Assignment 2 Report

Chenqiu Zhao(zhao.chenqiu@ualberta.ca)

In this assignment, three python source files are proposed corresponding to question 1 , question 2 and question 3 respectively. The mathematical proof of question 4 is proposed in the last section.

Question 1

To replace the sum of squared difference (SSD) by mutual information neural estimation (MINE), two key functions and a key class are implemented, please check "**question1.py**" for more details. Mine network is implemented in a class, in which three linear layers with the size of 2×10, 10×10 and 10×1 are contained. The source code is shown as below:

```
class Mine(nn.Module):
    def __init__(self, input_size=2, hidden_size=100):
        super().__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.fc2 = nn.Linear(hidden_size, hidden_size)
        self.fc3 = nn.Linear(hidden_size, 1)
        nn.init.normal_(self.fc1.weight,std=0.02)
        nn.init.constant_(self.fc1.bias, 0)
        nn.init.normal_(self.fc2.weight,std=0.02)
        nn.init.constant_(self.fc2.bias, 0)
        nn.init.normal_(self.fc3.weight,std=0.02)
        nn.init.constant_(self.fc3.bias, 0)

    def forward(self, input):
        output = F.elu(self.fc1(input))
        output = F.elu(self.fc2(output))
        output = self.fc3(output)
        return output
```

Since there is a sampling procedure in the Mine loss, the function **sample_batch** is implemented as below:

```
def sample_batch(data, batch_size=100, sample_mode='joint'):
    if sample_mode == 'joint':
        index = np.random.choice(range(data.shape[0]), size=batch_size, replace=False)
        batch = data[index]
    else:
        joint_index = np.random.choice(range(data.shape[0]), size=batch_size, replace=False)
        marginal_index = np.random.choice(range(data.shape[0]), size=batch_size, replace=False)

        batch = torch.cat((data[joint_index][:,0].reshape(-1,1),
                                        data[marginal_index][:,1].reshape(-1,1)), 1)
    return batch
```

Finally, the function **MINE_loss** is implemented as below:

```
def MINE_loss(J_w, I, batch_size, important_ind):
    height, width = I.shape

    data = torch.cat(( J_w.view([height * width])[important_ind].unsqueeze(1), \
                       I.view(  [height * width])[important_ind].unsqueeze(1)), 1)
```

```
    joint, marginal = sample_batch(data, batch_size), sample_batch(data, batch_size, 'marginal')

    t = mine_net(joint)
    et = torch.exp(mine_net(marginal))

    mine_loss = -(torch.mean(t) - torch.log(torch.mean(et)))

    return mine_loss
```

A screenshot of program output is demonstrated as below:

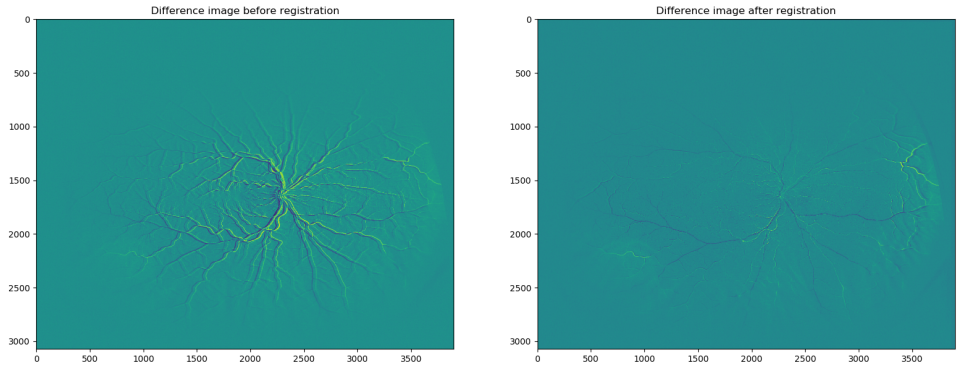

Figure 1: The demonstration of image registration based on MINE loss.

Question 2

We extend the function **MatrixExp** into a class, in order to implement our own back propagation rule. The source code is shown as follows:

```
class MatrixExp(Function):

    @staticmethod
    def forward(ctx, C):
        A = torch.eye(3).to(device)

        H = A
        for i in torch.arange(1, 100):
            A = torch.mm(A/i, C)
            H = H + A

        ctx.save_for_backward(H)
        return H

    @staticmethod
    def backward(ctx, grad_output):
        result, = ctx.saved_tensors

        return torch.mm(grad_output, result)
```

You can also check the file "**question2.py**" for more details. The program is running much faster than the old one. In addition, the iteration number of Taylor series is increased to 100 to demonstrate that our own backpropagation rule makes the program faster. We proposed a comparison between the program with class MatrixExp and the old program to demonstrate the efficiency, which is shown as Fig. 2. As we can see, the old program takes 112 seconds but the program with our own backpropagation only takes 46 seconds.

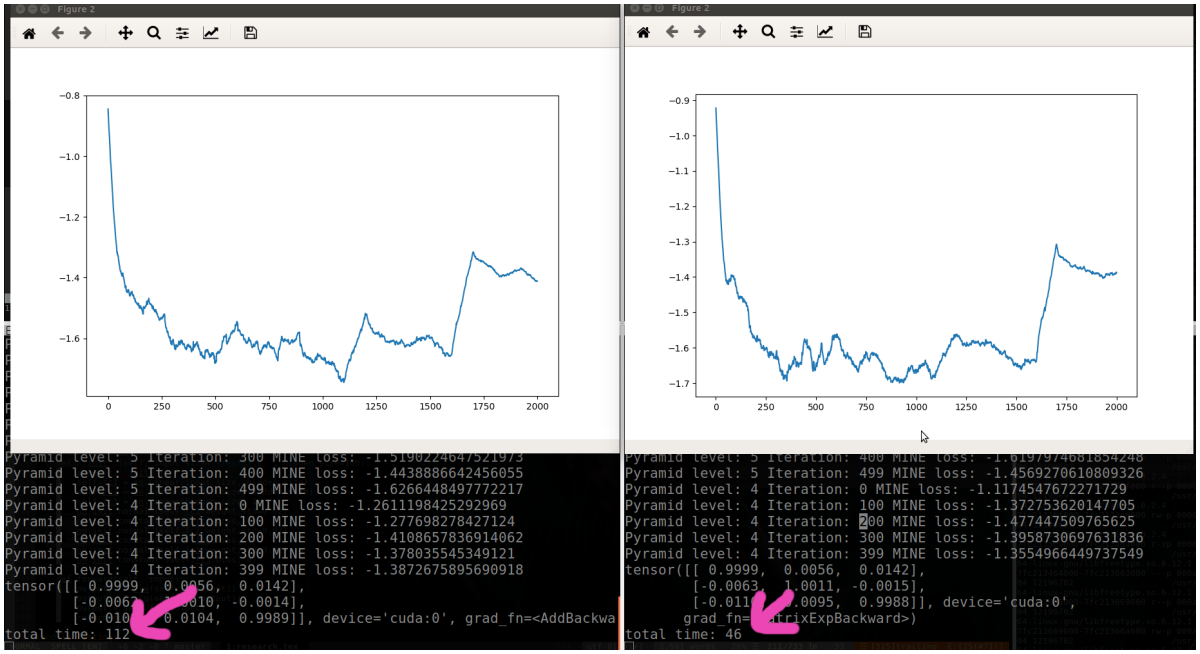

Figure 2: The comparison between problem with our own derivative rule and the original one. The running time of program with our own backpropagation rule is shown in the right part.

Question 3

To extend our program to color image, the channel of MINE loss is increased to 6. The **sample_batch** function is modified for data with 6 channel, and the function **PerspectiveTransform** is also extended to for transforming color image. Please check "**question3.py**" for more details. The source codes of these key functions are shown as follows:

```
class Mine(nn.Module):
    def __init__(self, input_size=6, hidden_size=100):
        super().__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.fc2 = nn.Linear(hidden_size, hidden_size)
        self.fc3 = nn.Linear(hidden_size, 1)
        nn.init.normal_(self.fc1.weight,std=0.02)
        nn.init.constant_(self.fc1.bias, 0)
        nn.init.normal_(self.fc2.weight,std=0.02)
        nn.init.constant_(self.fc2.bias, 0)
        nn.init.normal_(self.fc3.weight,std=0.02)
        nn.init.constant_(self.fc3.bias, 0)

    def forward(self, input):
        output = F.elu(self.fc1(input))
        output = F.elu(self.fc2(output))
        output = self.fc3(output)
        return output
```

```
def sample_batch(data, batch_size=100, sample_mode='joint'):
    if sample_mode == 'joint':
        index = np.random.choice(range(data.shape[0]), size=batch_size, replace=False)
        batch = data[index]

    else:
        joint_index = np.random.choice(range(data.shape[0]), size=batch_size, replace=False)
        marginal_index = np.random.choice(range(data.shape[0]), size=batch_size, replace=False)
```

```
        batch_joint = data[joint_index, 0:3]
        batch_marginal = data[marginal_index, 3:6]

        batch = torch.cat( (batch_joint, batch_marginal) , 1)
    return batch
```

```
def PerspectiveTransform(I, H, xv, yv):

    height, width, byte = I.shape

    xvt = (xv*H[0,0]+yv*H[0,1]+H[0,2])/(xv*H[2,0]+yv*H[2,1]+H[2,2])
    yvt = (xv*H[1,0]+yv*H[1,1]+H[1,2])/(xv*H[2,0]+yv*H[2,1]+H[2,2])

    J0 = F.grid_sample(I[:, :,
     0].view(1,1,height,width),torch.stack([xvt,yvt],2).unsqueeze(0)).squeeze()
    J1 = F.grid_sample(I[:, :,
     1].view(1,1,height,width),torch.stack([xvt,yvt],2).unsqueeze(0)).squeeze()
    J2 = F.grid_sample(I[:, :,
     2].view(1,1,height,width),torch.stack([xvt,yvt],2).unsqueeze(0)).squeeze()

    J0 = J0.unsqueeze(2)
    J1 = J1.unsqueeze(2)
    J2 = J2.unsqueeze(2)

    J = torch.cat((J0, J1, J2), 2)

    return J
```

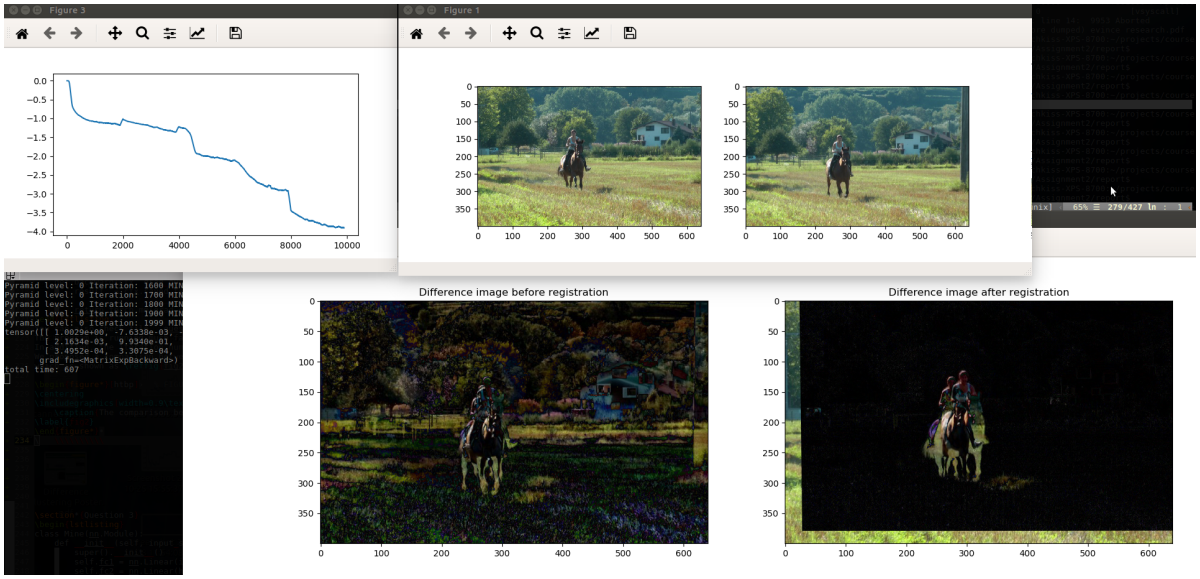

Figure 3: The demonstration of color image registration based on MINE loss.

Question 4

$$J(f) = \int f(x,z)P_{X,Z}(x,z)dxdz - ln(\int e^{f(x,z)}P_X(x)P_Z(z)dxdz) \tag{1}$$

$$Let \quad f(x,z) = \bar{f}(x,z) + \varepsilon\eta, \quad \eta = g(x,z) \tag{2}$$

$$\begin{aligned} G(\varepsilon) &= J(\bar{f}(x,z) + \varepsilon\eta) \\ &= \int (\bar{f}(x,z) + \varepsilon\eta)P_{X,Z}(x,z)dxdz - ln(\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz) \end{aligned} \tag{3}$$

$$\begin{aligned} \frac{dG(\varepsilon)}{d\varepsilon} &= \frac{d\left(\int (\bar{f}(x,z) + \varepsilon\eta)P_{X,Z}(x,z)dxdz - ln(\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz)\right)}{d\varepsilon} \\ &= \frac{d\left(\int (\bar{f}(x,z) + \varepsilon\eta)P_{X,Z}(x,z)dxdz\right)}{d\varepsilon} - \frac{d\left(ln(\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz)\right)}{d\varepsilon} \end{aligned} \tag{4}$$

According to the Leibniz integral rule:

$$\frac{d}{dx}\left(\int_a^b f(x,t)dt\right) = \int_a^b \frac{\partial}{\partial x}f(x,t)dt \tag{5}$$

we get:

$$\begin{aligned} &\frac{d\left(\int (\bar{f}(x,z) + \varepsilon\eta)P_{X,Z}(x,z)dxdz\right)}{d\varepsilon} \\ =&\int \frac{\partial\left((\bar{f}(x,z) + \varepsilon\eta)P_{X,Z}(x,z)dxdz\right)}{\partial\varepsilon} \\ =&\int \frac{\partial\left(\bar{f}(x,z)P_{X,Z}(x,z)dxdz\right)}{\partial\varepsilon} + \int \frac{\partial\left((\varepsilon\eta)P_{X,Z}(x,z)dxdz\right)}{\partial\varepsilon} \\ =&0 + \int \frac{\partial(\varepsilon\eta)}{\partial\varepsilon}P_{X,Z}(x,z)dxdz \\ =&\int \eta P_{X,Z}(x,z)dxdz \end{aligned} \tag{6}$$

$$
\begin{aligned}
&\frac{d\left(ln(\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz)\right)}{d\varepsilon} \\
=&\frac{1}{\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz}\cdot\frac{d\left(\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz\right)}{d\varepsilon} \\
=&\frac{1}{\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz}\cdot\int\frac{\partial\left(e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz\right)}{\partial\varepsilon} \\
=&\frac{1}{\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz}\cdot\int\frac{\partial e^{\bar{f}(x,z)+\varepsilon\eta}}{\partial\varepsilon}P_X(x)P_Z(z)dxdz \\
=&\frac{1}{\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz}\cdot\int e^{\bar{f}(x,z)+\varepsilon\eta}\frac{\partial(\bar{f}(x,z)+\varepsilon\eta)}{\partial\varepsilon}P_X(x)P_Z(z)dxdz \\
=&\frac{1}{\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz}\cdot\int e^{\bar{f}(x,z)+\varepsilon\eta}\eta P_X(x)P_Z(z)dxdz \\
=&\frac{\int e^{\bar{f}(x,z)+\varepsilon\eta}\eta P_X(x)P_Z(z)dxdz}{\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz}
\end{aligned}
\tag{7}
$$

Then, the Eqn. 4 becomes:

$$
\begin{aligned}
\frac{dG(\varepsilon)}{d\varepsilon}&=\frac{d\left(\int(\bar{f}(x,z)+\varepsilon\eta)P_{X,Z}(x,z)dxdz\right)}{d\varepsilon}-\frac{d\left(ln(\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz)\right)}{d\varepsilon} \\
&=\int\eta P_{X,Z}(x,z)dxdz-\frac{\int e^{\bar{f}(x,z)+\varepsilon\eta}\eta P_X(x)P_Z(z)dxdz}{\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz}
\end{aligned}
\tag{8}
$$

In order to achieve the maximum value, we need $\lim_{\varepsilon\to 0}\frac{dG(\varepsilon)}{d\varepsilon}=0$. Then we get:

$$
\begin{aligned}
&\lim_{\varepsilon\to 0}\frac{dG(\varepsilon)}{d\varepsilon}=\lim_{\varepsilon\to 0}\left(\int\eta P_{X,Z}(x,z)dxdz-\frac{\int e^{\bar{f}(x,z)+\varepsilon\eta}\eta P_X(x)P_Z(z)dxdz}{\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz}\right)=0 \\
\Rightarrow&\int\eta P_{X,Z}(x,z)dxdz-\lim_{\varepsilon\to 0}\left(\frac{\int e^{\bar{f}(x,z)+\varepsilon\eta}\eta P_X(x)P_Z(z)dxdz}{\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz}\right)=0 \\
\Rightarrow&\int\eta P_{X,Z}(x,z)dxdz=\lim_{\varepsilon\to 0}\left(\frac{\int e^{\bar{f}(x,z)+\varepsilon\eta}\eta P_X(x)P_Z(z)dxdz}{\int e^{\bar{f}(x,z)+\varepsilon\eta}P_X(x)P_Z(z)dxdz}\right) \\
\Rightarrow&\int\eta P_{X,Z}(x,z)dxdz=\frac{\int e^{\bar{f}(x,z)+0\cdot\eta}\eta P_X(x)P_Z(z)dxdz}{\int e^{\bar{f}(x,z)+0\cdot\eta}P_X(x)P_Z(z)dxdz} \\
\Rightarrow&\int\eta P_{X,Z}(x,z)dxdz=\frac{\int e^{\bar{f}(x,z)}\eta P_X(x)P_Z(z)dxdz}{\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz} \\
\Rightarrow&\int\eta\left(P_{X,Z}(x,z)\right)dxdz=\int\eta\left(\frac{e^{\bar{f}(x,z)}P_X(x)P_Z(z)}{\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz}\right)dxdz\quad\because\eta\neq 0 \\
\Rightarrow&P_{X,Z}(x,z)=\frac{e^{\bar{f}(x,z)}P_X(x)P_Z(z)}{\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz}
\end{aligned}
\tag{9}
$$

With the Eqn. 9, we have:

$$
\begin{aligned}
P_{X,Z}(x,z) &= \frac{e^{\bar{f}(x,z)}P_X(x)P_Z(z)}{\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz} \\
\Rightarrow e^{\bar{f}(x,z)} &= \frac{P_{X,Z}(x,z)\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz}{P_X(x)P_Z(z)} \\
\Rightarrow \bar{f}(x,z) &= ln\left(\frac{P_{X,Z}(x,z)\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz}{P_X(x)P_Z(z)}\right) \\
\Rightarrow \bar{f}(x,z) &= ln\left(\frac{P_{X,Z}(x,z)\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz}{P_X(x)P_Z(z)}\right) \\
\Rightarrow \bar{f}(x,z) &= ln\left(\frac{P_{X,Z}(x,z)}{P_X(x)P_Z(z)}\right) + ln\left(\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz\right)
\end{aligned}
\tag{10}
$$

Then, we plug $\bar{f}(x,z)$ into $J(f)$:

$$
\begin{aligned}
J(\bar{f}(x,z)) &= \int \bar{f}(x,z)P_{X,Z}(x,z)dxdz - ln\left(\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz\right) \\
&= \int \left[ln\left(\frac{P_{X,Z}(x,z)}{P_X(x)P_Z(z)}\right) + ln\left(\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz\right)\right]P_{X,Z}(x,z)dxdz \\
&\quad - ln\left(\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz\right) \\
&= \int ln\left(\frac{P_{X,Z}(x,z)}{P_X(x)P_Z(z)}\right)P_{X,Z}(x,z)dxdz + \int ln\left(\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz\right)P_{X,Z}(x,z)dxdz \\
&\quad - ln\left(\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz\right) \\
&= \int ln\left(\frac{P_{X,Z}(x,z)}{P_X(x)P_Z(z)}\right)P_{X,Z}(x,z)dxdz + ln\left(\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz\right)\int P_{X,Z}(x,z)dxdz \\
&\quad - ln\left(\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz\right) \quad \because \int P_{X,Z}(x,z)dxdz = 1 \\
&= \int ln\left(\frac{P_{X,Z}(x,z)}{P_X(x)P_Z(z)}\right)P_{X,Z}(x,z)dxdz + ln\left(\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz\right)\cdot 1 \\
&\quad - ln\left(\int e^{\bar{f}(x,z)}P_X(x)P_Z(z)dxdz\right) \\
&= \int ln\left(\frac{P_{X,Z}(x,z)}{P_X(x)P_Z(z)}\right)P_{X,Z}(x,z)dxdz \\
\Rightarrow J(\bar{f}(x,z)) &= \int ln\left(\frac{P_{X,Z}(x,z)}{P_X(x)P_Z(z)}\right)P_{X,Z}(x,z)dxdz
\end{aligned}
\tag{11}
$$