3D Convolution 2D Deconvolution Network

August 7, 2019

```
In [1]: import os
        import numpy as np
        import pandas as pd
        from matplotlib import pyplot as plt
        from PIL import Image

In [2]: import torch
        import torch.nn as nn
        import torch.optim as optim
        import torch.nn.functional as F

        import torchvision
        import torchvision.transforms as transforms

        import torch.utils as utils
        from torch import autograd

In [3]: torch.set_printoptions(linewidth=30)
        torch.set_grad_enabled(True)

        device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

In [4]: # Hyper Parameters
        num_epochs = 100
        batch_size = 20
        learning_rate = 0.003

In [5]: data_dir = './data/'
        raw_dir = os.path.join(data_dir, 'raw/')
        img_dir = os.path.join(data_dir, 'image/')

In [6]: transform = transforms.Compose([
            transforms.ToTensor(),
            transforms.Normalize(mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5))
        ])

        list_data = []
        list_label = []
        for filename in os.listdir(raw_dir):
            isovalue = int(filename.split('_')[1].strip('.raw'))

            f = np.fromfile(raw_dir + filename, dtype='uint8')
            f = (f.astype('float') - isovalue / 2) / 255
            raw_img = torch.Tensor(f).reshape([1, 64, 64, 64])
            list_data.append(raw_img)

            if os.path.isfile(img_dir + filename.replace('.raw', '.png')):
                item = filename.replace('.raw', '.png')
                im = transform(Image.open(img_dir + item))
                list_label.append(im)

        tensor_data = torch.stack(list_data)
        tensor_label = torch.stack(list_label)

        dataset = utils.data.TensorDataset(tensor_data, tensor_label)
```

```
In [7]: tensor_data.shape, tensor_label.shape
```

```
Out[7]: (torch.Size([4800, 1, 64, 64, 64]), torch.Size([4800, 3, 64, 64]))
```

```
In [9]: sample_size = 3200

        batch_size = 16
        val_split = 0.2
        shuffle_dataset = True
        random_seed = 42

        indices = list(range(sample_size))
        split = int(np.floor(val_split * sample_size))

        if shuffle_dataset:
            np.random.seed(random_seed)
            np.random.shuffle(indices)

        train_indices, valid_indices = indices[split:], indices[:split]
```

```
In [10]: train_sampler = utils.data.SubsetRandomSampler(train_indices)
         valid_sampler = utils.data.SubsetRandomSampler(valid_indices)
```

```
In [11]: train_loader = torch.utils.data.DataLoader(dataset,
                                                    batch_size=batch_size,
                                                    sampler=train_sampler)
         valid_loader = torch.utils.data.DataLoader(dataset,
                                                    batch_size=batch_size,
                                                    sampler=valid_sampler)
```

```
In [12]: def get_correct_number(preds, labels):
             return preds.argmax(dim=1).eq(labels).sum().item()
```

```
In [13]: class Network(nn.Module):
             def __init__(self):
                 super(Network, self).__init__()

                 # Convolution 1
                 self.conv1 = nn.Conv3d(1, 16, kernel_size=3, padding=1)
                 nn.init.xavier_uniform(self.conv1.weight)
                 self.max1 = nn.MaxPool3d(kernel_size=(2, 2, 2),
                                          stride=(2, 2, 2),
                                          return_indices=True)

                 # Convolution 2
                 self.conv2 = nn.Conv3d(16, 32, kernel_size=3, padding=1)
                 nn.init.xavier_uniform(self.conv2.weight)
                 self.max2 = nn.MaxPool3d(kernel_size=(2, 2, 2),
                                          stride=(2, 2, 2),
                                          return_indices=True)

                 # Convolution 3
                 self.conv3 = nn.Conv3d(32, 64, kernel_size=3, padding=1)
                 nn.init.xavier_uniform(self.conv3.weight)
                 self.max3 = nn.MaxPool3d(kernel_size=(2, 2, 2),
                                          stride=(2, 2, 2),
                                          return_indices=True)

                 # Convolution 4
                 self.conv4 = nn.Conv3d(64, 128, kernel_size=3, padding=1)
                 nn.init.xavier_uniform(self.conv4.weight)
                 self.max4 = nn.MaxPool3d(kernel_size=(2, 2, 2),
                                          stride=(2, 2, 2),
                                          return_indices=True)

                 # Fully Connected / Dense Layer 1
                 self.fc1 = nn.Linear(128 * 4 * 4 * 4, 128 * 4 * 4)

                 # De Convolution 1
                 self.maxUn1 = torch.nn.MaxUnpool2d(2, stride=2)
                 self.deconv1 = torch.nn.ConvTranspose2d(128, 64, 3, padding=1)

                 # De Convolution 2
                 self.maxUn2 = torch.nn.MaxUnpool2d(2, stride=2)
                 self.deconv2 = torch.nn.ConvTranspose2d(64, 32, 3, padding=1)

                 # De Convolution 3
                 self.maxUn3 = torch.nn.MaxUnpool2d(2, stride=2)
```

```
        self.deconv3 = torch.nn.ConvTranspose2d(32, 16, 3, padding=1)

        # De Convolution 4
        self.maxUn4 = torch.nn.MaxUnpool2d(2, stride=2)
        self.deconv4 = torch.nn.ConvTranspose2d(16, 3, 3, padding=1)

    def forward(self, data):
        out = F.leaky_relu(self.conv1(data))
        size1 = out[:, :, 0, :, :].size()
        out, indices1 = self.max1(out)

        out = F.leaky_relu(self.conv2(out))
        size2 = out[:, :, 0, :, :].size()
        out, indices2 = self.max2(out)

        out = F.leaky_relu(self.conv3(out))
        size3 = out[:, :, 0, :, :].size()
        out, indices3 = self.max3(out)

        out = F.leaky_relu(self.conv4(out))
        size4 = out[:, :, 0, :, :].size()
        out, indices4 = self.max4(out)

        out = out.view(out.size(0), -1)
        out = F.leaky_relu(self.fc1(out))
        out = out.view(16, 128, 4, 4)

        indices1 = flatten_indices(indices1)
        indices2 = flatten_indices(indices2)
        indices3 = flatten_indices(indices3)
        indices4 = flatten_indices(indices4)

        out = self.maxUn1(out, indices4, output_size=size4)
        out = F.leaky_relu(self.deconv1(out))

        out = self.maxUn2(out, indices3, output_size=size3)
        out = F.leaky_relu(self.deconv2(out))

        out = self.maxUn3(out, indices2, output_size=size2)
        out = F.leaky_relu(self.deconv3(out))

        out = self.maxUn4(out, indices1, output_size=size1)
        out = F.leaky_relu(self.deconv4(out))

        return out
```

```
In [14]: def flatten_indices(indices):
             indices = indices[:, :, 0, :, :]
             max = indices.size()[2] * indices.size()[3] * 4
             return (indices.int() - ((indices >= max).int() * max)).long()
```

```
In [15]: import torch.optim as optim
         import pytorch_msssim
```

```
In [16]: with torch.cuda.device(0):
             network = Network()
             optim = optim.Adam(network.parameters(), lr=0.001)

             for epoch in range(5):
                 total_loss = 0
                 total_correct = 0

                 for volume, image in train_loader:
                     pred = network(volume)
                     loss = pytorch_msssim.msssim(pred, image, normalize=True)

                     network.zero_grad()
                     loss.backward()
                     optim.step()

                     total_loss += loss.item()

                 print("epoch:", epoch, "total_loss:", total_loss)
```

```
/opt/anaconda/lib/python3.7/site-packages/ipykernel_launcher.py:7: UserWarning: nn.init.xavier_
  import sys
/opt/anaconda/lib/python3.7/site-packages/ipykernel_launcher.py:14: UserWarning: nn.init.xavier

/opt/anaconda/lib/python3.7/site-packages/ipykernel_launcher.py:21: UserWarning: nn.init.xavier
/opt/anaconda/lib/python3.7/site-packages/ipykernel_launcher.py:28: UserWarning: nn.init.xavier


epoch: 0 total_loss: 29.378464579582214
epoch: 1 total_loss: 18.09668856859207
epoch: 2 total_loss: 14.076175168156624
epoch: 3 total_loss: 12.400116141885519
epoch: 4 total_loss: 10.590228825807571
```