UNetWithEfficientNet

May 6, 2020

[1]:
```
#Mount Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[2]:
```
#Restart runtime after running once

!pip install segmentation-models-pytorch==0.1.0
!pip install -U catalyst
```

```
Requirement already satisfied: segmentation-models-pytorch==0.1.0 in
/usr/local/lib/python3.6/dist-packages (0.1.0)
Requirement already satisfied: pretrainedmodels==0.7.4 in
/usr/local/lib/python3.6/dist-packages (from segmentation-models-pytorch==0.1.0)
(0.7.4)
Requirement already satisfied: torchvision>=0.3.0 in
/usr/local/lib/python3.6/dist-packages (from segmentation-models-pytorch==0.1.0)
(0.6.0+cu101)
Requirement already satisfied: efficientnet-pytorch>=0.5.1 in
/usr/local/lib/python3.6/dist-packages (from segmentation-models-pytorch==0.1.0)
(0.6.3)
Requirement already satisfied: torch in /usr/local/lib/python3.6/dist-packages
(from pretrainedmodels==0.7.4->segmentation-models-pytorch==0.1.0) (1.5.0+cu101)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages
(from pretrainedmodels==0.7.4->segmentation-models-pytorch==0.1.0) (4.38.0)
Requirement already satisfied: munch in /usr/local/lib/python3.6/dist-packages
(from pretrainedmodels==0.7.4->segmentation-models-pytorch==0.1.0) (2.5.0)
Requirement already satisfied: pillow>=4.1.1 in /usr/local/lib/python3.6/dist-
packages (from torchvision>=0.3.0->segmentation-models-pytorch==0.1.0) (7.0.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages
(from torchvision>=0.3.0->segmentation-models-pytorch==0.1.0) (1.18.3)
Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packages
(from torch->pretrainedmodels==0.7.4->segmentation-models-pytorch==0.1.0)
(0.16.0)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages
(from munch->pretrainedmodels==0.7.4->segmentation-models-pytorch==0.1.0)
```

```
(1.12.0)
Requirement already up-to-date: catalyst in /usr/local/lib/python3.6/dist-
packages (20.4.2)
Requirement already satisfied, skipping upgrade: scikit-image>=0.14.2 in
/usr/local/lib/python3.6/dist-packages (from catalyst) (0.16.2)
Requirement already satisfied, skipping upgrade: tqdm>=4.33.0 in
/usr/local/lib/python3.6/dist-packages (from catalyst) (4.38.0)
Requirement already satisfied, skipping upgrade: torch>=1.0.0 in
/usr/local/lib/python3.6/dist-packages (from catalyst) (1.5.0+cu101)
Requirement already satisfied, skipping upgrade: ipython in
/usr/local/lib/python3.6/dist-packages (from catalyst) (5.5.0)
Requirement already satisfied, skipping upgrade: crc32c>=1.7 in
/usr/local/lib/python3.6/dist-packages (from catalyst) (2.0)
Requirement already satisfied, skipping upgrade: plotly>=4.1.0 in
/usr/local/lib/python3.6/dist-packages (from catalyst) (4.4.1)
Requirement already satisfied, skipping upgrade: torchvision>=0.2.1 in
/usr/local/lib/python3.6/dist-packages (from catalyst) (0.6.0+cu101)
Requirement already satisfied, skipping upgrade: opencv-python in
/usr/local/lib/python3.6/dist-packages (from catalyst) (4.1.2.30)
Requirement already satisfied, skipping upgrade: tensorboard>=1.14.0 in
/usr/local/lib/python3.6/dist-packages (from catalyst) (2.2.1)
Requirement already satisfied, skipping upgrade: numpy>=1.16.4 in
/usr/local/lib/python3.6/dist-packages (from catalyst) (1.18.3)
Requirement already satisfied, skipping upgrade: imageio in
/usr/local/lib/python3.6/dist-packages (from catalyst) (2.4.1)
Requirement already satisfied, skipping upgrade: pandas>=0.22 in
/usr/local/lib/python3.6/dist-packages (from catalyst) (1.0.3)
Requirement already satisfied, skipping upgrade: scikit-learn>=0.20 in
/usr/local/lib/python3.6/dist-packages (from catalyst) (0.22.2.post1)
Requirement already satisfied, skipping upgrade: matplotlib in
/usr/local/lib/python3.6/dist-packages (from catalyst) (3.2.1)
Requirement already satisfied, skipping upgrade: GitPython>=2.1.11 in
/usr/local/lib/python3.6/dist-packages (from catalyst) (3.1.2)
Requirement already satisfied, skipping upgrade: packaging in
/usr/local/lib/python3.6/dist-packages (from catalyst) (20.3)
Requirement already satisfied, skipping upgrade: tensorboardX in
/usr/local/lib/python3.6/dist-packages (from catalyst) (2.0)
Requirement already satisfied, skipping upgrade: deprecation in
/usr/local/lib/python3.6/dist-packages (from catalyst) (2.1.0)
Requirement already satisfied, skipping upgrade: Pillow in
/usr/local/lib/python3.6/dist-packages (from catalyst) (7.0.0)
Requirement already satisfied, skipping upgrade: PyYAML in
/usr/local/lib/python3.6/dist-packages (from catalyst) (3.13)
Requirement already satisfied, skipping upgrade: scipy>=0.19.0 in
/usr/local/lib/python3.6/dist-packages (from scikit-image>=0.14.2->catalyst)
(1.4.1)
Requirement already satisfied, skipping upgrade: PyWavelets>=0.4.0 in
/usr/local/lib/python3.6/dist-packages (from scikit-image>=0.14.2->catalyst)
```

```
(1.1.1)
Requirement already satisfied, skipping upgrade: networkx>=2.0 in
/usr/local/lib/python3.6/dist-packages (from scikit-image>=0.14.2->catalyst)
(2.4)
Requirement already satisfied, skipping upgrade: future in
/usr/local/lib/python3.6/dist-packages (from torch>=1.0.0->catalyst) (0.16.0)
Requirement already satisfied, skipping upgrade: traitlets>=4.2 in
/usr/local/lib/python3.6/dist-packages (from ipython->catalyst) (4.3.3)
Requirement already satisfied, skipping upgrade: pygments in
/usr/local/lib/python3.6/dist-packages (from ipython->catalyst) (2.1.3)
Requirement already satisfied, skipping upgrade: pickleshare in
/usr/local/lib/python3.6/dist-packages (from ipython->catalyst) (0.7.5)
Requirement already satisfied, skipping upgrade: setuptools>=18.5 in
/usr/local/lib/python3.6/dist-packages (from ipython->catalyst) (46.1.3)
Requirement already satisfied, skipping upgrade: prompt-toolkit<2.0.0,>=1.0.4 in
/usr/local/lib/python3.6/dist-packages (from ipython->catalyst) (1.0.18)
Requirement already satisfied, skipping upgrade: decorator in
/usr/local/lib/python3.6/dist-packages (from ipython->catalyst) (4.4.2)
Requirement already satisfied, skipping upgrade: pexpect; sys_platform !=
"win32" in /usr/local/lib/python3.6/dist-packages (from ipython->catalyst)
(4.8.0)
Requirement already satisfied, skipping upgrade: simplegeneric>0.8 in
/usr/local/lib/python3.6/dist-packages (from ipython->catalyst) (0.8.1)
Requirement already satisfied, skipping upgrade: retrying>=1.3.3 in
/usr/local/lib/python3.6/dist-packages (from plotly>=4.1.0->catalyst) (1.3.3)
Requirement already satisfied, skipping upgrade: six in
/usr/local/lib/python3.6/dist-packages (from plotly>=4.1.0->catalyst) (1.12.0)
Requirement already satisfied, skipping upgrade: google-auth<2,>=1.6.3 in
/usr/local/lib/python3.6/dist-packages (from tensorboard>=1.14.0->catalyst)
(1.7.2)
Requirement already satisfied, skipping upgrade: google-auth-
oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.6/dist-packages (from
tensorboard>=1.14.0->catalyst) (0.4.1)
Requirement already satisfied, skipping upgrade: protobuf>=3.6.0 in
/usr/local/lib/python3.6/dist-packages (from tensorboard>=1.14.0->catalyst)
(3.10.0)
Requirement already satisfied, skipping upgrade: markdown>=2.6.8 in
/usr/local/lib/python3.6/dist-packages (from tensorboard>=1.14.0->catalyst)
(3.2.1)
Requirement already satisfied, skipping upgrade: grpcio>=1.24.3 in
/usr/local/lib/python3.6/dist-packages (from tensorboard>=1.14.0->catalyst)
(1.28.1)
Requirement already satisfied, skipping upgrade: tensorboard-plugin-wit>=1.6.0
in /usr/local/lib/python3.6/dist-packages (from tensorboard>=1.14.0->catalyst)
(1.6.0.post3)
Requirement already satisfied, skipping upgrade: requests<3,>=2.21.0 in
/usr/local/lib/python3.6/dist-packages (from tensorboard>=1.14.0->catalyst)
(2.23.0)
```

```
Requirement already satisfied, skipping upgrade: wheel>=0.26; python_version >=
"3" in /usr/local/lib/python3.6/dist-packages (from
tensorboard>=1.14.0->catalyst) (0.34.2)
Requirement already satisfied, skipping upgrade: absl-py>=0.4 in
/usr/local/lib/python3.6/dist-packages (from tensorboard>=1.14.0->catalyst)
(0.9.0)
Requirement already satisfied, skipping upgrade: werkzeug>=0.11.15 in
/usr/local/lib/python3.6/dist-packages (from tensorboard>=1.14.0->catalyst)
(1.0.1)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.6.1 in
/usr/local/lib/python3.6/dist-packages (from pandas>=0.22->catalyst) (2.8.1)
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in
/usr/local/lib/python3.6/dist-packages (from pandas>=0.22->catalyst) (2018.9)
Requirement already satisfied, skipping upgrade: joblib>=0.11 in
/usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.20->catalyst)
(0.14.1)
Requirement already satisfied, skipping upgrade: cycler>=0.10 in
/usr/local/lib/python3.6/dist-packages (from matplotlib->catalyst) (0.10.0)
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in
/usr/local/lib/python3.6/dist-packages (from matplotlib->catalyst) (1.2.0)
Requirement already satisfied, skipping upgrade:
pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-
packages (from matplotlib->catalyst) (2.4.7)
Requirement already satisfied, skipping upgrade: gitdb<5,>=4.0.1 in
/usr/local/lib/python3.6/dist-packages (from GitPython>=2.1.11->catalyst)
(4.0.5)
Requirement already satisfied, skipping upgrade: ipython-genutils in
/usr/local/lib/python3.6/dist-packages (from traitlets>=4.2->ipython->catalyst)
(0.2.0)
Requirement already satisfied, skipping upgrade: wcwidth in
/usr/local/lib/python3.6/dist-packages (from prompt-
toolkit<2.0.0,>=1.0.4->ipython->catalyst) (0.1.9)
Requirement already satisfied, skipping upgrade: ptyprocess>=0.5 in
/usr/local/lib/python3.6/dist-packages (from pexpect; sys_platform !=
"win32"->ipython->catalyst) (0.6.0)
Requirement already satisfied, skipping upgrade: rsa<4.1,>=3.1.4 in
/usr/local/lib/python3.6/dist-packages (from google-
auth<2,>=1.6.3->tensorboard>=1.14.0->catalyst) (4.0)
Requirement already satisfied, skipping upgrade: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.6/dist-packages (from google-
auth<2,>=1.6.3->tensorboard>=1.14.0->catalyst) (0.2.8)
Requirement already satisfied, skipping upgrade: cachetools<3.2,>=2.0.0 in
/usr/local/lib/python3.6/dist-packages (from google-
auth<2,>=1.6.3->tensorboard>=1.14.0->catalyst) (3.1.1)
Requirement already satisfied, skipping upgrade: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.6/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard>=1.14.0->catalyst) (1.3.0)
Requirement already satisfied, skipping upgrade: certifi>=2017.4.17 in
```

```
/usr/local/lib/python3.6/dist-packages (from
requests<3,>=2.21.0->tensorboard>=1.14.0->catalyst) (2020.4.5.1)
Requirement already satisfied, skipping upgrade: idna<3,>=2.5 in
/usr/local/lib/python3.6/dist-packages (from
requests<3,>=2.21.0->tensorboard>=1.14.0->catalyst) (2.9)
Requirement already satisfied, skipping upgrade:
urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-
packages (from requests<3,>=2.21.0->tensorboard>=1.14.0->catalyst) (1.24.3)
Requirement already satisfied, skipping upgrade: chardet<4,>=3.0.2 in
/usr/local/lib/python3.6/dist-packages (from
requests<3,>=2.21.0->tensorboard>=1.14.0->catalyst) (3.0.4)
Requirement already satisfied, skipping upgrade: smmap<4,>=3.0.1 in
/usr/local/lib/python3.6/dist-packages (from
gitdb<5,>=4.0.1->GitPython>=2.1.11->catalyst) (3.0.4)
Requirement already satisfied, skipping upgrade: pyasn1>=0.1.3 in
/usr/local/lib/python3.6/dist-packages (from rsa<4.1,>=3.1.4->google-
auth<2,>=1.6.3->tensorboard>=1.14.0->catalyst) (0.4.8)
Requirement already satisfied, skipping upgrade: oauthlib>=3.0.0 in
/usr/local/lib/python3.6/dist-packages (from requests-oauthlib>=0.7.0->google-
auth-oauthlib<0.5,>=0.4.1->tensorboard>=1.14.0->catalyst) (3.1.0)
```

[3]:
```
#Dependencies

#Handles data
import json
import numpy as np
import matplotlib.pyplot as plt
import cv2
import glob
from operator import itemgetter
import pickle

#Torch utilities
from typing import List
from pathlib import Path
from torch.utils.data import Dataset
import torch

#Data Loader utilities
import collections
from sklearn.model_selection import train_test_split
from torch.utils.data import DataLoader

#Model building and training
import segmentation_models_pytorch as smp
from torch import nn
```

```python
from catalyst.contrib.nn import DiceLoss, IoULoss
from torch import optim
from torch.optim import AdamW
from catalyst import utils

from catalyst.contrib.nn import RAdam, Lookahead
from catalyst.dl import SupervisedRunner

from catalyst.dl.callbacks import DiceCallback, IouCallback, \
  CriterionCallback, AccuracyCallback, MulticlassDiceMetricCallback
```

```
/usr/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning:

numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from
C header, got 216 from PyObject

/usr/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning:

numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from
C header, got 216 from PyObject

/usr/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning:

numpy.ufunc size changed, may indicate binary incompatibility. Expected 192 from
C header, got 216 from PyObject
```

[4]:
```python
#Set seed for better reproducibility
SEED = 42
utils.set_global_seed(SEED)
utils.prepare_cudnn(deterministic=True)
```

```
/usr/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning:

numpy.ufunc size changed, may indicate binary incompatibility. Expected 216, got
192

/usr/lib/python3.6/importlib/_bootstrap.py:219: ImportWarning:

can't resolve package from __spec__ or __package__, falling back on __name__ and
__path__
```

[0]:
```python
#Define and establish a dataset class
class SegmentationDataset(Dataset):
    def __init__(
        self,
        image_arr_path,
        mask_arr_path,
    ) -> None:
        self.images = np.load(image_arr_path)
        self.masks = np.load(mask_arr_path)

    def __len__(self) -> int:
        return len(self.images)

    def __getitem__(self, idx: int) -> dict:
        image = self.images[idx]
        image = np.swapaxes(image, 2, 0)
        image = np.swapaxes(image, 2, 1)
        image = torch.from_numpy(image).float()
        result = {"image": image}

        if self.masks is not None:
            mask = self.masks[idx]
            mask = np.swapaxes(mask, 2, 0)
            mask = np.swapaxes(mask, 2, 1)
            mask = torch.from_numpy(mask).float()
            result["mask"] = mask
        return result
```

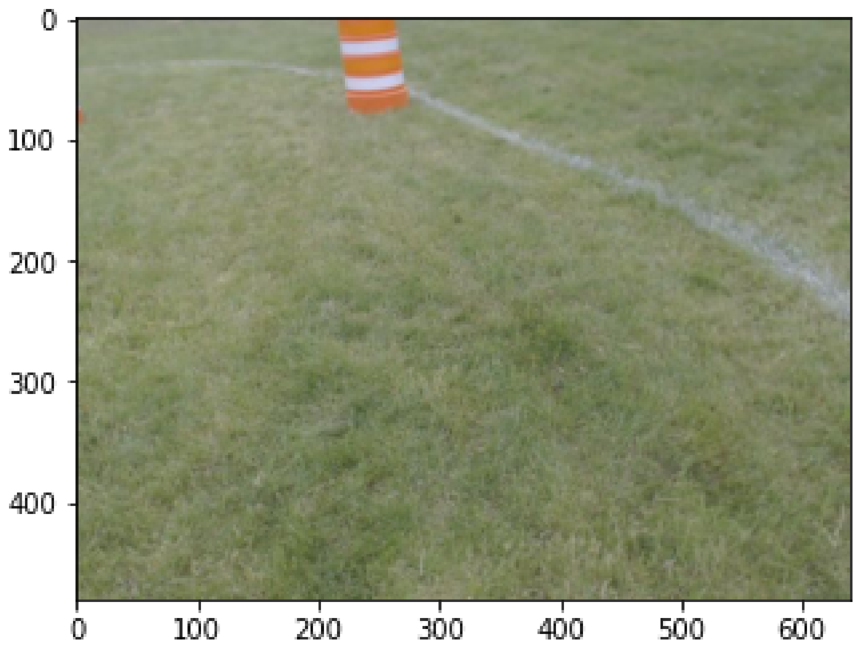
[0]:
```
#Load dataset once to enable visualizion prior to model training
dset = SegmentationDataset(image_arr_path="/content/drive/Shared drives/
↪Intelligent Ground Vehicle Competition/Previous Year Resources/2019-2020␣
↪Season/Software/Collab Notebooks/train_images.npy",
                           mask_arr_path="/content/drive/Shared drives/
↪Intelligent Ground Vehicle Competition/Previous Year Resources/2019-2020␣
↪Season/Software/Collab Notebooks/train_masks.npy")
```

[7]:
```
#Show sizes of the image and mask
out = dset[0]
out["image"].shape, out["mask"].shape, len(dset)
```

[7]:
```
(torch.Size([3, 480, 640]), torch.Size([1, 480, 640]), 592)
```

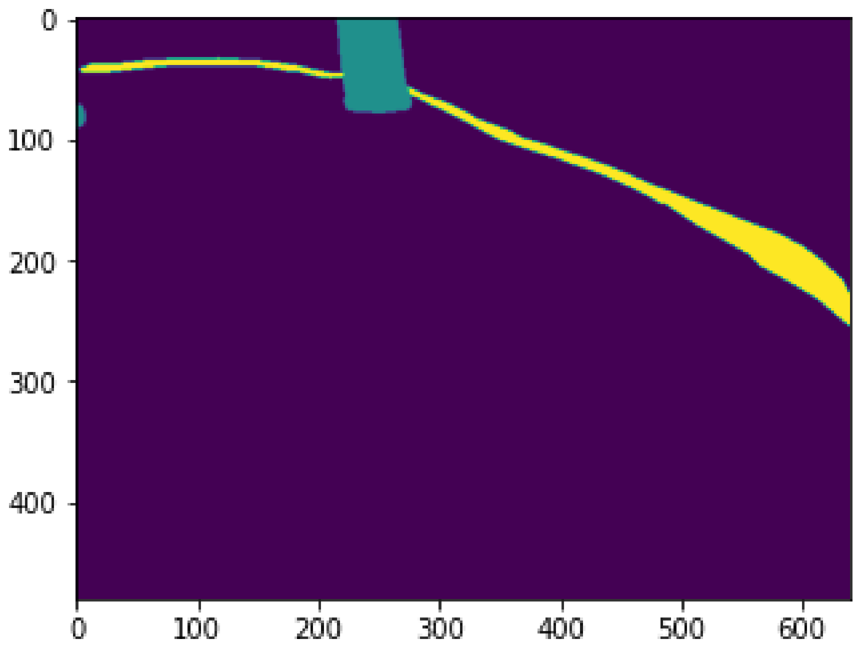
[9]:
```
#Show an image
image = np.asarray(dset[40]['image'])
image = np.swapaxes(image, 2, 0)
image = np.swapaxes(image, 1, 0)
image = image.astype(np.uint8)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
```

[9]:
```
<matplotlib.image.AxesImage at 0x7efd7710b550>
```



```
[10]: #Show associated mask
      mask = np.squeeze(dset[40]['mask'])
      plt.imshow(mask)
```

```
[10]: <matplotlib.image.AxesImage at 0x7efd76894b38>
```



[0]:
```
#Set up U-Net with EfficientNet backbone pretrained on ImageNet

ENCODER = 'efficientnet-b3'
ENCODER_WEIGHTS = 'imagenet'
DEVICE = 'cuda'
ACTIVATION = None

model = smp.Unet(
    encoder_name=ENCODER,
    encoder_weights=ENCODER_WEIGHTS,
    classes=3,
    activation=ACTIVATION,
)
```

[0]:
```
#Define loaders

def get_loaders(
    images: List[Path],
    masks: List[Path],
    image_arr_path: str,
    mask_arr_path: str,
    random_state: int,
    valid_size: float = 0.1,
    batch_size: int = 12,
    num_workers: int = 4,
    ) -> dict:

    indices = np.arange(len(images))

    train_indices, valid_indices = train_test_split(
      indices, test_size=valid_size, random_state=random_state, shuffle=True
    )

    np_images = np.array(images)
    np_masks = np.array(masks)


    train_dataset = SegmentationDataset(image_arr_path, mask_arr_path)
    train_dataset.images = np_images[train_indices]
    train_dataset.masks = np_masks[train_indices]

    valid_dataset = SegmentationDataset(image_arr_path, mask_arr_path)
    valid_dataset.images = np_images[valid_indices]
    valid_dataset.masks = np_masks[valid_indices]

    train_loader = DataLoader(
      train_dataset,
      batch_size=batch_size,
      shuffle=False,
      num_workers=num_workers,
      drop_last=False,
    )

    valid_loader = DataLoader(
      valid_dataset,
      batch_size=batch_size,
      shuffle=False,
      num_workers=num_workers,
      drop_last=False,
    )

    loaders = collections.OrderedDict()
    loaders["train"] = train_loader
    loaders["valid"] = valid_loader

    return loaders
```

[0]:
```
#Get loaders

loaders = get_loaders(
    images=np.load("/content/drive/Shared drives/Intelligent Ground Vehicle␣
 ↪Competition/Previous Year Resources/2019-2020 Season/Software/Collab␣
 ↪Notebooks/train_images.npy"),
    masks=np.load("/content/drive/Shared drives/Intelligent Ground Vehicle␣
 ↪Competition/Previous Year Resources/2019-2020 Season/Software/Collab␣
 ↪Notebooks/train_masks.npy"),
    image_arr_path="/content/drive/Shared drives/Intelligent Ground Vehicle␣
 ↪Competition/Previous Year Resources/2019-2020 Season/Software/Collab␣
 ↪Notebooks/train_images.npy",
    mask_arr_path="/content/drive/Shared drives/Intelligent Ground Vehicle␣
 ↪Competition/Previous Year Resources/2019-2020 Season/Software/Collab␣
 ↪Notebooks/train_masks.npy",
    random_state=420,
    valid_size=0.1,
    batch_size=3,
    num_workers=2,
)
```

[0]:
```
#    Helpful code taken from Joseph Chen
#
#    Copyright 2019 Division of Medical Image Computing, German Cancer Research␣
 ↪Center (DKFZ), Heidelberg, Germany
```

```python
#
#    Licensed under the Apache License, Version 2.0 (the "License");
#    you may not use this file except in compliance with the License.
#    You may obtain a copy of the License at
#
#        http://www.apache.org/licenses/LICENSE-2.0
#
#    Unless required by applicable law or agreed to in writing, software
#    distributed under the License is distributed on an "AS IS" BASIS,
#    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#    See the License for the specific language governing permissions and
#    limitations under the License.

import torch
from torch import nn
import numpy as np

def sum_tensor(inp, axes, keepdim=False):
    axes = np.unique(axes).astype(int)
    if keepdim:
        for ax in axes:
            inp = inp.sum(int(ax), keepdim=True)
    else:
        for ax in sorted(axes, reverse=True):
            inp = inp.sum(int(ax))
    return inp

def softmax_helper(x):
    rpt = [1 for _ in range(len(x.size()))]
    rpt[1] = x.size(1)
    x_max = x.max(1, keepdim=True)[0].repeat(*rpt)
    e_x = torch.exp(x - x_max)
    return e_x / e_x.sum(1, keepdim=True).repeat(*rpt)

class CrossentropyND(nn.CrossEntropyLoss):
    """
    Network has to have NO NONLINEARITY!
    """
    def forward(self, inp, target):
        target = target.long()
        num_classes = inp.size()[1]

        i0 = 1
        i1 = 2

        while i1 < len(inp.shape):
            inp = inp.transpose(i0, i1)
            i0 += 1
            i1 += 1

        inp = inp.contiguous()
        inp = inp.view(-1, num_classes)

        target = target.view(-1,)

        return super(CrossentropyND, self).forward(inp, target)

def get_tp_fp_fn(net_output, gt, axes=None, mask=None, square=False):
    """
    net_output must be (b, c, x, y(, z)))
    gt must be a label map (shape (b, 1, x, y(, z)) OR shape (b, x, y(, z))) or␣
 ↪one hot encoding (b, c, x, y(, z))
    if mask is provided it must have shape (b, 1, x, y(, z)))
    :param net_output:
    :param gt:
    :param axes:
    :param mask: mask must be 1 for valid pixels and 0 for invalid pixels
    :param square: if True then fp, tp and fn will be squared before summation
    :return:
    """
    if axes is None:
        axes = tuple(range(2, len(net_output.size())))

    shp_x = net_output.shape
    shp_y = gt.shape

    with torch.no_grad():
        if len(shp_x) != len(shp_y):
            gt = gt.view((shp_y[0], 1, *shp_y[1:]))

        if all([i == j for i, j in zip(net_output.shape, gt.shape)]):
            # if this is the case then gt is probably already a one hot encoding
            y_onehot = gt
        else:
            gt = gt.long()
            y_onehot = torch.zeros(shp_x)
            if net_output.device.type == "cuda":
                y_onehot = y_onehot.cuda(net_output.device.index)
            y_onehot.scatter_(1, gt, 1)

    tp = net_output * y_onehot
    fp = net_output * (1 - y_onehot)
    fn = (1 - net_output) * y_onehot
    if mask is not None:
        tp = torch.stack(tuple(x_i * mask[:, 0] for x_i in torch.unbind(tp,␣
↪dim=1)), dim=1)
        fp = torch.stack(tuple(x_i * mask[:, 0] for x_i in torch.unbind(fp,␣
↪dim=1)), dim=1)
        fn = torch.stack(tuple(x_i * mask[:, 0] for x_i in torch.unbind(fn,␣
↪dim=1)), dim=1)

    if square:
        tp = tp ** 2
        fp = fp ** 2
        fn = fn ** 2

    tp = sum_tensor(tp, axes, keepdim=False)
    fp = sum_tensor(fp, axes, keepdim=False)
    fn = sum_tensor(fn, axes, keepdim=False)

    return tp, fp, fn


class SoftDiceLoss(nn.Module):
    def __init__(self, apply_nonlin=None, batch_dice=False, do_bg=True,
                 smooth=1., square=False):
        super(SoftDiceLoss, self).__init__()

        self.square = square
        self.do_bg = do_bg
        self.batch_dice = batch_dice
        self.apply_nonlin = apply_nonlin
        self.smooth = smooth

    def forward(self, x, y, loss_mask=None):
        shp_x = x.shape

        if self.batch_dice:
            axes = [0] + list(range(2, len(shp_x)))
        else:
            axes = list(range(2, len(shp_x)))

        if self.apply_nonlin is not None:
            x = self.apply_nonlin(x)

        tp, fp, fn = get_tp_fp_fn(x, y, axes, loss_mask, self.square)

        dc = (2 * tp + self.smooth) / (2 * tp + fp + fn + self.smooth)

        if not self.do_bg:
            if self.batch_dice:
                dc = dc[1:]
            else:
                dc = dc[:, 1:]
        dc = dc.mean()

        return -dc


class DC_and_CE_loss(nn.Module):
    def __init__(self, soft_dice_kwargs, ce_kwargs, aggregate="sum"):
        super(DC_and_CE_loss, self).__init__()
        self.aggregate = aggregate
        self.ce = CrossentropyND(**ce_kwargs)
        self.dc = SoftDiceLoss(apply_nonlin=softmax_helper, **soft_dice_kwargs)

    def forward(self, net_output, target):
        dc_loss = self.dc(net_output, target)
        ce_loss = self.ce(net_output, target)
        if self.aggregate == "sum":
            result = ce_loss + dc_loss
        else:
            raise NotImplementedError("did not work")
        return result
```

[0]:
```
#Define loss criterion
criterion = {
    "CE": CrossentropyND(),
}

#Configure model optimization
learning_rate = 0.001
encoder_learning_rate = 0.0005
encoder_weight_decay = 0.00003
optimizer_weight_decay = 0.0003
optim_factor = 0.25
optim_patience = 2

optimizer = AdamW(model.parameters(), lr=0.001, betas=(0.9, 0.999), eps=1e-08,␣
 ↪weight_decay=0.01, amsgrad=False)

#Use scheduler for improved results
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer,␣
 ↪factor=optim_factor, patience=optim_patience)

num_epochs = 10
device = utils.get_device()
```

```
runner = SupervisedRunner(device=device, input_key="image",␣
 ↪input_target_key="mask")
```

[0]:
```
#Establish calculations during training through Catalyst callbacks
callbacks = [
        CriterionCallback(
            input_key="mask",
            prefix="loss",
            criterion_key="CE"
        ),
        MulticlassDiceMetricCallback(input_key="mask")
        ]
```

[17]:
```
#Run training loop
runner.train(
    model=model,
    criterion=criterion,
    optimizer=optimizer,
    scheduler=scheduler,
    loaders=loaders,
    callbacks=callbacks,
    logdir='content/full_model2', #this logdir must be changed with every new␣
 ↪run
    num_epochs=num_epochs,
    main_metric="loss",
    minimize_metric=True,
    fp16=None,
    verbose=True,
)
```

```
1/10 * Epoch (train):   0% 0/178 [00:00<?, ?it/s]

/usr/local/lib/python3.6/dist-packages/efficientnet_pytorch/utils.py:45:
DeprecationWarning:

'saved_variables' is deprecated; use 'saved_tensors'


1/10 * Epoch (train):   1% 1/178 [00:04<14:02,  4.76s/it, loss=1.809]

/pytorch/torch/csrc/utils/python_arg_parser.cpp:756: UserWarning:

This overload of add is deprecated:
        add(Number alpha, Tensor other)
Consider using one of the following signatures instead:
        add(Tensor other, *, Number alpha)
```

```
1/10 * Epoch (train): 100% 178/178 [01:19<00:00,  2.25it/s, loss=0.053]
1/10 * Epoch (valid): 100% 20/20 [00:02<00:00,  7.02it/s, loss=0.131]
[2020-05-06 03:12:32,198]
1/10 * Epoch 1 (_base): lr=0.0010 | momentum=0.9000
1/10 * Epoch 1 (train): dice_0=0.9633 | dice_1=0.0117 | dice_2=0.5902 |
dice_mean=0.5218 | loss=0.2070
1/10 * Epoch 1 (valid): dice_0=0.9830 | dice_1=0.4567 | dice_2=0.7392 |
dice_mean=0.7263 | loss=0.1035
2/10 * Epoch (train): 100% 178/178 [01:10<00:00,  2.52it/s, loss=0.042]
2/10 * Epoch (valid): 100% 20/20 [00:02<00:00,  6.94it/s, loss=0.087]
[2020-05-06 03:13:47,599]
2/10 * Epoch 2 (_base): lr=0.0010 | momentum=0.9000
2/10 * Epoch 2 (train): dice_0=0.9899 | dice_1=0.5369 | dice_2=0.8212 |
dice_mean=0.7827 | loss=0.0570
2/10 * Epoch 2 (valid): dice_0=0.9871 | dice_1=0.6465 | dice_2=0.7723 |
dice_mean=0.8020 | loss=0.0754
3/10 * Epoch (train): 100% 178/178 [01:10<00:00,  2.53it/s, loss=0.034]
3/10 * Epoch (valid): 100% 20/20 [00:02<00:00,  7.16it/s, loss=0.062]
[2020-05-06 03:15:02,779]
3/10 * Epoch 3 (_base): lr=0.0010 | momentum=0.9000
3/10 * Epoch 3 (train): dice_0=0.9916 | dice_1=0.7126 | dice_2=0.8468 |
dice_mean=0.8503 | loss=0.0443
3/10 * Epoch 3 (valid): dice_0=0.9880 | dice_1=0.7442 | dice_2=0.7713 |
dice_mean=0.8345 | loss=0.0698
4/10 * Epoch (train): 100% 178/178 [01:10<00:00,  2.53it/s, loss=0.033]
4/10 * Epoch (valid): 100% 20/20 [00:02<00:00,  7.08it/s, loss=0.043]
[2020-05-06 03:16:18,282]
4/10 * Epoch 4 (_base): lr=0.0010 | momentum=0.9000
4/10 * Epoch 4 (train): dice_0=0.9936 | dice_1=0.8940 | dice_2=0.8615 |
dice_mean=0.9164 | loss=0.0350
4/10 * Epoch 4 (valid): dice_0=0.9888 | dice_1=0.8702 | dice_2=0.7936 |
dice_mean=0.8842 | loss=0.0689
5/10 * Epoch (train): 100% 178/178 [01:10<00:00,  2.52it/s, loss=0.030]
5/10 * Epoch (valid): 100% 20/20 [00:02<00:00,  7.08it/s, loss=0.042]
[2020-05-06 03:17:33,545]
5/10 * Epoch 5 (_base): lr=0.0010 | momentum=0.9000
5/10 * Epoch 5 (train): dice_0=0.9939 | dice_1=0.9067 | dice_2=0.8665 |
dice_mean=0.9224 | loss=0.0328
5/10 * Epoch 5 (valid): dice_0=0.9890 | dice_1=0.9114 | dice_2=0.8091 |
dice_mean=0.9031 | loss=0.0744
6/10 * Epoch (train): 100% 178/178 [01:10<00:00,  2.52it/s, loss=0.027]
6/10 * Epoch (valid): 100% 20/20 [00:02<00:00,  7.05it/s, loss=0.042]
[2020-05-06 03:18:48,028]
6/10 * Epoch 6 (_base): lr=0.0010 | momentum=0.9000
6/10 * Epoch 6 (train): dice_0=0.9942 | dice_1=0.9276 | dice_2=0.8717 |
dice_mean=0.9312 | loss=0.0304
6/10 * Epoch 6 (valid): dice_0=0.9852 | dice_1=0.7874 | dice_2=0.6706 |
dice_mean=0.8144 | loss=0.0824
```

```
7/10 * Epoch (train): 100% 178/178 [01:10<00:00,  2.53it/s, loss=0.023]
7/10 * Epoch (valid): 100% 20/20 [00:02<00:00,  7.00it/s, loss=0.024]
[2020-05-06 03:20:02,766]
7/10 * Epoch 7 (_base): lr=0.0010 | momentum=0.9000
7/10 * Epoch 7 (train): dice_0=0.9943 | dice_1=0.9276 | dice_2=0.8752 |
dice_mean=0.9324 | loss=0.0293
7/10 * Epoch 7 (valid): dice_0=0.9903 | dice_1=0.9452 | dice_2=0.8133 |
dice_mean=0.9163 | loss=0.0635
8/10 * Epoch (train): 100% 178/178 [01:10<00:00,  2.53it/s, loss=0.021]
8/10 * Epoch (valid): 100% 20/20 [00:02<00:00,  7.02it/s, loss=0.021]
[2020-05-06 03:21:17,944]
8/10 * Epoch 8 (_base): lr=0.0010 | momentum=0.9000
8/10 * Epoch 8 (train): dice_0=0.9952 | dice_1=0.9630 | dice_2=0.8904 |
dice_mean=0.9495 | loss=0.0240
8/10 * Epoch 8 (valid): dice_0=0.9907 | dice_1=0.9427 | dice_2=0.8233 |
dice_mean=0.9189 | loss=0.0634
9/10 * Epoch (train): 100% 178/178 [01:10<00:00,  2.53it/s, loss=0.017]
9/10 * Epoch (valid): 100% 20/20 [00:02<00:00,  6.99it/s, loss=0.021]
[2020-05-06 03:22:32,991]
9/10 * Epoch 9 (_base): lr=0.0010 | momentum=0.9000
9/10 * Epoch 9 (train): dice_0=0.9956 | dice_1=0.9658 | dice_2=0.9006 |
dice_mean=0.9540 | loss=0.0213
9/10 * Epoch 9 (valid): dice_0=0.9904 | dice_1=0.9552 | dice_2=0.8139 |
dice_mean=0.9199 | loss=0.0638
10/10 * Epoch (train): 100% 178/178 [01:10<00:00,  2.53it/s, loss=0.019]
10/10 * Epoch (valid): 100% 20/20 [00:02<00:00,  7.10it/s, loss=0.020]
[2020-05-06 03:23:47,139]
10/10 * Epoch 10 (_base): lr=0.0010 | momentum=0.9000
10/10 * Epoch 10 (train): dice_0=0.9959 | dice_1=0.9723 | dice_2=0.9058 |
dice_mean=0.9580 | loss=0.0197
10/10 * Epoch 10 (valid): dice_0=0.9907 | dice_1=0.9630 | dice_2=0.8243 |
dice_mean=0.9260 | loss=0.0664
Top best models:
content/full_model2/checkpoints/train.8.pth     0.0634
```

[0]:
```
#Test model on test dataset
test_data = SegmentationDataset("/content/drive/Shared drives/Intelligent␣
 ↪Ground Vehicle Competition/Previous Year Resources/2019-2020 Season/Software/
 ↪Collab Notebooks/test_images.npy",
                                "/content/drive/Shared drives/Intelligent␣
 ↪Ground Vehicle Competition/Previous Year Resources/2019-2020 Season/Software/
 ↪Collab Notebooks/test_masks.npy")
```

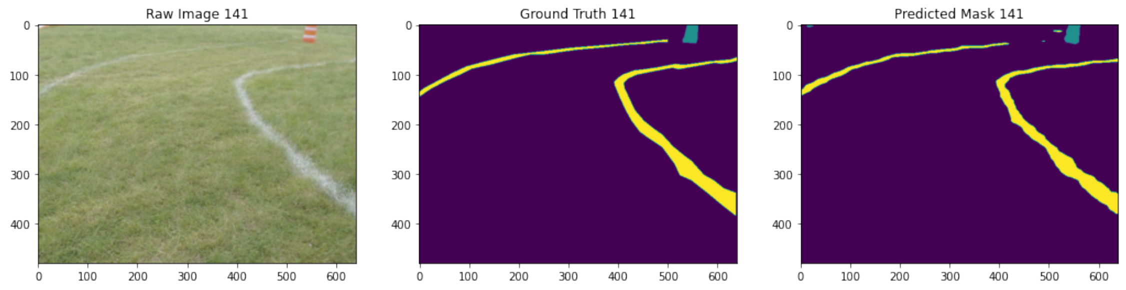
[0]:
```
#Create loader for predictions
infer_loader = DataLoader(
    test_data,
    batch_size=12,
    shuffle=False,
    num_workers=4
)
```

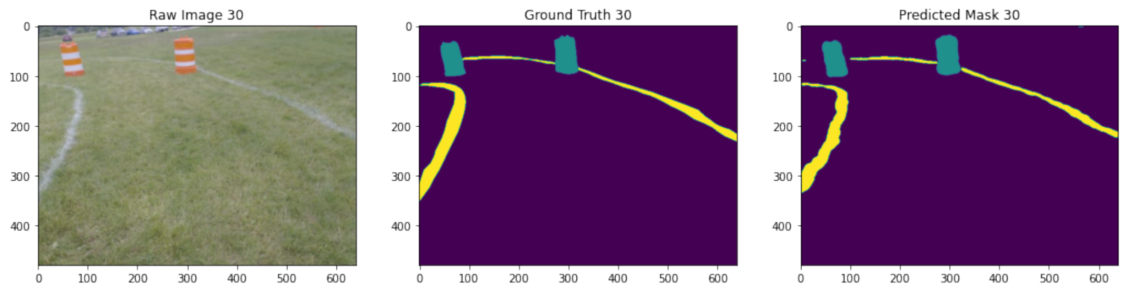
[20]:
```
#Get model predictions on test dataset
predictions = np.vstack(list(map(
    lambda x: x["logits"].cpu().numpy(),
    runner.predict_loader(loader=infer_loader, resume=f"content/full_model2/
 ↪checkpoints/best.pth")
)))

print(type(predictions))
print(predictions.shape)
```

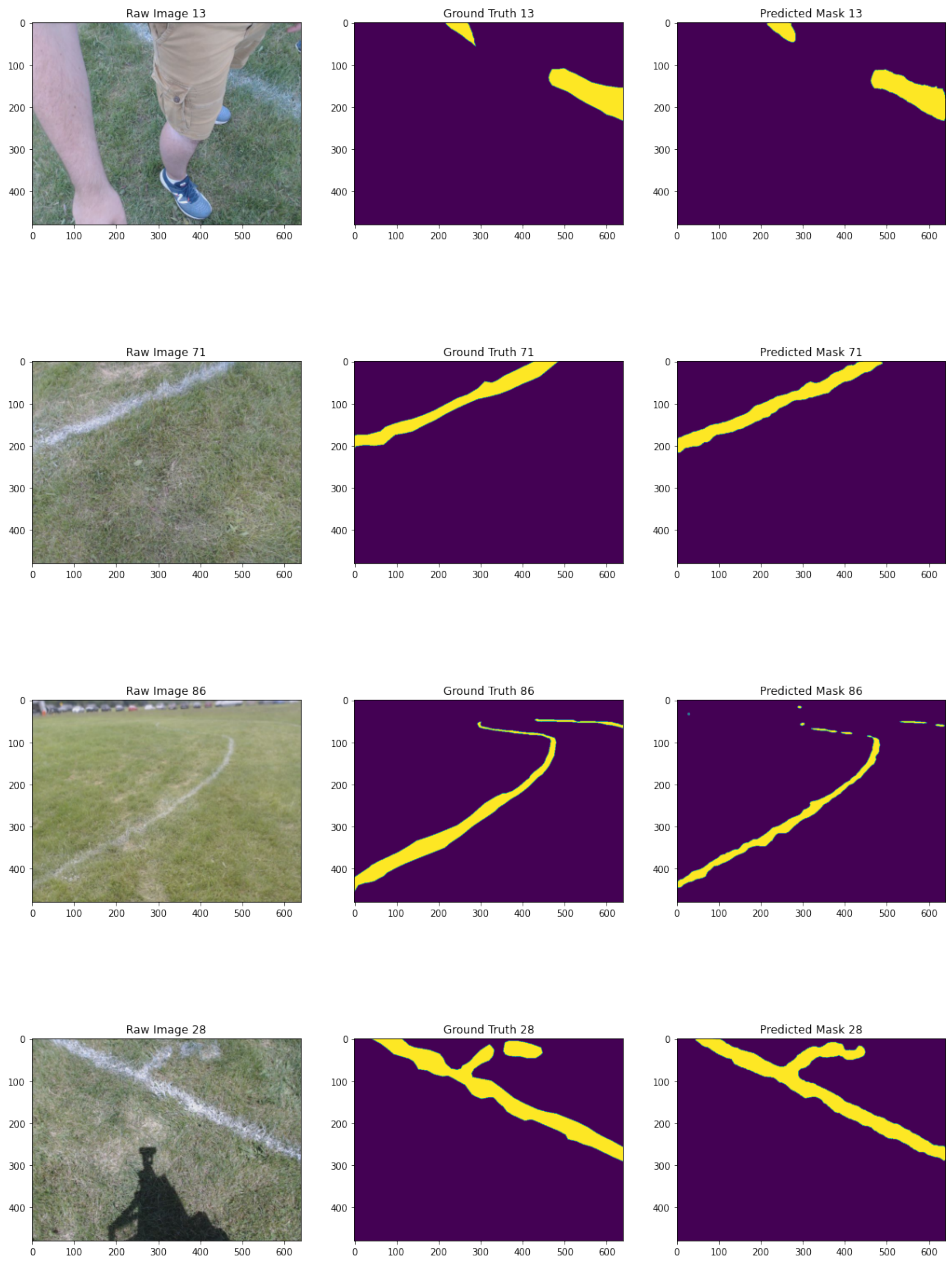
```
<class 'numpy.ndarray'>
(149, 3, 480, 640)
```

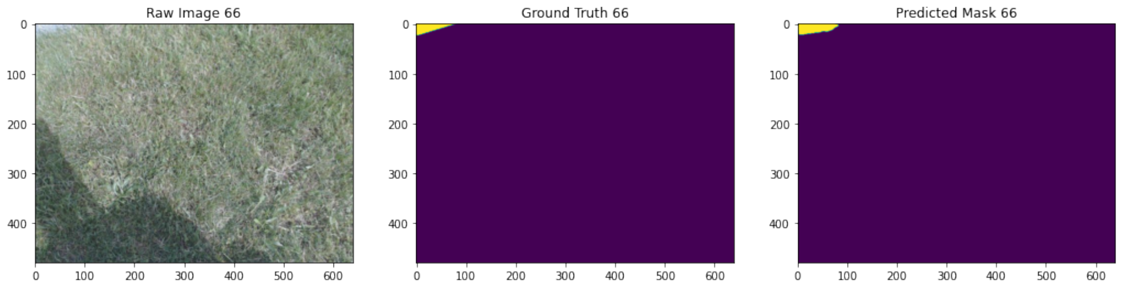
[34]:
```
#Display sample prediction results

pred_results = {}
rand_nums = np.random.randint(low=1, high=148, size=18)
rand_nums = np.insert(rand_nums, 0, 30)
rand_nums = np.insert(rand_nums, 0, 141)

for num in rand_nums:

    #Show image
    image = np.asarray(test_data[num]['image'])
    image = np.swapaxes(image, 2, 0)
    image = np.swapaxes(image, 1, 0)
    image = image.astype(np.uint8)


    #Show mask
    mask = np.squeeze(test_data[num]['mask'])

    #Show predicted mask
    pred_mask = np.asarray(predictions[num])
    pred_mask = np.swapaxes(pred_mask, 2, 0)
    pred_mask = np.swapaxes(pred_mask, 1, 0)
    pred_mask = pred_mask.astype(np.float64)
    pred_mask = np.argmax(pred_mask, axis=2)

    #Add three images to list
    images = []
    images.append(image)
    images.append(mask)
```

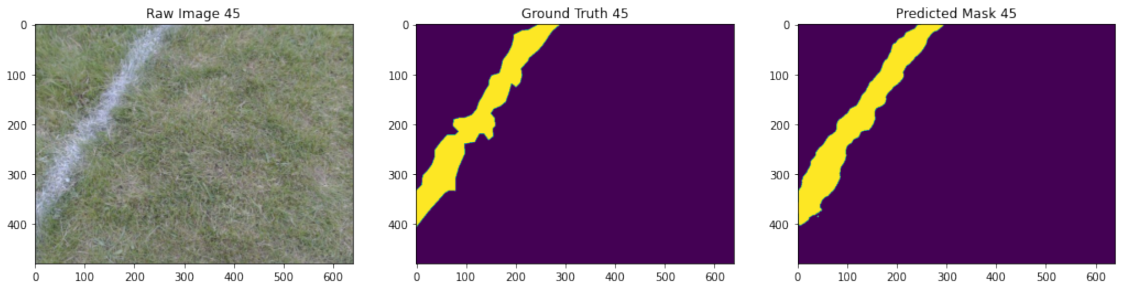
```python
images.append(pred_mask)

#Show and plot all three images
plt.figure(figsize=(30,30))
columns = 5
for i, image in enumerate(images):
    image_plot = plt.subplot(len(images) / columns + 1, columns, i + 1)
    if i == 0:
        label = 'Raw Image {}'.format(num)
        image_plot.set_title(label)
        result = plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    elif i == 1:
        label = 'Ground Truth {}'.format(num)
        image_plot.set_title(label)
        result = plt.imshow(image)
    elif i == 2:
        label = 'Predicted Mask {}'.format(num)
        image_plot.set_title(label)
        result = plt.imshow(image)
    pred_results[label] = result
```

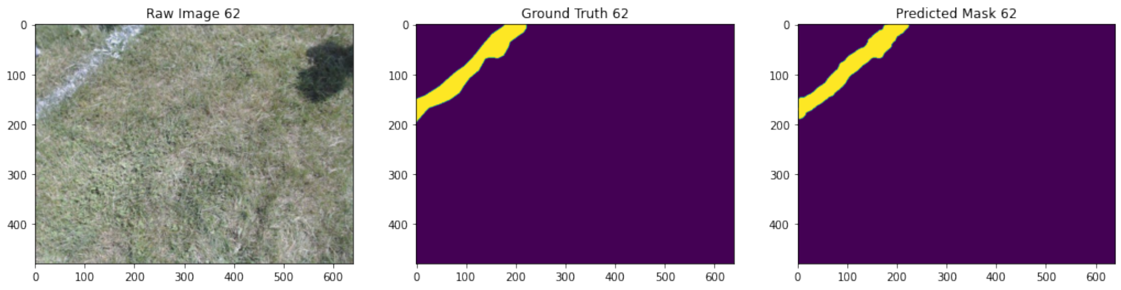


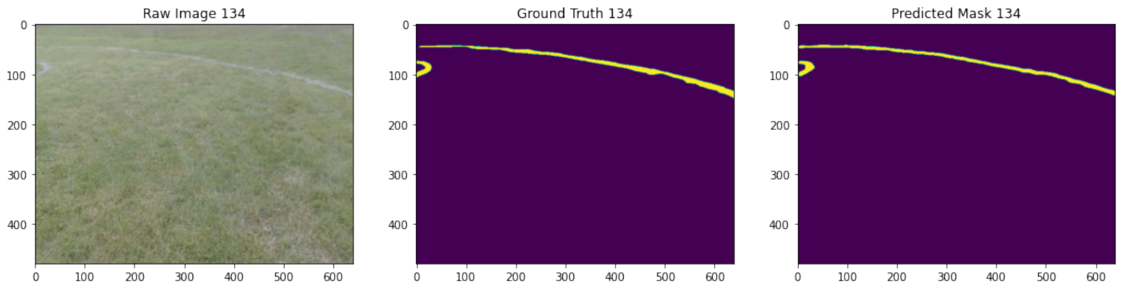


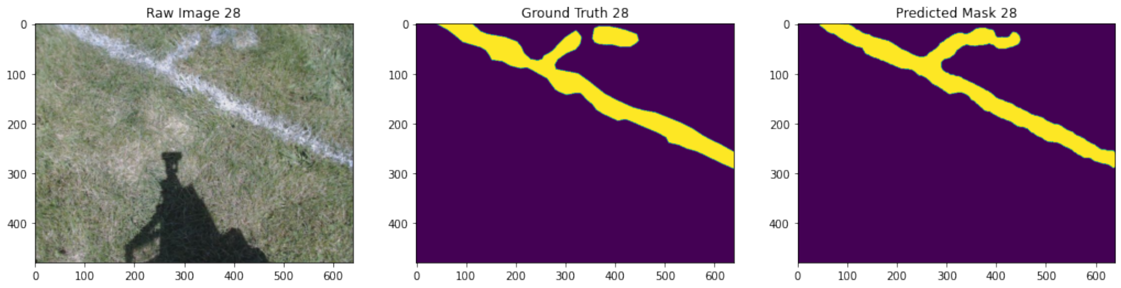

Raw Image 13
Ground Truth 13
Predicted Mask 13
Raw Image 71
Ground Truth 71
Predicted Mask 71
Raw Image 86
Ground Truth 86
Predicted Mask 86
Raw Image 28
Ground Truth 28
Predicted Mask 28

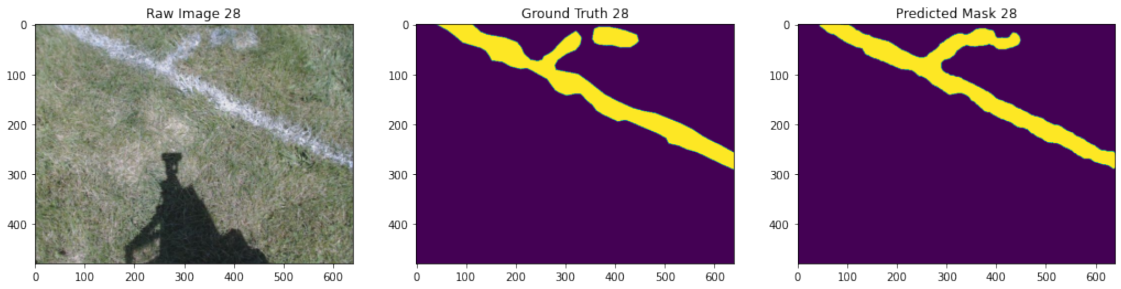

Raw Image 66
Ground Truth 66
Predicted Mask 66

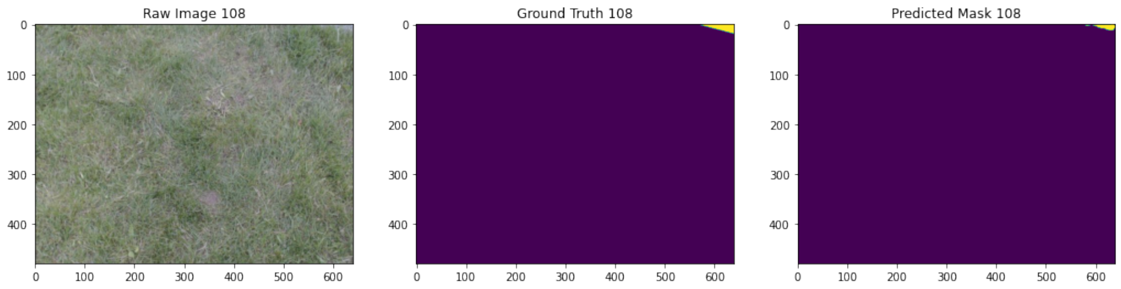

Raw Image 45
Ground Truth 45
Predicted Mask 45

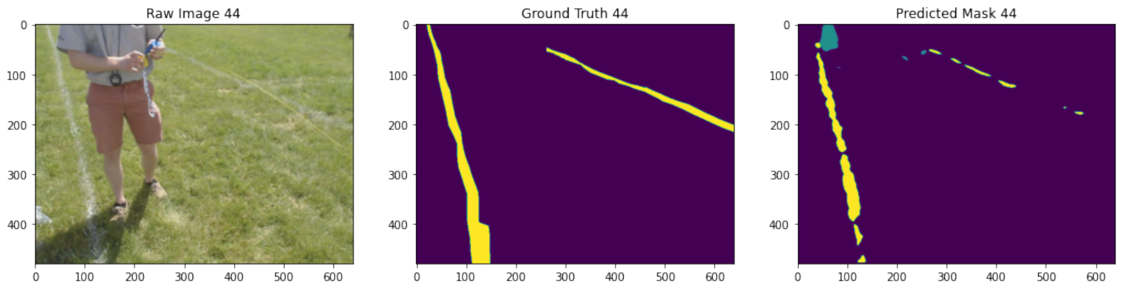

Raw Image 62
Ground Truth 62
Predicted Mask 62

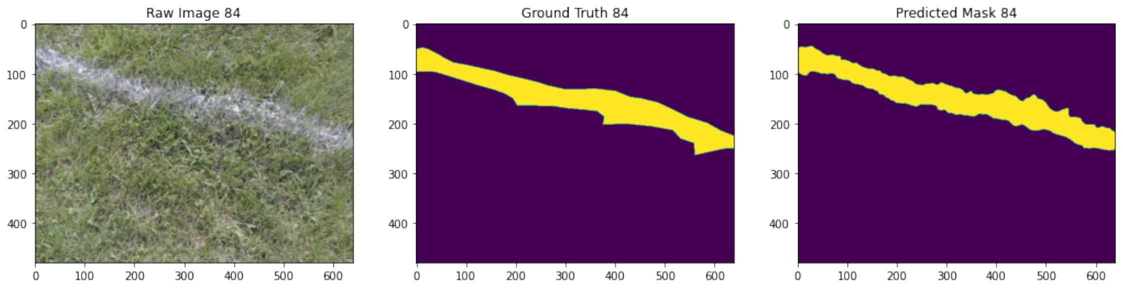

Raw Image 134
Ground Truth 134
Predicted Mask 134

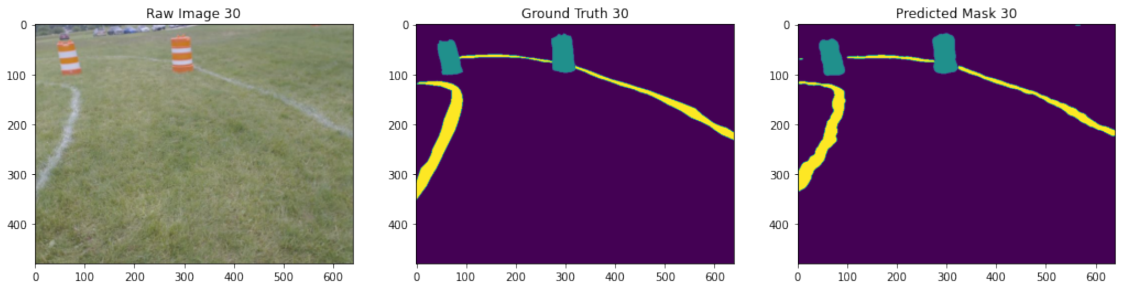

Raw Image 28
Ground Truth 28
Predicted Mask 28
0
100
200
300
400
500
600

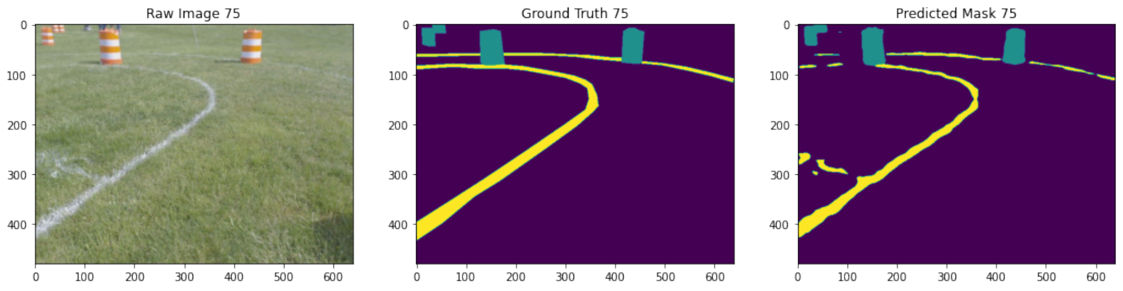

Raw Image 28
Ground Truth 28
Predicted Mask 28
0
100
200
300
400
500
600

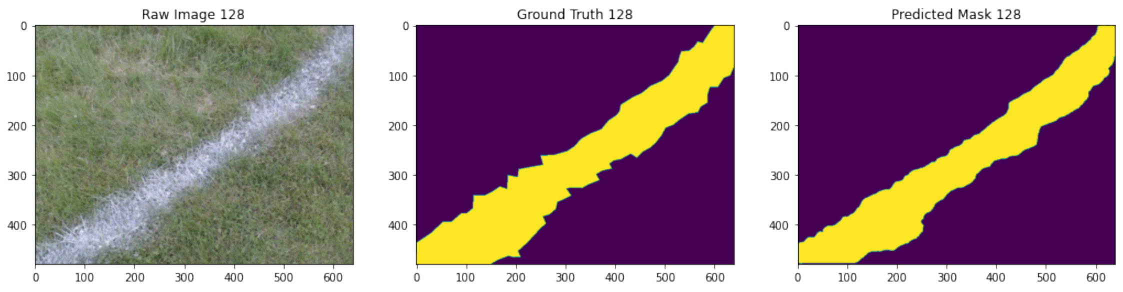

Raw Image 108
Ground Truth 108
Predicted Mask 108
0
100
200
300
400
500
600

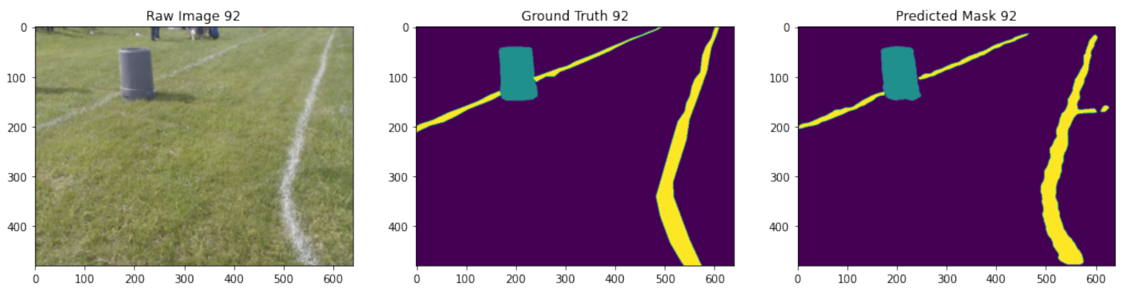

Raw Image 44
Ground Truth 44
Predicted Mask 44
0
100
200
300
400
500
600

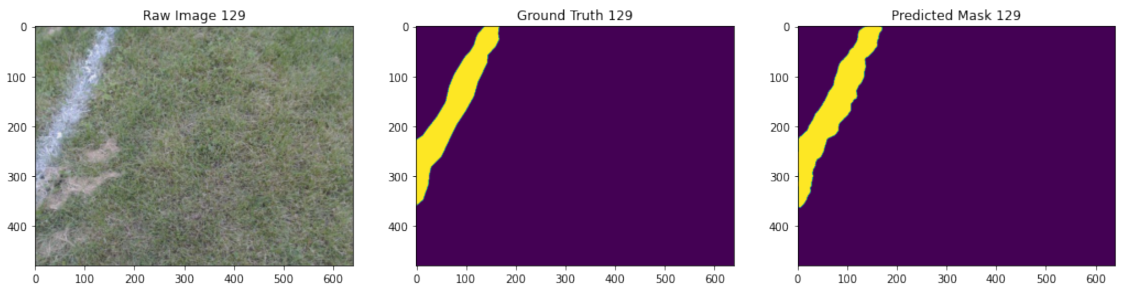

Raw Image 84
Ground Truth 84
Predicted Mask 84


Raw Image 30
Ground Truth 30
Predicted Mask 30


Raw Image 75
Ground Truth 75
Predicted Mask 75


Raw Image 128
Ground Truth 128
Predicted Mask 128





[35]:
```
#Display dictionary of sample test results
pred_results
```

[35]:
```
{'Ground Truth 108': <matplotlib.image.AxesImage at 0x7efd29149940>,
 'Ground Truth 128': <matplotlib.image.AxesImage at 0x7efd298e5dd8>,
 'Ground Truth 129': <matplotlib.image.AxesImage at 0x7efd2922bc88>,
 'Ground Truth 13': <matplotlib.image.AxesImage at 0x7efd28cc37f0>,
 'Ground Truth 134': <matplotlib.image.AxesImage at 0x7efd28e1c358>,
 'Ground Truth 141': <matplotlib.image.AxesImage at 0x7efd28c20ef0>,
 'Ground Truth 28': <matplotlib.image.AxesImage at 0x7efd2906c208>,
 'Ground Truth 30': <matplotlib.image.AxesImage at 0x7efd29996160>,
 'Ground Truth 44': <matplotlib.image.AxesImage at 0x7efd293910b8>,
 'Ground Truth 45': <matplotlib.image.AxesImage at 0x7efd28ed44a8>,
 'Ground Truth 62': <matplotlib.image.AxesImage at 0x7efd28da6be0>,
 'Ground Truth 66': <matplotlib.image.AxesImage at 0x7efd2cb7c278>,
 'Ground Truth 71': <matplotlib.image.AxesImage at 0x7efd2c6f9320>,
 'Ground Truth 75': <matplotlib.image.AxesImage at 0x7efd291ec6a0>,
 'Ground Truth 84': <matplotlib.image.AxesImage at 0x7efd29842fd0>,
 'Ground Truth 86': <matplotlib.image.AxesImage at 0x7efd7f127438>,
 'Ground Truth 92': <matplotlib.image.AxesImage at 0x7efd2977d550>,
 'Predicted Mask 108': <matplotlib.image.AxesImage at 0x7efd29120f60>,
 'Predicted Mask 128': <matplotlib.image.AxesImage at 0x7efd29aec438>,
 'Predicted Mask 129': <matplotlib.image.AxesImage at 0x7efd29ea32e8>,
 'Predicted Mask 13': <matplotlib.image.AxesImage at 0x7efd2978e780>,
 'Predicted Mask 134': <matplotlib.image.AxesImage at 0x7efd290b6978>,
 'Predicted Mask 141': <matplotlib.image.AxesImage at 0x7efd28c0f3c8>,
 'Predicted Mask 28': <matplotlib.image.AxesImage at 0x7efd29233828>,
 'Predicted Mask 30': <matplotlib.image.AxesImage at 0x7efd29467588>,
 'Predicted Mask 44': <matplotlib.image.AxesImage at 0x7efd28f036d8>,
 'Predicted Mask 45': <matplotlib.image.AxesImage at 0x7efd28c63ac8>,
 'Predicted Mask 62': <matplotlib.image.AxesImage at 0x7efd28e3c240>,
 'Predicted Mask 66': <matplotlib.image.AxesImage at 0x7efd289cd080>,
 'Predicted Mask 71': <matplotlib.image.AxesImage at 0x7efd2c7e67b8>,
 'Predicted Mask 75': <matplotlib.image.AxesImage at 0x7efd291dbcc0>,
 'Predicted Mask 84': <matplotlib.image.AxesImage at 0x7efd2986ce10>,
 'Predicted Mask 86': <matplotlib.image.AxesImage at 0x7efd290b90b8>,
 'Predicted Mask 92': <matplotlib.image.AxesImage at 0x7efd292cbb70>,
 'Raw Image 108': <matplotlib.image.AxesImage at 0x7efd29166320>,
 'Raw Image 128': <matplotlib.image.AxesImage at 0x7efd298dd160>,
 'Raw Image 129': <matplotlib.image.AxesImage at 0x7efd29280668>,
 'Raw Image 13': <matplotlib.image.AxesImage at 0x7efd28cfa2b0>,
 'Raw Image 134': <matplotlib.image.AxesImage at 0x7efd28e7ce48>,
 'Raw Image 141': <matplotlib.image.AxesImage at 0x7efd28b52e80>,
 'Raw Image 28': <matplotlib.image.AxesImage at 0x7efd28f7bba8>,
 'Raw Image 30': <matplotlib.image.AxesImage at 0x7efd29993908>,
 'Raw Image 44': <matplotlib.image.AxesImage at 0x7efd29a38a58>,
 'Raw Image 45': <matplotlib.image.AxesImage at 0x7efd28ce2e48>,
 'Raw Image 62': <matplotlib.image.AxesImage at 0x7efd28d6b5c0>,
 'Raw Image 66': <matplotlib.image.AxesImage at 0x7efd29f77b70>,
 'Raw Image 71': <matplotlib.image.AxesImage at 0x7efd29920cc0>,
 'Raw Image 75': <matplotlib.image.AxesImage at 0x7efd29da1080>,
 'Raw Image 84': <matplotlib.image.AxesImage at 0x7efd293291d0>,
 'Raw Image 86': <matplotlib.image.AxesImage at 0x7efd29ac2b00>,
 'Raw Image 92': <matplotlib.image.AxesImage at 0x7efd29c4def0>}
```

[0]:
```
#Pickle sample test results
f = open("pred_results.pkl","wb")
pickle.dump(pred_results,f)
f.close()
```