Visualizations

April 15, 2021

[1]:

```
# Drive mount for dataset and result save access
from google.colab import drive
drive.mount('/content/drive',force_remount=True)
%cd /content/drive/MyDrive/nimbronet_final/
```

```
Mounted at /content/drive
/content/drive/MyDrive/nimbronet_final
```

[2]:

```
import argparse
import cv2
import matplotlib.pyplot as plt
import numpy as np
import torch
from torchvision import transforms
from utils.dataloader import (
    blobDataset,
    SegDataset,
    blob_dataloader,
    segmentation_dataloader,
)
from utils.model import nimbrRoNet2
from utils.metrics import metrics
from utils.losses import losses
from torchsummary import summary
```

```
Using cache found in /root/.cache/torch/hub/pytorch_vision_v0.6.0
```

0.1 Dataloader and Model Initialization

[3]:

```
parser = argparse.ArgumentParser(description="Nimbronet training")
parser.add_argument(
    "-b", "--batch_size", default=1, type=int, help="mini-batch size (default:␣
 ↪8)"
)
parser.set_defaults(augment=True)
args = parser.parse_args(args=[])
```

```
blob_dir = '/content/drive/MyDrive/Nimbronet/data/blob/'
seg_dir =  '/content/drive/MyDrive/Nimbronet/data/segmentation/'

transfs = transforms.Compose(
        [
            transforms.Resize((480, 640)),
            transforms.ToTensor(),
        ]
    )

# Initializing the detection and segmentation dataloaders
_, _, testbdataloader, len_blob = blob_dataloader(blob_dir, transfs, args)
_, _, testsdataloader, len_seg = segmentation_dataloader(seg_dir, transfs, args)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Device used: ", device)

nnet2 = nimbrRoNet2()
nnet2 = nnet2.to(device)
nnet2.load_state_dict(torch.load('/content/drive/MyDrive/Nimbronet/
 ↪Final_model_300.pt')) #loading pretrained model weights

summary(nnet2,(3,480,640))
```

```
Device used:  cuda
```

```
Using cache found in /root/.cache/torch/hub/pytorch_vision_v0.6.0
```

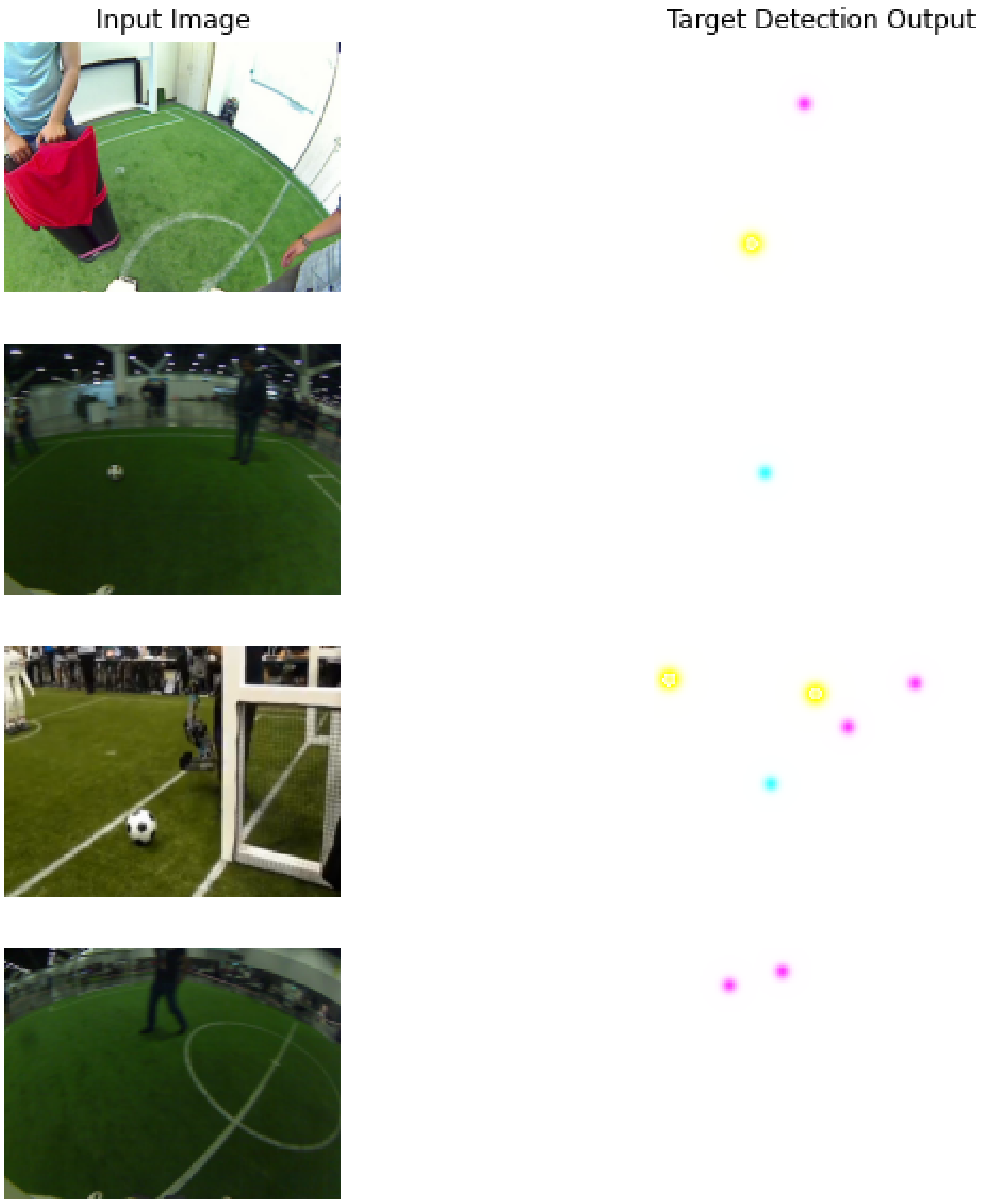
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 240, 320]           9,408
       BatchNorm2d-2         [-1, 64, 240, 320]             128
              ReLU-3         [-1, 64, 240, 320]               0
         MaxPool2d-4         [-1, 64, 120, 160]               0
            Conv2d-5         [-1, 64, 120, 160]          36,864
       BatchNorm2d-6         [-1, 64, 120, 160]             128
              ReLU-7         [-1, 64, 120, 160]               0
            Conv2d-8         [-1, 64, 120, 160]          36,864
       BatchNorm2d-9         [-1, 64, 120, 160]             128
             ReLU-10         [-1, 64, 120, 160]               0
       BasicBlock-11         [-1, 64, 120, 160]               0
           Conv2d-12         [-1, 64, 120, 160]          36,864
      BatchNorm2d-13         [-1, 64, 120, 160]             128
             ReLU-14         [-1, 64, 120, 160]               0
           Conv2d-15         [-1, 64, 120, 160]          36,864
      BatchNorm2d-16         [-1, 64, 120, 160]             128
             ReLU-17         [-1, 64, 120, 160]               0
       BasicBlock-18         [-1, 64, 120, 160]               0
           Conv2d-19          [-1, 128, 60, 80]          73,728
      BatchNorm2d-20          [-1, 128, 60, 80]             256
             ReLU-21          [-1, 128, 60, 80]               0
           Conv2d-22          [-1, 128, 60, 80]         147,456
      BatchNorm2d-23          [-1, 128, 60, 80]             256
           Conv2d-24          [-1, 128, 60, 80]           8,192
      BatchNorm2d-25          [-1, 128, 60, 80]             256
             ReLU-26          [-1, 128, 60, 80]               0
       BasicBlock-27          [-1, 128, 60, 80]               0
           Conv2d-28          [-1, 128, 60, 80]         147,456
      BatchNorm2d-29          [-1, 128, 60, 80]             256
             ReLU-30          [-1, 128, 60, 80]               0
           Conv2d-31          [-1, 128, 60, 80]         147,456
      BatchNorm2d-32          [-1, 128, 60, 80]             256
             ReLU-33          [-1, 128, 60, 80]               0
       BasicBlock-34          [-1, 128, 60, 80]               0
           Conv2d-35          [-1, 256, 30, 40]         294,912
      BatchNorm2d-36          [-1, 256, 30, 40]             512
             ReLU-37          [-1, 256, 30, 40]               0
           Conv2d-38          [-1, 256, 30, 40]         589,824
      BatchNorm2d-39          [-1, 256, 30, 40]             512
           Conv2d-40          [-1, 256, 30, 40]          32,768
      BatchNorm2d-41          [-1, 256, 30, 40]             512
             ReLU-42          [-1, 256, 30, 40]               0
       BasicBlock-43          [-1, 256, 30, 40]               0
           Conv2d-44          [-1, 256, 30, 40]         589,824
      BatchNorm2d-45          [-1, 256, 30, 40]             512
             ReLU-46          [-1, 256, 30, 40]               0
           Conv2d-47          [-1, 256, 30, 40]         589,824
      BatchNorm2d-48          [-1, 256, 30, 40]             512
             ReLU-49          [-1, 256, 30, 40]               0
       BasicBlock-50          [-1, 256, 30, 40]               0
           Conv2d-51          [-1, 512, 15, 20]       1,179,648
      BatchNorm2d-52          [-1, 512, 15, 20]           1,024
             ReLU-53          [-1, 512, 15, 20]               0
           Conv2d-54          [-1, 512, 15, 20]       2,359,296
      BatchNorm2d-55          [-1, 512, 15, 20]           1,024
           Conv2d-56          [-1, 512, 15, 20]         131,072
      BatchNorm2d-57          [-1, 512, 15, 20]           1,024
             ReLU-58          [-1, 512, 15, 20]               0
       BasicBlock-59          [-1, 512, 15, 20]               0
           Conv2d-60          [-1, 512, 15, 20]       2,359,296
      BatchNorm2d-61          [-1, 512, 15, 20]           1,024
             ReLU-62          [-1, 512, 15, 20]               0
           Conv2d-63          [-1, 512, 15, 20]       2,359,296
      BatchNorm2d-64          [-1, 512, 15, 20]           1,024
             ReLU-65          [-1, 512, 15, 20]               0
       BasicBlock-66          [-1, 512, 15, 20]               0
           Conv2d-67        [-1, 128, 120, 160]           8,192
           Conv2d-68          [-1, 256, 60, 80]          32,768
           Conv2d-69          [-1, 256, 30, 40]          65,536
  ConvTranspose2d-70          [-1, 256, 30, 40]         524,288
             ReLU-71          [-1, 512, 30, 40]               0
      BatchNorm2d-72          [-1, 512, 30, 40]           1,024
  ConvTranspose2d-73          [-1, 256, 60, 80]         524,288
             ReLU-74          [-1, 512, 60, 80]               0
      BatchNorm2d-75          [-1, 512, 60, 80]           1,024
  ConvTranspose2d-76        [-1, 128, 120, 160]         262,144
             ReLU-77        [-1, 256, 120, 160]               0
      BatchNorm2d-78        [-1, 256, 120, 160]             512
           Conv2d-79          [-1, 3, 120, 160]             768
           Conv2d-80          [-1, 3, 120, 160]             768
================================================================
Total params: 12,597,824
Trainable params: 12,597,824
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 3.52
Forward/backward pass size (MB): 568.07
Params size (MB): 48.06
Estimated Total Size (MB): 619.64
----------------------------------------------------------------
```

0.2 Detection dataset samples

[25]:

```
f, axarr = plt.subplots(4,2, figsize=(10., 10.))
axarr[0,0].title.set_text('Input Image')
axarr[0,1].title.set_text('Target Detection Output')

for idx, data in enumerate(testbdataloader):
  image, target = data[0][0], data[1][0]

  image = transforms.ToPILImage()(image)
  target = transforms.ToPILImage()(target)

  axarr[idx,0].imshow(image)
  axarr[idx,0].axis('off')

  axarr[idx,1].imshow(target)
  axarr[idx,1].axis('off')

  if idx == 3:
    break
```

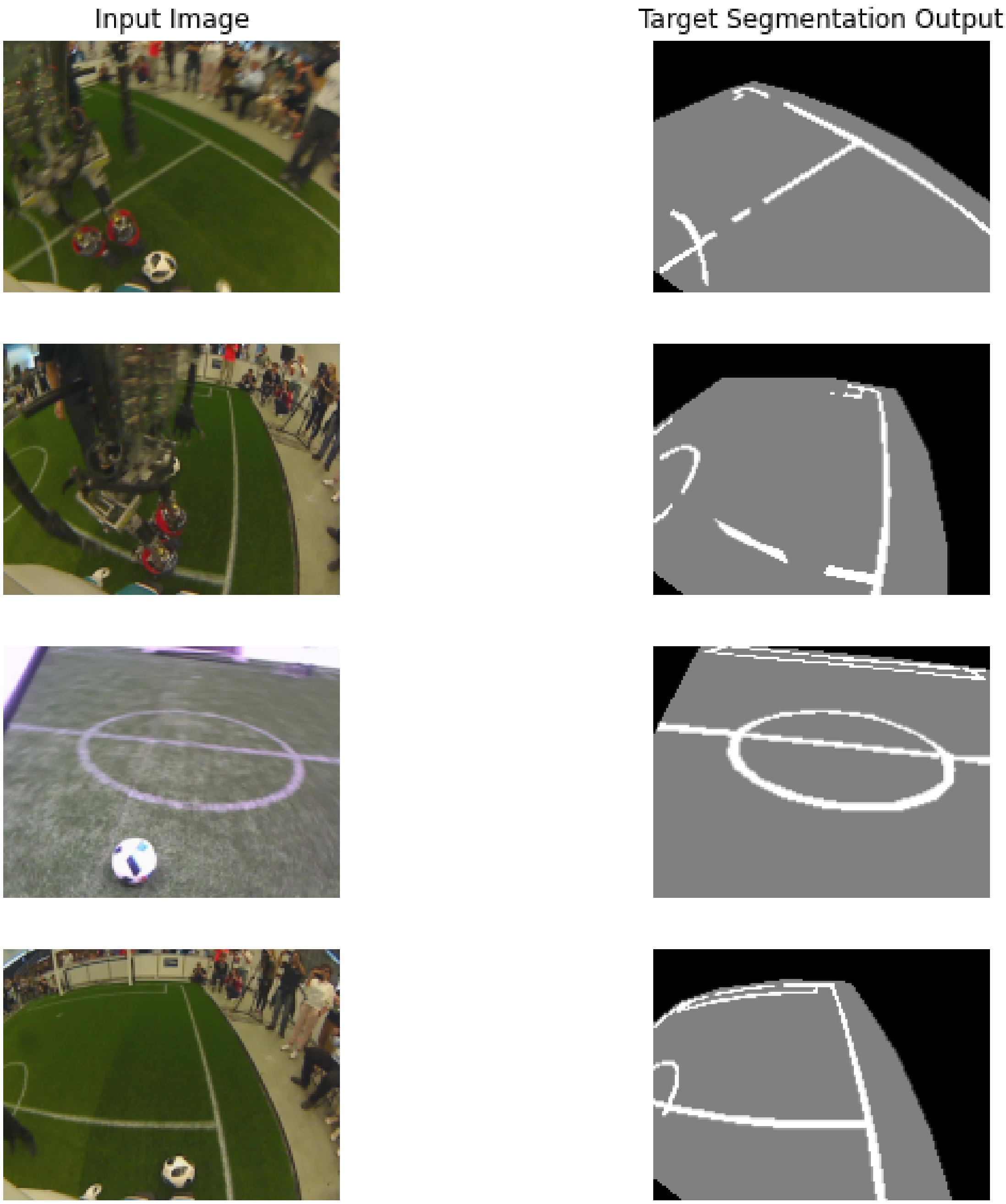
```
plt.show()
```



0.3 Segmentation dataset samples

[24]:

```
f, axarr = plt.subplots(4,2, figsize=(10., 10.))
axarr[0,0].title.set_text('Input Image')
axarr[0,1].title.set_text('Target Segmentation Output')

for idx, data in enumerate(testsdataloader):
  image, target = data[0][0], data[1][0]

  image = transforms.ToPILImage()(image)
  # target = transforms.ToPILImage()(target)

  axarr[idx,0].imshow(image)
  axarr[idx,0].axis('off')

  axarr[idx,1].imshow(target,cmap='gray')
  axarr[idx,1].axis('off')

  if idx == 3:
    break
plt.show()
```

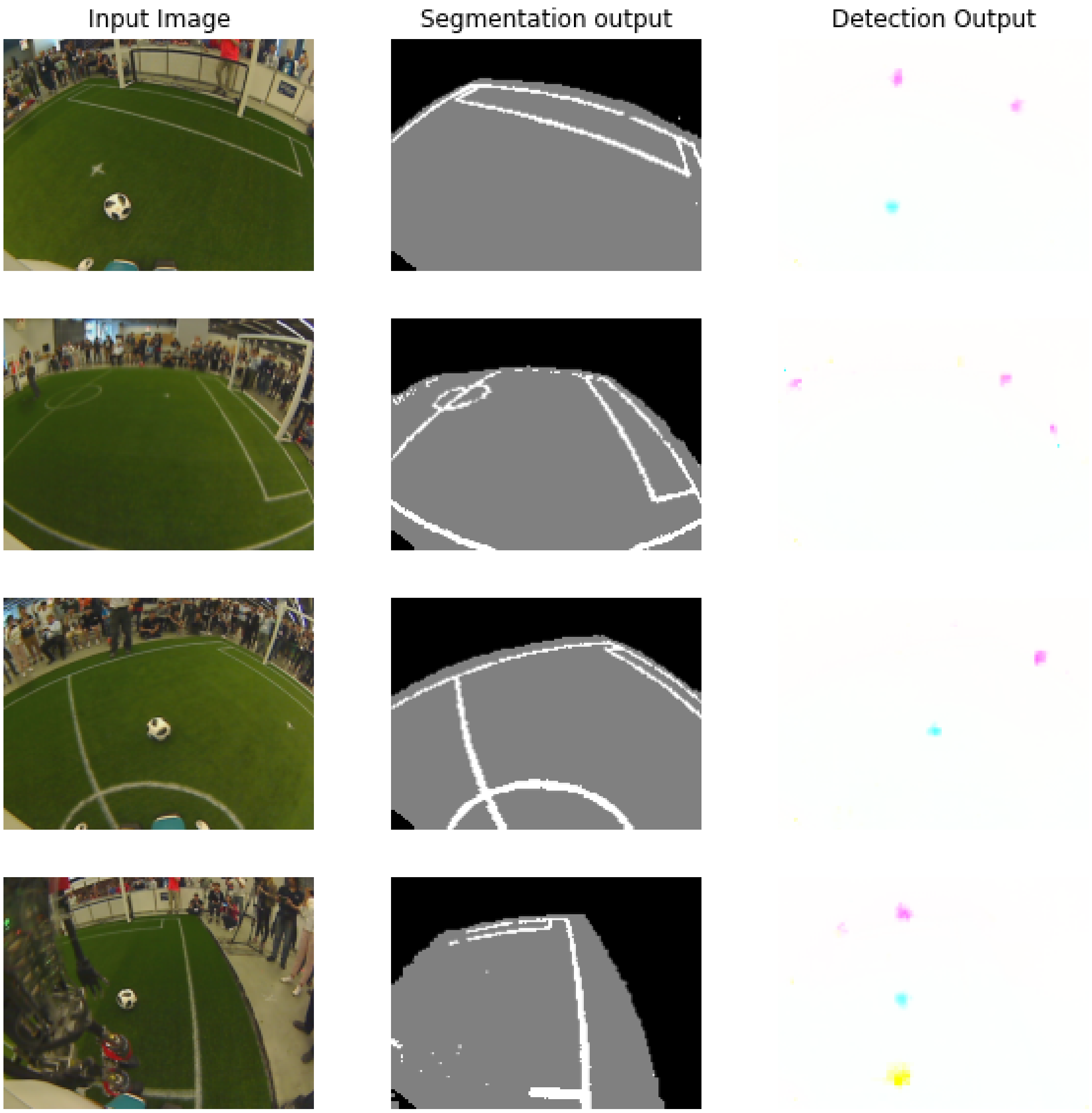


0.4 Network output visualization

[18]:

```python
f, axarr = plt.subplots(4,3, figsize=(10., 10.))
axarr[0,0].title.set_text('Input Image')
axarr[0,1].title.set_text('Segmentation output')
axarr[0,2].title.set_text('Detection Output')
```

```
for idx, data in enumerate(testsdataloader):
  image, target = data[0], data[1]
  seg_pred,blob_pred = nnet2(image.to(device))

  image = transforms.ToPILImage()(image[0])

  blob_pred = torch.transpose(blob_pred,1,2)
  blob_pred = torch.transpose(blob_pred,2,3)
  blob_pred = blob_pred.squeeze().detach().cpu().numpy()

  seg_pred = torch.argmax(seg_pred,1,keepdim=True).squeeze().detach().cpu()


  axarr[idx,0].imshow(image)
  axarr[idx,0].axis('off')

  axarr[idx,1].imshow(seg_pred, cmap='gray')
  axarr[idx,1].axis('off')

  axarr[idx,2].imshow(np.clip(blob_pred,0,1))
  axarr[idx,2].axis('off')

  if idx == 3:
    break
plt.savefig('results_all.pdf',bbox_inches='tight')
plt.show()
```


Input Image
Segmentation output
Detection Output