_.transpose((2, 0, 1))

    return img\_, resize\_h, resize\_w

def eval\_batch(args, eval\_net, img\_lst, crop\_size=513, flip=True):

    result\_lst = []

    batch\_size = len(img\_lst)

    batch\_img = np.zeros((args.batch\_size, 3, crop\_size, crop\_size), dtype=np.float32)

    resize\_hw = []

    for l in range(batch\_size):

        img\_ = img\_lst[l]

        img\_, resize\_h, resize\_w = pre\_process(args, img\_, crop\_size)

        batch\_img[l] = img\_

        resize\_hw.append([resize\_h, resize\_w])

    batch\_img = np.ascontiguousarray(batch\_img)

    net\_out = eval\_net(Tensor(batch\_img, mstype.float32))

    net\_out = net\_out.asnumpy()

    if flip:

        batch\_img = batch\_img[:, :, :, ::-1]

        net\_out\_flip = eval\_net(Tensor(batch\_img, mstype.float32))

        net\_out += net\_out\_flip.asnumpy()[:, :, :, ::-1]

    for bs in range(batch\_size):

        probs\_ = net\_out[bs][:, :resize\_hw[bs][0], :resize\_hw[bs][1]].transpose((1, 2, 0))

        ori\_h, ori\_w = img\_lst[bs].shape[0], img\_lst[bs].shape[1]

        probs\_ = cv2.resize(probs\_, (ori\_w, ori\_h))

        result\_lst.append(probs\_)

    return result\_lst

def eval\_batch\_scales(args, eval\_net, img\_lst, scales,

                      base\_crop\_size=513, flip=True):

    sizes\_ = [int((base\_crop\_size - 1) \* sc) + 1 for sc in scales]

    probs\_lst = eval\_batch(args, eval\_net, img\_lst, crop\_size=sizes\_[0], flip=flip)

    #print(sizes\_)

    for crop\_size\_ in sizes\_[1:]:

        probs\_lst\_tmp = eval\_batch(args, eval\_net, img\_lst, crop\_size=crop\_size\_, flip=flip)

        for pl, \_ in enumerate(probs\_lst):

            probs\_lst[pl] += probs\_lst\_tmp[pl]

    result\_msk = []

    for i in probs\_lst:

        result\_msk.append(i.argmax(axis=2))

    return result\_msk

# The color source: print(list(colors.cnames.keys()))

#print(list(colors.cnames.keys()))

num\_class = {0: 'background', 1: 'aeroplane', 2: 'bicycle', 3: 'bird', 4: 'boat', 5: 'bottle', 6: 'bus', 7: 'car', 8: 'cat',

             9: 'chair', 10: 'cow', 11: 'diningtable', 12: 'dog', 13: 'horse', 14: 'motorbike', 15: 'person', 16: 'pottedplant',

             17: 'sheep', 18: 'sofa', 19: 'train', 20: 'tvmonitor', 21: 'edge'}

num\_color = {0:'aliceblue', 1:'grey', 2:'red', 3:'green', 4:'darkorange', 5:'lime', 6:'bisque',

     7:'black', 8:'blanchedalmond', 9:'blue', 10:'blueviolet', 11:'brown', 12:'burlywood', 13:'cadetblue',

     14:'darkorange', 15:'tan', 16:'darkviolet', 17:'cornflowerblue', 18:'yellow', 19:'crimson', 20:'darkcyan'}

color\_dic = [num\_color[k] for k in sorted(num\_color.keys())]

bounds = list(range(21))

cmap = mpl.colors.ListedColormap(color\_dic)

norm = mpl.colors.BoundaryNorm(bounds, cmap.N)

def num\_to\_ClassAndColor(num\_list):

    color\_ = []

    class\_ = []

    for num in num\_list:

        color\_.append(num\_class[num])

        class\_.append(num\_color[num])

    return color\_,class\_

def net\_eval(args):

    # network

    if args.model == 'deeplab\_v3\_s16':

        network = DeepLabV3('eval', args.num\_classes, 16, args.freeze\_bn)

    elif args.model == 'deeplab\_v3\_s8':

        network = DeepLabV3('eval', args.num\_classes, 8, args.freeze\_bn)

    else:

        raise NotImplementedError('model [{:s}] not recognized'.format(args.model))

    eval\_net = BuildEvalNetwork(network)

    # load model

    param\_dict = load\_checkpoint(args.ckpt\_file)

    load\_param\_into\_net(eval\_net, param\_dict)

    eval\_net.set\_train(False)

    # data list

    with open(args.data\_lst) as f:

        img\_lst = f.readlines()

    # evaluate

    hist = np.zeros((args.num\_classes, args.num\_classes))

    batch\_img\_lst = []

    batch\_msk\_lst = []

    bi = 0

    image\_num = 0

    for i, line in enumerate(img\_lst):

        id\_ = line.strip()

        img\_path = os.path.join(cfg.voc\_img\_dir, id\_ + '.jpg')

        msk\_path = os.path.join(cfg.voc\_anno\_gray\_dir, id\_ + '.png')

        img\_ = cv2.imread(img\_path)

        msk\_ = cv2.imread(msk\_path, cv2.IMREAD\_GRAYSCALE)

        batch\_img\_lst.append(img\_)

        batch\_msk\_lst.append(msk\_)

        if args.if\_png:

            batch\_res = eval\_batch\_scales(args, eval\_net, batch\_img\_lst, scales=args.scales,

                                          base\_crop\_size=args.crop\_size, flip=args.flip)

            height ,weight = batch\_res[0].shape

            batch\_msk\_lst[0][batch\_msk\_lst[0]==args.ignore\_label] = 0

            plt.figure(figsize=(3 \* weight/1024\*10, 2 \* height/1024\*10))

            plt.subplot(1,3,1)

            image = Image.open(img\_path)

            plt.imshow(image)

            plt.subplot(1,3,2)

            plt.imshow(image)

            plt.imshow(batch\_res[0],alpha=0.8,interpolation='none', cmap=cmap, norm=norm)

            plt.subplot(1,3,3)

            plt.imshow(image)

            plt.imshow(batch\_msk\_lst[0],alpha=0.8,interpolation='none', cmap=cmap, norm=norm)

            plt.show()

            prediction\_num = np.unique(batch\_res[0])

            real\_num = np.unique(batch\_msk\_lst[0])

            prediction\_color,prediction\_class = num\_to\_ClassAndColor(prediction\_num)

            print('prediction num:',prediction\_num)

            print('prediction color:',prediction\_color)

            print('prediction class:',prediction\_class)

            real\_color,real\_class = num\_to\_ClassAndColor(real\_num)

            print('groundtruth num:',real\_num)

            print('groundtruth color:',real\_color)

            print('groundtruth class:',real\_class)

            batch\_img\_lst = []

            batch\_msk\_lst = []

            if i < args.num\_png-1:

                continue

            else:

                return

        bi += 1

        if bi == args.batch\_size:

            batch\_res = eval\_batch\_scales(args, eval\_net, batch\_img\_lst, scales=args.scales,

                                          base\_crop\_size=args.crop\_size, flip=args.flip)

            for mi in range(args.batch\_size):

                hist += cal\_hist(batch\_msk\_lst[mi].flatten(), batch\_res[mi].flatten(), args.num\_classes)

            bi = 0

            batch\_img\_lst = []

            batch\_msk\_lst = []

            if (i+1)%100 == 0:

                print('processed {} images'.format(i+1))

        image\_num = i

    if bi > 0:

        batch\_res = eval\_batch\_scales(args, eval\_net, batch\_img\_lst, scales=args.scales,

                                      base\_crop\_size=args.crop\_size, flip=args.flip)

        for mi in range(bi):

            hist += cal\_hist(batch\_msk\_lst[mi].flatten(), batch\_res[mi].flatten(), args.num\_classes)

        if (i+1) % 100 == 0:

            print('processed {} images'.format(image\_num + 1))

    iu = np.diag(hist) / (hist.sum(1) + hist.sum(0) - np.diag(hist))

    print('mean IoU', np.nanmean(iu))

# test  1

cfg = edict({

    "batch\_size": 1,

    "crop\_size": 513,

    "image\_mean": [103.53, 116.28, 123.675],

    "image\_std": [57.375, 57.120, 58.395],

    "scales": [1.0],           # [0.5,0.75,1.0,1.25,1.75]

    'flip': True,

    'ignore\_label': 255,

    'num\_classes':21,

    'model': 'deeplab\_v3\_s8',

    'freeze\_bn': True,

    'if\_png':False,

    'num\_png':10

})

data\_path = './VOC2012'

# if not os.path.exists(data\_path):

#     mox.file.copy\_parallel(src\_url="s3://share-course/dataset/voc2012\_raw/", dst\_url=data\_path)

cfg.data\_file = data\_path

# dataset

dataset = SegDataset(image\_mean=cfg.image\_mean,

                                    image\_std=cfg.image\_std,

                                    data\_file=cfg.data\_file)

dataset.get\_gray\_dataset()

cfg.data\_lst = os.path.join(cfg.data\_file,'ImageSets/Segmentation/val.txt')

cfg.voc\_img\_dir = os.path.join(cfg.data\_file,'JPEGImages')

cfg.voc\_anno\_gray\_dir = os.path.join(cfg.data\_file,'SegmentationClassGray')

ckpt\_path = './model'

# if not os.path.exists(ckpt\_path):

#     mox.file.copy\_parallel(src\_url="s3://yyq-3/DATA/code/deeplabv3/model", dst\_url=ckpt\_path)   #if yours model had saved

cfg.ckpt\_file = os.path.join(ckpt\_path,'deeplab\_v3\_s8-3\_91.ckpt')

print('loading checkpoing:',cfg.ckpt\_file)

net\_eval(cfg)

# test 2

cfg = edict({

    "batch\_size": 1,

    "crop\_size": 513,

    "image\_mean": [103.53, 116.28, 123.675],

    "image\_std": [57.375, 57.120, 58.395],

    "scales": [1.0],           # [0.5,0.75,1.0,1.25,1.75]

    'flip': True,

    'ignore\_label': 255,

    'num\_classes':21,

    'model': 'deeplab\_v3\_s8',

    'freeze\_bn': True,

    'if\_png':True,

    'num\_png':3

})

# import moxing as mox

data\_path = './VOC2012'

# if not os.path.exists(data\_path):

#     mox.file.copy\_parallel(src\_url="s3://share-course/dataset/voc2012\_raw/", dst\_url=data\_path)

cfg.data\_file = data\_path

# dataset

dataset = SegDataset(image\_mean=cfg.image\_mean,

                                    image\_std=cfg.image\_std,

                                    data\_file=cfg.data\_file)

dataset.get\_gray\_dataset()

cfg.data\_lst = os.path.join(cfg.data\_file,'ImageSets/Segmentation/val.txt')

cfg.voc\_img\_dir = os.path.join(cfg.data\_file,'JPEGImages')

cfg.voc\_anno\_gray\_dir = os.path.join(cfg.data\_file,'SegmentationClassGray')

ckpt\_path = './model'

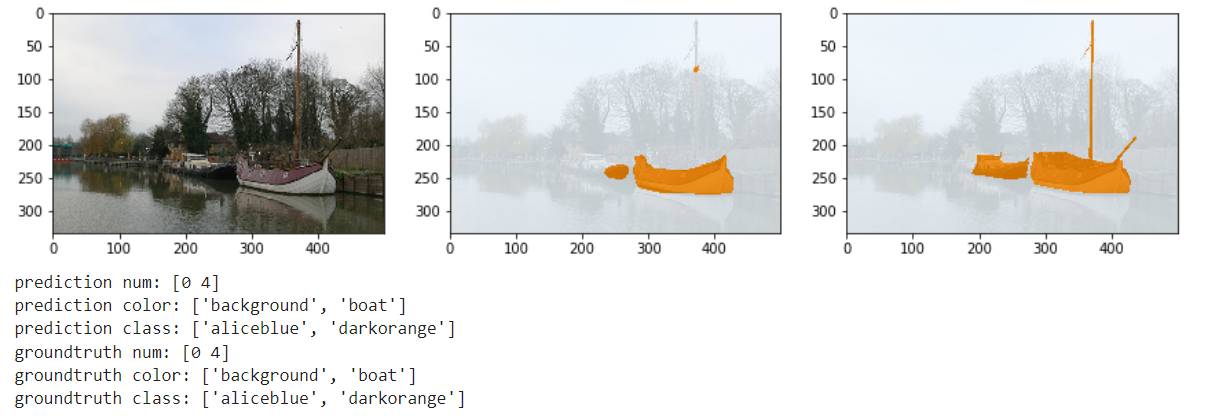
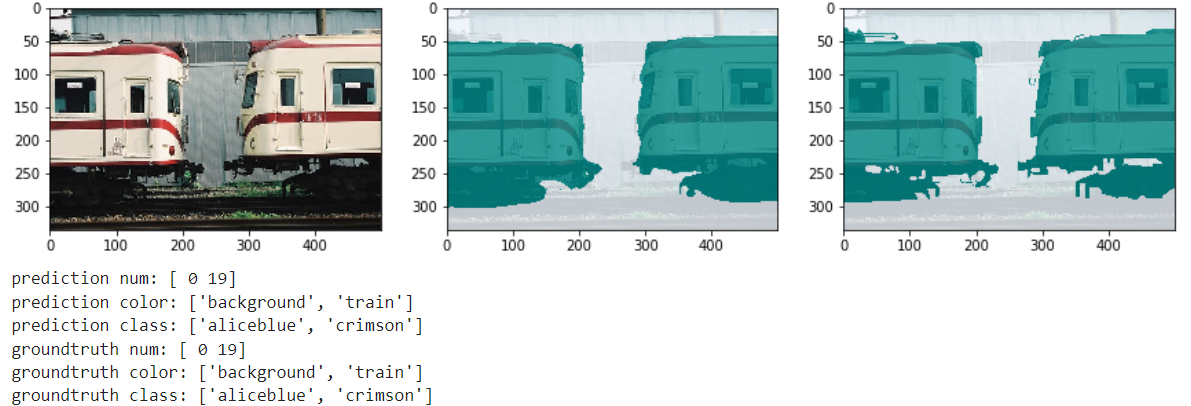
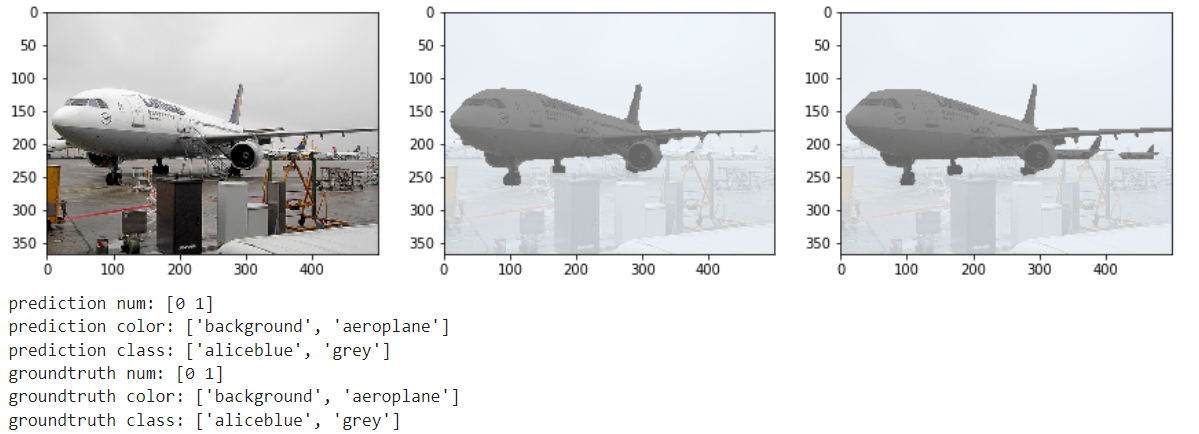
# if not os.path.exists(ckpt\_path):

#     mox.file.copy\_parallel(src\_url="s3://yyq-3/DATA/code/deeplabv3/model", dst\_url=ckpt\_path)     #if yours model had saved

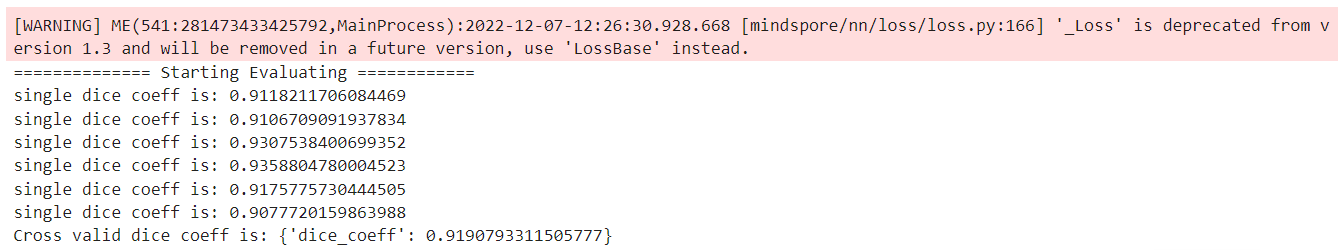
cfg.ckpt\_file = os.path.join(ckpt\_path,'deeplab\_v3\_s8-3\_91.ckpt')

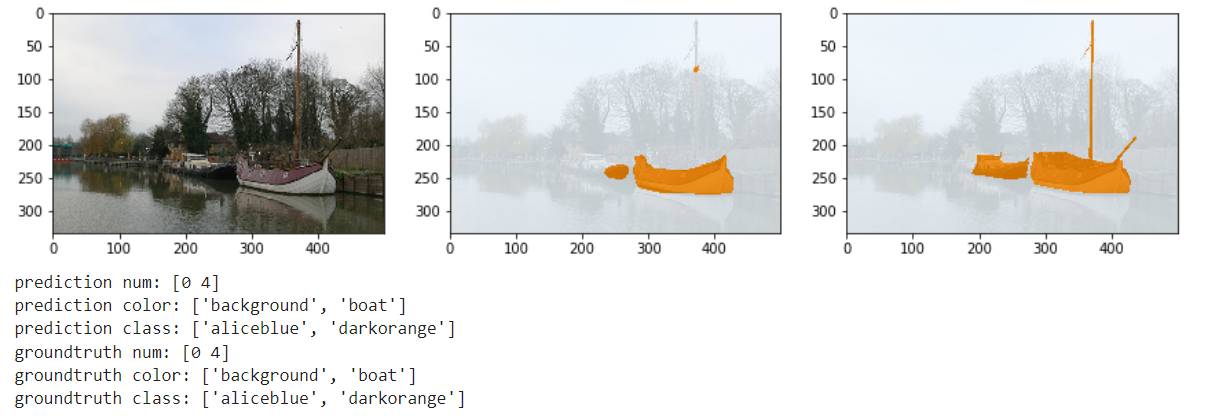
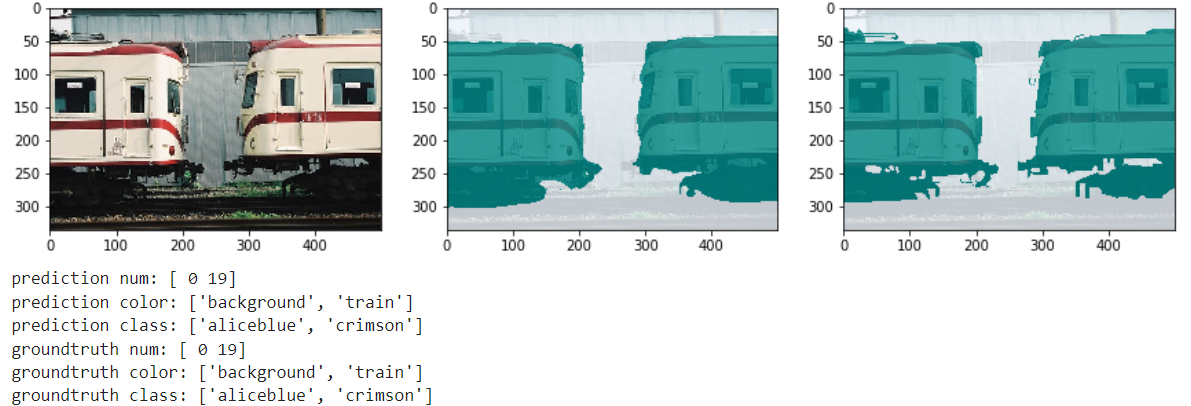
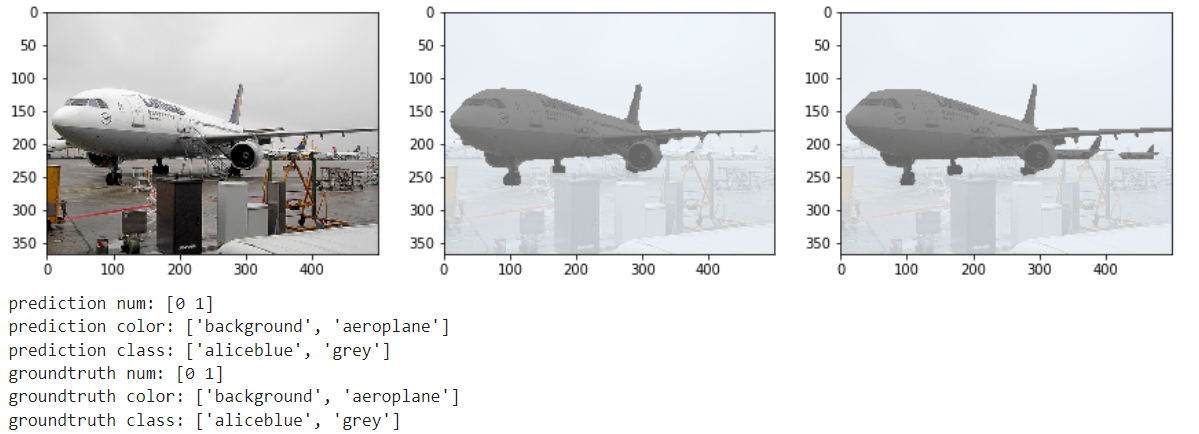
print('loading checkpoing:',cfg.ckpt\_file)

net\_eval(cfg)



1. 实验结果





1. 实验总结

本实验主要介绍如何使用MindSpore在voc2012数据集上训练和推理deeplabv3网络模型，从而实现图像语义分割任务。通过本实验了解如何处理图像分割数据标签，定义和训练卷积神经网络等的基本操作。