IRVISFTrainModel

February 28, 2023

[ ]:
```
##Libraries
from __future__ import print_function
import math
import time
import sys
import os
import random
import glob
import cv2
from glob import glob
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
from PIL import Image
import PIL.ImageOps
from sklearn.metrics import confusion_matrix
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torch.autograd import Variable
import torchvision
import torchvision.datasets as datasets
import torchvision.models as models
import torchvision.transforms as transforms
import torchvision.utils
from torch.utils.data import DataLoader, Dataset

import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
DATA_DIR = "data/"
trn_dir = f'{DATA_DIR}/train'
tst_dir = f'{DATA_DIR}/test'
#parameters
sz = 64
batch_size = 16
```

[3]:
```
os.listdir(DATA_DIR)
```

[3]:
```
['train', 'test']
```

[5]:
```
trn_fnames = glob(f'{trn_dir}/*/*.jpg')
trn_fnames[:3]
```

[5]:
```
['data//train/visible/040072.jpg',
 'data//train/visible/250412.jpg',
 'data//train/visible/140485.jpg']
```

[9]:
```
img = plt.imread(trn_fnames[1])
plt.imshow(img,cmap='brg');
plt.axis('off')
```

[9]:
```
(-0.5, 1279.5, 1023.5, -0.5)
```

[10]:
```
train_ds = datasets.ImageFolder(trn_dir)

tfms = transforms.Compose([
    transforms.Resize((sz, sz//2)),  # PIL Image
#     transforms.Grayscale(),
    transforms.ToTensor(),        # Tensor
```

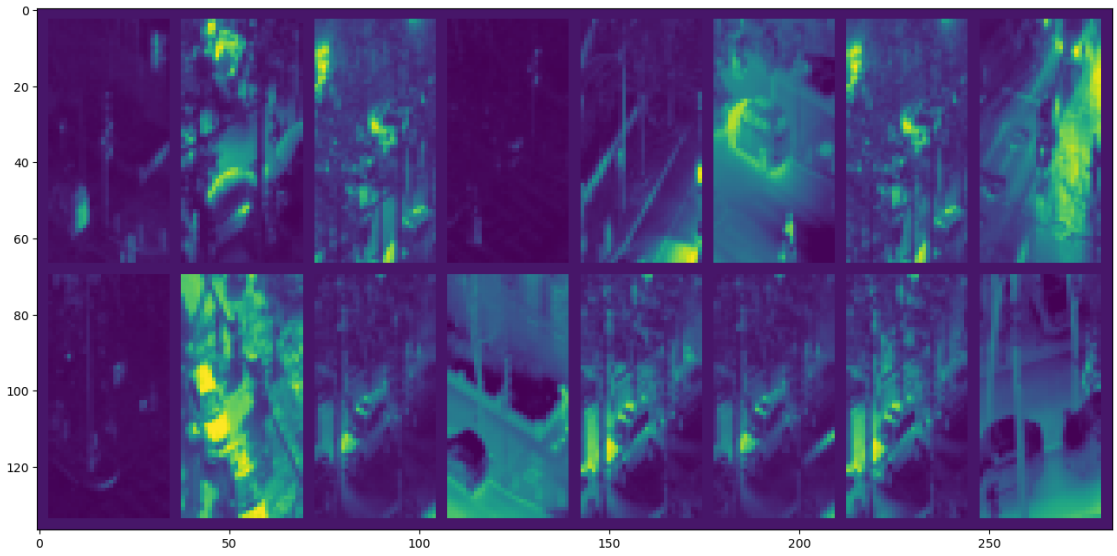
```
    transforms.Normalize([0.44 , 0.053, 0.062], [0.076, 0.079, 0.085])
])

train_ds = datasets.ImageFolder(trn_dir, transform=tfms)
test_ds = datasets.ImageFolder(tst_dir, transform=tfms)

len(train_ds), len(test_ds)
```

[10]:
```
(24050, 6926)
```

[11]:
```
train_dl = torch.utils.data.DataLoader(train_ds, batch_size=batch_size, ␣
 ↪shuffle=True, num_workers=8)
test_dl = torch.utils.data.DataLoader(test_ds, batch_size=batch_size,␣
 ↪shuffle=True, num_workers=8)
```

[12]:
```
inputs, targets = next(iter(train_dl))
out = torchvision.utils.make_grid(inputs, padding=3)
plt.figure(figsize=(16, 12))
plt.imshow(out[-1])
```

[12]:
```
<matplotlib.image.AxesImage at 0x7fa6255804c0>
```



[13]:
```
#define CNN model
class CNN(nn.Module):

    def __init__(self):
        super(CNN, self).__init__()
```

```
        self.conv1_1 = nn.Sequential(
            nn.Conv2d(1, 64, kernel_size=3, padding=1),
            nn.BatchNorm2d(64),
            nn.LeakyReLU(0.1, inplace=True),

        )

        self.conv1_2 = nn.Sequential(
            nn.Conv2d(1, 64, kernel_size=3, padding=1),
            nn.BatchNorm2d(64),

            nn.LeakyReLU(0.1, inplace=True),
        )

        self.conv1_3 = nn.Sequential(
            nn.Conv2d(1, 64, kernel_size=3, padding=1),
            nn.BatchNorm2d(64),
            nn.LeakyReLU(0.1, inplace=True),
        )

        self.conv2_1 = nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=3, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(2)
        )

        self.conv2_2 = nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=3, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(2)
        )

        self.conv2_3 = nn.Sequential(
            nn.Conv2d(64, 128, kernel_size=3, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(2)
        )

        self.conv3_1 = nn.Sequential(
            nn.Conv2d(128, 128, kernel_size=3, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(2)
```

```
        )

        self.conv3_2 = nn.Sequential(
            nn.Conv2d(128, 128, kernel_size=3, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(2)
        )

        self.conv3_3 = nn.Sequential(
            nn.Conv2d(128, 128, kernel_size=3, padding=1),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(2)
        )

        self.conv4 = nn.Sequential(
            nn.Conv2d(128, 256, kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(2)
        )

        self.conv5 = nn.Sequential(
            nn.Conv2d(128*2, 256, kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(2)
        )

        self.fc1 = nn.Linear(256*8*4*2, 2)

    def forward(self, x, y, z):
        outx = self.conv1_1(x)
        outx = self.conv2_1(outx)
        outx = self.conv3_1(outx)
        outx = self.conv4(outx)
        outx = outx.view(outx.size(0), -1)

        outy = self.conv1_2(y)
        outy = self.conv2_2(outy)
        outy = self.conv3_2(outy)

        outz = self.conv1_3(z)
        outz = self.conv2_3(outz)
        outz = self.conv3_3(outz)
```

```
        oyz=torch.cat([outy,outz],1)

        oyz = self.conv5(oyz)
        oyz = oyz.view(oyz.size(0), -1)

        oo=torch.cat([outx,oyz],1)
        out = self.fc1(oo)

        return out
```

[14]:
```
#Create model
model = CNN()

use_gpu = torch.cuda.is_available()
device = torch.device("cuda") if torch.cuda.is_available() else torch.
 ↪device("cpu")

if use_gpu:
    model = model.cuda()
    model.cuda()
    model = torch.nn.DataParallel(model, device_ids=range(torch.cuda.
 ↪device_count()))

device = torch.device("cuda") if torch.cuda.is_available() else torch.
 ↪device("cpu")
model
```

[14]:
```
CNN(
  (conv1_1): Sequential(
    (0): Conv2d(1, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.1, inplace=True)
  )
  (conv1_2): Sequential(
    (0): Conv2d(1, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.1, inplace=True)
  )
  (conv1_3): Sequential(
    (0): Conv2d(1, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.1, inplace=True)
  )
  (conv2_1): Sequential(
```

```
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.1, inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (conv2_2): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.1, inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (conv2_3): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.1, inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (conv3_1): Sequential(
    (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.1, inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (conv3_2): Sequential(
    (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.1, inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (conv3_3): Sequential(
    (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.1, inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
```

```
  (conv4): Sequential(
    (0): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.1, inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (conv5): Sequential(
    (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.1, inplace=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (fc1): Linear(in_features=16384, out_features=2, bias=True)
)
```

[15]:
```
criterion = nn.CrossEntropyLoss()

optimizer = optim.SGD(model.parameters(), lr=0.0002, momentum=0.9)
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=1, gamma=0.9)
```

[16]:
```
def to_var(x, volatile=False):
    if torch.cuda.is_available():
        x = x.cuda()
    return Variable(x, volatile=volatile)
```

[18]:
```
#train fit
a = time.time()
num_epochs = 20
losses = []
for epoch in range(num_epochs):
    for i, (inputs, targets) in enumerate(train_dl):

        inputs = to_var(inputs)
#         inputs2 = to_var(inputs2)
#         inputs3 = to_var(inputs3)
        targets = to_var(targets)

        inputs1=inputs[:,0,:,:]
        inputs1=inputs1.resize(inputs1.shape[0],1,64,32)
        inputs2=inputs[:,1,:,:]
        inputs2=inputs1.resize(inputs2.shape[0],1,64,32)
        inputs3=inputs[:,2,:,:]
        inputs3=inputs1.resize(inputs3.shape[0],1,64,32)
```

```
        # forwad pass
        optimizer.zero_grad()
        outputs = model(inputs1,inputs2,inputs3)

        # loss
        loss = criterion(outputs, targets)
        losses += [loss.item()]

        # backward pass
        loss.backward()

        # update parameters
        optimizer.step()

        # report
        if (i + 1) % 50 == 0:
            print('Epoch [%2d/%2d], Step [%3d/%3d], Loss: %.4f'
                  % (epoch + 1, num_epochs, i + 1, len(train_ds) // batch_size,␣
 ↪loss.item()))

b = time.time()
print('Total Time of Training {:.1000}s'.format(b - a))
```

```
Epoch [ 1/20], Step [ 50/1503], Loss: 0.0811
Epoch [ 1/20], Step [100/1503], Loss: 0.0089
Epoch [ 1/20], Step [150/1503], Loss: 0.0102
Epoch [ 1/20], Step [200/1503], Loss: 0.0043
Epoch [ 1/20], Step [250/1503], Loss: 0.0072
Epoch [ 1/20], Step [300/1503], Loss: 0.0076
Epoch [ 1/20], Step [350/1503], Loss: 0.0063
Epoch [ 1/20], Step [400/1503], Loss: 0.0041
Epoch [ 1/20], Step [450/1503], Loss: 0.0008
Epoch [ 1/20], Step [500/1503], Loss: 0.0049
Epoch [ 1/20], Step [550/1503], Loss: 0.0051
Epoch [ 1/20], Step [600/1503], Loss: 0.0028
Epoch [ 1/20], Step [650/1503], Loss: 0.0041
Epoch [ 1/20], Step [700/1503], Loss: 0.0030
Epoch [ 1/20], Step [750/1503], Loss: 0.0063
Epoch [ 1/20], Step [800/1503], Loss: 0.0117
Epoch [ 1/20], Step [850/1503], Loss: 0.0019
Epoch [ 1/20], Step [900/1503], Loss: 0.0370
Epoch [ 1/20], Step [950/1503], Loss: 0.0006
Epoch [ 1/20], Step [1000/1503], Loss: 0.0016
Epoch [ 1/20], Step [1050/1503], Loss: 0.0088
Epoch [ 1/20], Step [1100/1503], Loss: 0.0056
Epoch [ 1/20], Step [1150/1503], Loss: 0.0003
```

```
Epoch [ 1/20], Step [1200/1503], Loss: 0.0015
Epoch [ 1/20], Step [1250/1503], Loss: 0.0003
Epoch [ 1/20], Step [1300/1503], Loss: 0.0014
Epoch [ 1/20], Step [1350/1503], Loss: 0.0058
Epoch [ 1/20], Step [1400/1503], Loss: 0.0004
Epoch [ 1/20], Step [1450/1503], Loss: 0.0016
Epoch [ 1/20], Step [1500/1503], Loss: 0.0028
Epoch [ 2/20], Step [ 50/1503], Loss: 0.0014
Epoch [ 2/20], Step [100/1503], Loss: 0.0009
Epoch [ 2/20], Step [150/1503], Loss: 0.0023
Epoch [ 2/20], Step [200/1503], Loss: 0.0015
Epoch [ 2/20], Step [250/1503], Loss: 0.0012
Epoch [ 2/20], Step [300/1503], Loss: 0.0011
Epoch [ 2/20], Step [350/1503], Loss: 0.0017
Epoch [ 2/20], Step [400/1503], Loss: 0.0004
Epoch [ 2/20], Step [450/1503], Loss: 0.0006
Epoch [ 2/20], Step [500/1503], Loss: 0.0044
Epoch [ 2/20], Step [550/1503], Loss: 0.0011
Epoch [ 2/20], Step [600/1503], Loss: 0.0010
Epoch [ 2/20], Step [650/1503], Loss: 0.0021
Epoch [ 2/20], Step [700/1503], Loss: 0.0009
Epoch [ 2/20], Step [750/1503], Loss: 0.0009
Epoch [ 2/20], Step [800/1503], Loss: 0.0005
Epoch [ 2/20], Step [850/1503], Loss: 0.0021
Epoch [ 2/20], Step [900/1503], Loss: 0.0023
Epoch [ 2/20], Step [950/1503], Loss: 0.0017
Epoch [ 2/20], Step [1000/1503], Loss: 0.0011
Epoch [ 2/20], Step [1050/1503], Loss: 0.0005
Epoch [ 2/20], Step [1100/1503], Loss: 0.0003
Epoch [ 2/20], Step [1150/1503], Loss: 0.0291
Epoch [ 2/20], Step [1200/1503], Loss: 0.0007
Epoch [ 2/20], Step [1250/1503], Loss: 0.0011
Epoch [ 2/20], Step [1300/1503], Loss: 0.0019
Epoch [ 2/20], Step [1350/1503], Loss: 0.0004
Epoch [ 2/20], Step [1400/1503], Loss: 0.0004
Epoch [ 2/20], Step [1450/1503], Loss: 0.0004
Epoch [ 2/20], Step [1500/1503], Loss: 0.0004
Epoch [ 3/20], Step [ 50/1503], Loss: 0.0263
Epoch [ 3/20], Step [100/1503], Loss: 0.0008
Epoch [ 3/20], Step [150/1503], Loss: 0.0004
Epoch [ 3/20], Step [200/1503], Loss: 0.0009
Epoch [ 3/20], Step [250/1503], Loss: 0.0004
Epoch [ 3/20], Step [300/1503], Loss: 0.0004
Epoch [ 3/20], Step [350/1503], Loss: 0.0001
Epoch [ 3/20], Step [400/1503], Loss: 0.0026
Epoch [ 3/20], Step [450/1503], Loss: 0.0003
Epoch [ 3/20], Step [500/1503], Loss: 0.0009
Epoch [ 3/20], Step [550/1503], Loss: 0.0003
```

```
Epoch [ 3/20], Step [600/1503], Loss: 0.0004
Epoch [ 3/20], Step [650/1503], Loss: 0.0011
Epoch [ 3/20], Step [700/1503], Loss: 0.0003
Epoch [ 3/20], Step [750/1503], Loss: 0.0004
Epoch [ 3/20], Step [800/1503], Loss: 0.0002
Epoch [ 3/20], Step [850/1503], Loss: 0.0001
Epoch [ 3/20], Step [900/1503], Loss: 0.0003
Epoch [ 3/20], Step [950/1503], Loss: 0.0012
Epoch [ 3/20], Step [1000/1503], Loss: 0.0002
Epoch [ 3/20], Step [1050/1503], Loss: 0.0001
Epoch [ 3/20], Step [1100/1503], Loss: 0.0006
Epoch [ 3/20], Step [1150/1503], Loss: 0.0002
Epoch [ 3/20], Step [1200/1503], Loss: 0.0016
Epoch [ 3/20], Step [1250/1503], Loss: 0.0043
Epoch [ 3/20], Step [1300/1503], Loss: 0.0002
Epoch [ 3/20], Step [1350/1503], Loss: 0.0002
Epoch [ 3/20], Step [1400/1503], Loss: 0.0003
Epoch [ 3/20], Step [1450/1503], Loss: 0.0001
Epoch [ 3/20], Step [1500/1503], Loss: 0.0002
Epoch [ 4/20], Step [ 50/1503], Loss: 0.0001
Epoch [ 4/20], Step [100/1503], Loss: 0.0004
Epoch [ 4/20], Step [150/1503], Loss: 0.0001
Epoch [ 4/20], Step [200/1503], Loss: 0.0003
Epoch [ 4/20], Step [250/1503], Loss: 0.0007
Epoch [ 4/20], Step [300/1503], Loss: 0.0002
Epoch [ 4/20], Step [350/1503], Loss: 0.0001
Epoch [ 4/20], Step [400/1503], Loss: 0.0002
Epoch [ 4/20], Step [450/1503], Loss: 0.0003
Epoch [ 4/20], Step [500/1503], Loss: 0.0003
Epoch [ 4/20], Step [550/1503], Loss: 0.0002
Epoch [ 4/20], Step [600/1503], Loss: 0.0002
Epoch [ 4/20], Step [650/1503], Loss: 0.0002
Epoch [ 4/20], Step [700/1503], Loss: 0.0003
Epoch [ 4/20], Step [750/1503], Loss: 0.0001
Epoch [ 4/20], Step [800/1503], Loss: 0.0014
Epoch [ 4/20], Step [850/1503], Loss: 0.0001
Epoch [ 4/20], Step [900/1503], Loss: 0.0003
Epoch [ 4/20], Step [950/1503], Loss: 0.0001
Epoch [ 4/20], Step [1000/1503], Loss: 0.0005
Epoch [ 4/20], Step [1050/1503], Loss: 0.0003
Epoch [ 4/20], Step [1100/1503], Loss: 0.0002
Epoch [ 4/20], Step [1150/1503], Loss: 0.0020
Epoch [ 4/20], Step [1200/1503], Loss: 0.0002
Epoch [ 4/20], Step [1250/1503], Loss: 0.0002
Epoch [ 4/20], Step [1300/1503], Loss: 0.0003
Epoch [ 4/20], Step [1350/1503], Loss: 0.0001
Epoch [ 4/20], Step [1400/1503], Loss: 0.0001
Epoch [ 4/20], Step [1450/1503], Loss: 0.0006
```

```
Epoch [ 4/20], Step [1500/1503], Loss: 0.0011
Epoch [ 5/20], Step [ 50/1503], Loss: 0.0002
Epoch [ 5/20], Step [100/1503], Loss: 0.0001
Epoch [ 5/20], Step [150/1503], Loss: 0.0002
Epoch [ 5/20], Step [200/1503], Loss: 0.0004
Epoch [ 5/20], Step [250/1503], Loss: 0.0004
Epoch [ 5/20], Step [300/1503], Loss: 0.0001
Epoch [ 5/20], Step [350/1503], Loss: 0.0008
Epoch [ 5/20], Step [400/1503], Loss: 0.0019
Epoch [ 5/20], Step [450/1503], Loss: 0.0007
Epoch [ 5/20], Step [500/1503], Loss: 0.0002
Epoch [ 5/20], Step [550/1503], Loss: 0.0003
Epoch [ 5/20], Step [600/1503], Loss: 0.0010
Epoch [ 5/20], Step [650/1503], Loss: 0.0005
Epoch [ 5/20], Step [700/1503], Loss: 0.0001
Epoch [ 5/20], Step [750/1503], Loss: 0.0001
Epoch [ 5/20], Step [800/1503], Loss: 0.0002
Epoch [ 5/20], Step [850/1503], Loss: 0.0003
Epoch [ 5/20], Step [900/1503], Loss: 0.0002
Epoch [ 5/20], Step [950/1503], Loss: 0.0000
Epoch [ 5/20], Step [1000/1503], Loss: 0.0001
Epoch [ 5/20], Step [1050/1503], Loss: 0.0002
Epoch [ 5/20], Step [1100/1503], Loss: 0.0001
Epoch [ 5/20], Step [1150/1503], Loss: 0.0001
Epoch [ 5/20], Step [1200/1503], Loss: 0.0003
Epoch [ 5/20], Step [1250/1503], Loss: 0.0001
Epoch [ 5/20], Step [1300/1503], Loss: 0.0001
Epoch [ 5/20], Step [1350/1503], Loss: 0.0001
Epoch [ 5/20], Step [1400/1503], Loss: 0.0002
Epoch [ 5/20], Step [1450/1503], Loss: 0.0005
Epoch [ 5/20], Step [1500/1503], Loss: 0.0004
Epoch [ 6/20], Step [ 50/1503], Loss: 0.0005
Epoch [ 6/20], Step [100/1503], Loss: 0.0002
Epoch [ 6/20], Step [150/1503], Loss: 0.0001
Epoch [ 6/20], Step [200/1503], Loss: 0.0001
Epoch [ 6/20], Step [250/1503], Loss: 0.0002
Epoch [ 6/20], Step [300/1503], Loss: 0.0004
Epoch [ 6/20], Step [350/1503], Loss: 0.0001
Epoch [ 6/20], Step [400/1503], Loss: 0.0001
Epoch [ 6/20], Step [450/1503], Loss: 0.0001
Epoch [ 6/20], Step [500/1503], Loss: 0.0002
Epoch [ 6/20], Step [550/1503], Loss: 0.0002
Epoch [ 6/20], Step [600/1503], Loss: 0.0004
Epoch [ 6/20], Step [650/1503], Loss: 0.0002
Epoch [ 6/20], Step [700/1503], Loss: 0.0004
Epoch [ 6/20], Step [750/1503], Loss: 0.0001
Epoch [ 6/20], Step [800/1503], Loss: 0.0002
Epoch [ 6/20], Step [850/1503], Loss: 0.0007
```

```
Epoch [ 6/20], Step [900/1503], Loss: 0.0001
Epoch [ 6/20], Step [950/1503], Loss: 0.0002
Epoch [ 6/20], Step [1000/1503], Loss: 0.0004
Epoch [ 6/20], Step [1050/1503], Loss: 0.0001
Epoch [ 6/20], Step [1100/1503], Loss: 0.0013
Epoch [ 6/20], Step [1150/1503], Loss: 0.0002
Epoch [ 6/20], Step [1200/1503], Loss: 0.0002
Epoch [ 6/20], Step [1250/1503], Loss: 0.0003
Epoch [ 6/20], Step [1300/1503], Loss: 0.0005
Epoch [ 6/20], Step [1350/1503], Loss: 0.0003
Epoch [ 6/20], Step [1400/1503], Loss: 0.0003
Epoch [ 6/20], Step [1450/1503], Loss: 0.0001
Epoch [ 6/20], Step [1500/1503], Loss: 0.0002
Epoch [ 7/20], Step [ 50/1503], Loss: 0.0005
Epoch [ 7/20], Step [100/1503], Loss: 0.0002
Epoch [ 7/20], Step [150/1503], Loss: 0.0003
Epoch [ 7/20], Step [200/1503], Loss: 0.0001
Epoch [ 7/20], Step [250/1503], Loss: 0.0003
Epoch [ 7/20], Step [300/1503], Loss: 0.0001
Epoch [ 7/20], Step [350/1503], Loss: 0.0003
Epoch [ 7/20], Step [400/1503], Loss: 0.0006
Epoch [ 7/20], Step [450/1503], Loss: 0.0001
Epoch [ 7/20], Step [500/1503], Loss: 0.0002
Epoch [ 7/20], Step [550/1503], Loss: 0.0001
Epoch [ 7/20], Step [600/1503], Loss: 0.0003
Epoch [ 7/20], Step [650/1503], Loss: 0.0005
Epoch [ 7/20], Step [700/1503], Loss: 0.0001
Epoch [ 7/20], Step [750/1503], Loss: 0.0001
Epoch [ 7/20], Step [800/1503], Loss: 0.0002
Epoch [ 7/20], Step [850/1503], Loss: 0.0001
Epoch [ 7/20], Step [900/1503], Loss: 0.0000
Epoch [ 7/20], Step [950/1503], Loss: 0.0003
Epoch [ 7/20], Step [1000/1503], Loss: 0.0002
Epoch [ 7/20], Step [1050/1503], Loss: 0.0005
Epoch [ 7/20], Step [1100/1503], Loss: 0.0031
Epoch [ 7/20], Step [1150/1503], Loss: 0.0001
Epoch [ 7/20], Step [1200/1503], Loss: 0.0001
Epoch [ 7/20], Step [1250/1503], Loss: 0.0001
Epoch [ 7/20], Step [1300/1503], Loss: 0.0001
Epoch [ 7/20], Step [1350/1503], Loss: 0.0002
Epoch [ 7/20], Step [1400/1503], Loss: 0.0015
Epoch [ 7/20], Step [1450/1503], Loss: 0.0002
Epoch [ 7/20], Step [1500/1503], Loss: 0.0003
Epoch [ 8/20], Step [ 50/1503], Loss: 0.0005
Epoch [ 8/20], Step [100/1503], Loss: 0.0003
Epoch [ 8/20], Step [150/1503], Loss: 0.0002
Epoch [ 8/20], Step [200/1503], Loss: 0.0001
Epoch [ 8/20], Step [250/1503], Loss: 0.0002
```

```
Epoch [ 8/20], Step [300/1503], Loss: 0.0001
Epoch [ 8/20], Step [350/1503], Loss: 0.0004
Epoch [ 8/20], Step [400/1503], Loss: 0.0000
Epoch [ 8/20], Step [450/1503], Loss: 0.0001
Epoch [ 8/20], Step [500/1503], Loss: 0.0000
Epoch [ 8/20], Step [550/1503], Loss: 0.0001
Epoch [ 8/20], Step [600/1503], Loss: 0.0001
Epoch [ 8/20], Step [650/1503], Loss: 0.0002
Epoch [ 8/20], Step [700/1503], Loss: 0.0006
Epoch [ 8/20], Step [750/1503], Loss: 0.0002
Epoch [ 8/20], Step [800/1503], Loss: 0.0001
Epoch [ 8/20], Step [850/1503], Loss: 0.0009
Epoch [ 8/20], Step [900/1503], Loss: 0.0002
Epoch [ 8/20], Step [950/1503], Loss: 0.0001
Epoch [ 8/20], Step [1000/1503], Loss: 0.0001
Epoch [ 8/20], Step [1050/1503], Loss: 0.0000
Epoch [ 8/20], Step [1100/1503], Loss: 0.0001
Epoch [ 8/20], Step [1150/1503], Loss: 0.0001
Epoch [ 8/20], Step [1200/1503], Loss: 0.0002
Epoch [ 8/20], Step [1250/1503], Loss: 0.0002
Epoch [ 8/20], Step [1300/1503], Loss: 0.0005
Epoch [ 8/20], Step [1350/1503], Loss: 0.0009
Epoch [ 8/20], Step [1400/1503], Loss: 0.0005
Epoch [ 8/20], Step [1450/1503], Loss: 0.0001
Epoch [ 8/20], Step [1500/1503], Loss: 0.0001
Epoch [ 9/20], Step [ 50/1503], Loss: 0.0005
Epoch [ 9/20], Step [100/1503], Loss: 0.0002
Epoch [ 9/20], Step [150/1503], Loss: 0.0001
Epoch [ 9/20], Step [200/1503], Loss: 0.0000
Epoch [ 9/20], Step [250/1503], Loss: 0.0001
Epoch [ 9/20], Step [300/1503], Loss: 0.0003
Epoch [ 9/20], Step [350/1503], Loss: 0.0001
Epoch [ 9/20], Step [400/1503], Loss: 0.0001
Epoch [ 9/20], Step [450/1503], Loss: 0.0001
Epoch [ 9/20], Step [500/1503], Loss: 0.0000
Epoch [ 9/20], Step [550/1503], Loss: 0.0003
Epoch [ 9/20], Step [600/1503], Loss: 0.0000
Epoch [ 9/20], Step [650/1503], Loss: 0.0002
Epoch [ 9/20], Step [700/1503], Loss: 0.0001
Epoch [ 9/20], Step [750/1503], Loss: 0.0000
Epoch [ 9/20], Step [800/1503], Loss: 0.0001
Epoch [ 9/20], Step [850/1503], Loss: 0.0001
Epoch [ 9/20], Step [900/1503], Loss: 0.0007
Epoch [ 9/20], Step [950/1503], Loss: 0.0001
Epoch [ 9/20], Step [1000/1503], Loss: 0.0004
Epoch [ 9/20], Step [1050/1503], Loss: 0.0000
Epoch [ 9/20], Step [1100/1503], Loss: 0.0001
Epoch [ 9/20], Step [1150/1503], Loss: 0.0002
```

```
Epoch [ 9/20], Step [1200/1503], Loss: 0.0003
Epoch [ 9/20], Step [1250/1503], Loss: 0.0002
Epoch [ 9/20], Step [1300/1503], Loss: 0.0001
Epoch [ 9/20], Step [1350/1503], Loss: 0.0001
Epoch [ 9/20], Step [1400/1503], Loss: 0.0004
Epoch [ 9/20], Step [1450/1503], Loss: 0.0001
Epoch [ 9/20], Step [1500/1503], Loss: 0.0000
Epoch [10/20], Step [ 50/1503], Loss: 0.0002
Epoch [10/20], Step [100/1503], Loss: 0.0000
Epoch [10/20], Step [150/1503], Loss: 0.0000
Epoch [10/20], Step [200/1503], Loss: 0.0001
Epoch [10/20], Step [250/1503], Loss: 0.0000
Epoch [10/20], Step [300/1503], Loss: 0.0000
Epoch [10/20], Step [350/1503], Loss: 0.0002
Epoch [10/20], Step [400/1503], Loss: 0.0001
Epoch [10/20], Step [450/1503], Loss: 0.0000
Epoch [10/20], Step [500/1503], Loss: 0.0004
Epoch [10/20], Step [550/1503], Loss: 0.0000
Epoch [10/20], Step [600/1503], Loss: 0.0002
Epoch [10/20], Step [650/1503], Loss: 0.0000
Epoch [10/20], Step [700/1503], Loss: 0.0001
Epoch [10/20], Step [750/1503], Loss: 0.0000
Epoch [10/20], Step [800/1503], Loss: 0.0002
Epoch [10/20], Step [850/1503], Loss: 0.0001
Epoch [10/20], Step [900/1503], Loss: 0.0001
Epoch [10/20], Step [950/1503], Loss: 0.0002
Epoch [10/20], Step [1000/1503], Loss: 0.0000
Epoch [10/20], Step [1050/1503], Loss: 0.0000
Epoch [10/20], Step [1100/1503], Loss: 0.0001
Epoch [10/20], Step [1150/1503], Loss: 0.0005
Epoch [10/20], Step [1200/1503], Loss: 0.0000
Epoch [10/20], Step [1250/1503], Loss: 0.0000
Epoch [10/20], Step [1300/1503], Loss: 0.0000
Epoch [10/20], Step [1350/1503], Loss: 0.0001
Epoch [10/20], Step [1400/1503], Loss: 0.0001
Epoch [10/20], Step [1450/1503], Loss: 0.0002
Epoch [10/20], Step [1500/1503], Loss: 0.0001
Epoch [11/20], Step [ 50/1503], Loss: 0.0001
Epoch [11/20], Step [100/1503], Loss: 0.0001
Epoch [11/20], Step [150/1503], Loss: 0.0002
Epoch [11/20], Step [200/1503], Loss: 0.0000
Epoch [11/20], Step [250/1503], Loss: 0.0000
Epoch [11/20], Step [300/1503], Loss: 0.0000
Epoch [11/20], Step [350/1503], Loss: 0.0008
Epoch [11/20], Step [400/1503], Loss: 0.0002
Epoch [11/20], Step [450/1503], Loss: 0.0000
Epoch [11/20], Step [500/1503], Loss: 0.0002
Epoch [11/20], Step [550/1503], Loss: 0.0001
```

```
Epoch [11/20], Step [600/1503], Loss: 0.0001
Epoch [11/20], Step [650/1503], Loss: 0.0002
Epoch [11/20], Step [700/1503], Loss: 0.0000
Epoch [11/20], Step [750/1503], Loss: 0.0000
Epoch [11/20], Step [800/1503], Loss: 0.0000
Epoch [11/20], Step [850/1503], Loss: 0.0001
Epoch [11/20], Step [900/1503], Loss: 0.0004
Epoch [11/20], Step [950/1503], Loss: 0.0001
Epoch [11/20], Step [1000/1503], Loss: 0.0029
Epoch [11/20], Step [1050/1503], Loss: 0.0001
Epoch [11/20], Step [1100/1503], Loss: 0.0000
Epoch [11/20], Step [1150/1503], Loss: 0.0001
Epoch [11/20], Step [1200/1503], Loss: 0.0002
Epoch [11/20], Step [1250/1503], Loss: 0.0004
Epoch [11/20], Step [1300/1503], Loss: 0.0000
Epoch [11/20], Step [1350/1503], Loss: 0.0001
Epoch [11/20], Step [1400/1503], Loss: 0.0000
Epoch [11/20], Step [1450/1503], Loss: 0.0001
Epoch [11/20], Step [1500/1503], Loss: 0.0003
Epoch [12/20], Step [ 50/1503], Loss: 0.0001
Epoch [12/20], Step [100/1503], Loss: 0.0001
Epoch [12/20], Step [150/1503], Loss: 0.0001
Epoch [12/20], Step [200/1503], Loss: 0.0112
Epoch [12/20], Step [250/1503], Loss: 0.0000
Epoch [12/20], Step [300/1503], Loss: 0.0001
Epoch [12/20], Step [350/1503], Loss: 0.0002
Epoch [12/20], Step [400/1503], Loss: 0.0002
Epoch [12/20], Step [450/1503], Loss: 0.0002
Epoch [12/20], Step [500/1503], Loss: 0.0005
Epoch [12/20], Step [550/1503], Loss: 0.0000
Epoch [12/20], Step [600/1503], Loss: 0.0000
Epoch [12/20], Step [650/1503], Loss: 0.0003
Epoch [12/20], Step [700/1503], Loss: 0.0002
Epoch [12/20], Step [750/1503], Loss: 0.0000
Epoch [12/20], Step [800/1503], Loss: 0.0001
Epoch [12/20], Step [850/1503], Loss: 0.0001
Epoch [12/20], Step [900/1503], Loss: 0.0000
Epoch [12/20], Step [950/1503], Loss: 0.0000
Epoch [12/20], Step [1000/1503], Loss: 0.0000
Epoch [12/20], Step [1050/1503], Loss: 0.0001
Epoch [12/20], Step [1100/1503], Loss: 0.0000
Epoch [12/20], Step [1150/1503], Loss: 0.0000
Epoch [12/20], Step [1200/1503], Loss: 0.0000
Epoch [12/20], Step [1250/1503], Loss: 0.0001
Epoch [12/20], Step [1300/1503], Loss: 0.0001
Epoch [12/20], Step [1350/1503], Loss: 0.0002
Epoch [12/20], Step [1400/1503], Loss: 0.0000
Epoch [12/20], Step [1450/1503], Loss: 0.0000
```

```
Epoch [12/20], Step [1500/1503], Loss: 0.0000
Epoch [13/20], Step [ 50/1503], Loss: 0.0001
Epoch [13/20], Step [100/1503], Loss: 0.0000
Epoch [13/20], Step [150/1503], Loss: 0.0002
Epoch [13/20], Step [200/1503], Loss: 0.0001
Epoch [13/20], Step [250/1503], Loss: 0.0000
Epoch [13/20], Step [300/1503], Loss: 0.0000
Epoch [13/20], Step [350/1503], Loss: 0.0009
Epoch [13/20], Step [400/1503], Loss: 0.0000
Epoch [13/20], Step [450/1503], Loss: 0.0001
Epoch [13/20], Step [500/1503], Loss: 0.0003
Epoch [13/20], Step [550/1503], Loss: 0.0001
Epoch [13/20], Step [600/1503], Loss: 0.0000
Epoch [13/20], Step [650/1503], Loss: 0.0001
Epoch [13/20], Step [700/1503], Loss: 0.0000
Epoch [13/20], Step [750/1503], Loss: 0.0001
Epoch [13/20], Step [800/1503], Loss: 0.0001
Epoch [13/20], Step [850/1503], Loss: 0.0001
Epoch [13/20], Step [900/1503], Loss: 0.0000
Epoch [13/20], Step [950/1503], Loss: 0.0000
Epoch [13/20], Step [1000/1503], Loss: 0.0001
Epoch [13/20], Step [1050/1503], Loss: 0.0001
Epoch [13/20], Step [1100/1503], Loss: 0.0002
Epoch [13/20], Step [1150/1503], Loss: 0.0001
Epoch [13/20], Step [1200/1503], Loss: 0.0000
Epoch [13/20], Step [1250/1503], Loss: 0.0000
Epoch [13/20], Step [1300/1503], Loss: 0.0001
Epoch [13/20], Step [1350/1503], Loss: 0.0007
Epoch [13/20], Step [1400/1503], Loss: 0.0000
Epoch [13/20], Step [1450/1503], Loss: 0.0001
Epoch [13/20], Step [1500/1503], Loss: 0.0000
Epoch [14/20], Step [ 50/1503], Loss: 0.0010
Epoch [14/20], Step [100/1503], Loss: 0.0000
Epoch [14/20], Step [150/1503], Loss: 0.0000
Epoch [14/20], Step [200/1503], Loss: 0.0003
Epoch [14/20], Step [250/1503], Loss: 0.0002
Epoch [14/20], Step [300/1503], Loss: 0.0001
Epoch [14/20], Step [350/1503], Loss: 0.0001
Epoch [14/20], Step [400/1503], Loss: 0.0001
Epoch [14/20], Step [450/1503], Loss: 0.0001
Epoch [14/20], Step [500/1503], Loss: 0.0000
Epoch [14/20], Step [550/1503], Loss: 0.0000
Epoch [14/20], Step [600/1503], Loss: 0.0005
Epoch [14/20], Step [650/1503], Loss: 0.0000
Epoch [14/20], Step [700/1503], Loss: 0.0000
Epoch [14/20], Step [750/1503], Loss: 0.0001
Epoch [14/20], Step [800/1503], Loss: 0.0001
Epoch [14/20], Step [850/1503], Loss: 0.0001
```

```
Epoch [14/20], Step [900/1503], Loss: 0.0001
Epoch [14/20], Step [950/1503], Loss: 0.0002
Epoch [14/20], Step [1000/1503], Loss: 0.0000
Epoch [14/20], Step [1050/1503], Loss: 0.0002
Epoch [14/20], Step [1100/1503], Loss: 0.0003
Epoch [14/20], Step [1150/1503], Loss: 0.0001
Epoch [14/20], Step [1200/1503], Loss: 0.0001
Epoch [14/20], Step [1250/1503], Loss: 0.0001
Epoch [14/20], Step [1300/1503], Loss: 0.0000
Epoch [14/20], Step [1350/1503], Loss: 0.0006
Epoch [14/20], Step [1400/1503], Loss: 0.0000
Epoch [14/20], Step [1450/1503], Loss: 0.0002
Epoch [14/20], Step [1500/1503], Loss: 0.0000
Epoch [15/20], Step [ 50/1503], Loss: 0.0000
Epoch [15/20], Step [100/1503], Loss: 0.0001
Epoch [15/20], Step [150/1503], Loss: 0.0000
Epoch [15/20], Step [200/1503], Loss: 0.0000
Epoch [15/20], Step [250/1503], Loss: 0.0001
Epoch [15/20], Step [300/1503], Loss: 0.0001
Epoch [15/20], Step [350/1503], Loss: 0.0000
Epoch [15/20], Step [400/1503], Loss: 0.0004
Epoch [15/20], Step [450/1503], Loss: 0.0001
Epoch [15/20], Step [500/1503], Loss: 0.0001
Epoch [15/20], Step [550/1503], Loss: 0.0000
Epoch [15/20], Step [600/1503], Loss: 0.0001
Epoch [15/20], Step [650/1503], Loss: 0.0001
Epoch [15/20], Step [700/1503], Loss: 0.0001
Epoch [15/20], Step [750/1503], Loss: 0.0001
Epoch [15/20], Step [800/1503], Loss: 0.0001
Epoch [15/20], Step [850/1503], Loss: 0.0001
Epoch [15/20], Step [900/1503], Loss: 0.0000
Epoch [15/20], Step [950/1503], Loss: 0.0001
Epoch [15/20], Step [1000/1503], Loss: 0.0001
Epoch [15/20], Step [1050/1503], Loss: 0.0001
Epoch [15/20], Step [1100/1503], Loss: 0.0002
Epoch [15/20], Step [1150/1503], Loss: 0.0009
Epoch [15/20], Step [1200/1503], Loss: 0.0042
Epoch [15/20], Step [1250/1503], Loss: 0.0010
Epoch [15/20], Step [1300/1503], Loss: 0.0001
Epoch [15/20], Step [1350/1503], Loss: 0.0001
Epoch [15/20], Step [1400/1503], Loss: 0.0001
Epoch [15/20], Step [1450/1503], Loss: 0.0000
Epoch [15/20], Step [1500/1503], Loss: 0.0001
Epoch [16/20], Step [ 50/1503], Loss: 0.0001
Epoch [16/20], Step [100/1503], Loss: 0.0001
Epoch [16/20], Step [150/1503], Loss: 0.0001
Epoch [16/20], Step [200/1503], Loss: 0.0000
Epoch [16/20], Step [250/1503], Loss: 0.0000
```

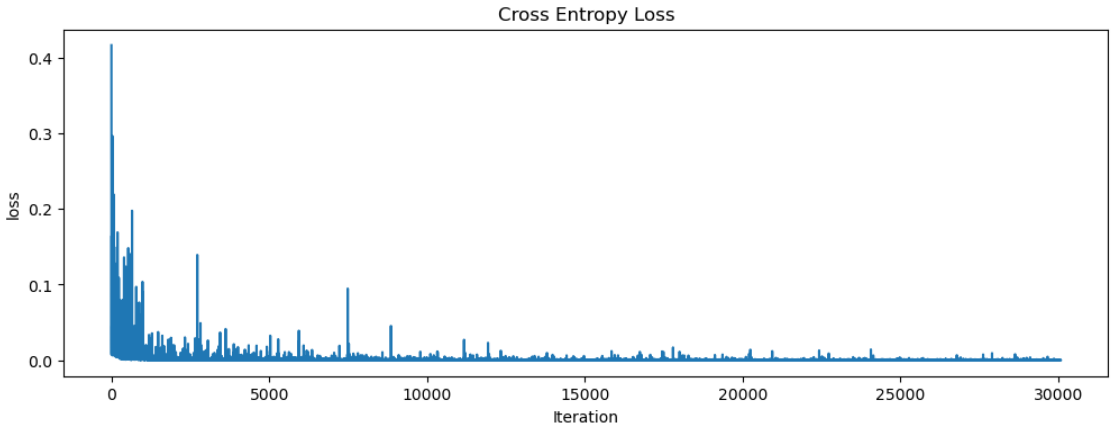
```
Epoch [16/20], Step [300/1503], Loss: 0.0001
Epoch [16/20], Step [350/1503], Loss: 0.0002
Epoch [16/20], Step [400/1503], Loss: 0.0000
Epoch [16/20], Step [450/1503], Loss: 0.0001
Epoch [16/20], Step [500/1503], Loss: 0.0001
Epoch [16/20], Step [550/1503], Loss: 0.0000
Epoch [16/20], Step [600/1503], Loss: 0.0000
Epoch [16/20], Step [650/1503], Loss: 0.0000
Epoch [16/20], Step [700/1503], Loss: 0.0000
Epoch [16/20], Step [750/1503], Loss: 0.0002
Epoch [16/20], Step [800/1503], Loss: 0.0000
Epoch [16/20], Step [850/1503], Loss: 0.0001
Epoch [16/20], Step [900/1503], Loss: 0.0001
Epoch [16/20], Step [950/1503], Loss: 0.0000
Epoch [16/20], Step [1000/1503], Loss: 0.0001
Epoch [16/20], Step [1050/1503], Loss: 0.0000
Epoch [16/20], Step [1100/1503], Loss: 0.0001
Epoch [16/20], Step [1150/1503], Loss: 0.0002
Epoch [16/20], Step [1200/1503], Loss: 0.0000
Epoch [16/20], Step [1250/1503], Loss: 0.0000
Epoch [16/20], Step [1300/1503], Loss: 0.0009
Epoch [16/20], Step [1350/1503], Loss: 0.0000
Epoch [16/20], Step [1400/1503], Loss: 0.0001
Epoch [16/20], Step [1450/1503], Loss: 0.0000
Epoch [16/20], Step [1500/1503], Loss: 0.0002
Epoch [17/20], Step [ 50/1503], Loss: 0.0001
Epoch [17/20], Step [100/1503], Loss: 0.0000
Epoch [17/20], Step [150/1503], Loss: 0.0014
Epoch [17/20], Step [200/1503], Loss: 0.0000
Epoch [17/20], Step [250/1503], Loss: 0.0001
Epoch [17/20], Step [300/1503], Loss: 0.0000
Epoch [17/20], Step [350/1503], Loss: 0.0004
Epoch [17/20], Step [400/1503], Loss: 0.0000
Epoch [17/20], Step [450/1503], Loss: 0.0001
Epoch [17/20], Step [500/1503], Loss: 0.0000
Epoch [17/20], Step [550/1503], Loss: 0.0002
Epoch [17/20], Step [600/1503], Loss: 0.0001
Epoch [17/20], Step [650/1503], Loss: 0.0000
Epoch [17/20], Step [700/1503], Loss: 0.0000
Epoch [17/20], Step [750/1503], Loss: 0.0000
Epoch [17/20], Step [800/1503], Loss: 0.0000
Epoch [17/20], Step [850/1503], Loss: 0.0002
Epoch [17/20], Step [900/1503], Loss: 0.0002
Epoch [17/20], Step [950/1503], Loss: 0.0001
Epoch [17/20], Step [1000/1503], Loss: 0.0000
Epoch [17/20], Step [1050/1503], Loss: 0.0001
Epoch [17/20], Step [1100/1503], Loss: 0.0000
Epoch [17/20], Step [1150/1503], Loss: 0.0009
```

```
Epoch [17/20], Step [1200/1503], Loss: 0.0002
Epoch [17/20], Step [1250/1503], Loss: 0.0000
Epoch [17/20], Step [1300/1503], Loss: 0.0000
Epoch [17/20], Step [1350/1503], Loss: 0.0000
Epoch [17/20], Step [1400/1503], Loss: 0.0004
Epoch [17/20], Step [1450/1503], Loss: 0.0001
Epoch [17/20], Step [1500/1503], Loss: 0.0000
Epoch [18/20], Step [ 50/1503], Loss: 0.0000
Epoch [18/20], Step [100/1503], Loss: 0.0001
Epoch [18/20], Step [150/1503], Loss: 0.0029
Epoch [18/20], Step [200/1503], Loss: 0.0001
Epoch [18/20], Step [250/1503], Loss: 0.0001
Epoch [18/20], Step [300/1503], Loss: 0.0000
Epoch [18/20], Step [350/1503], Loss: 0.0000
Epoch [18/20], Step [400/1503], Loss: 0.0001
Epoch [18/20], Step [450/1503], Loss: 0.0001
Epoch [18/20], Step [500/1503], Loss: 0.0009
Epoch [18/20], Step [550/1503], Loss: 0.0000
Epoch [18/20], Step [600/1503], Loss: 0.0017
Epoch [18/20], Step [650/1503], Loss: 0.0003
Epoch [18/20], Step [700/1503], Loss: 0.0001
Epoch [18/20], Step [750/1503], Loss: 0.0001
Epoch [18/20], Step [800/1503], Loss: 0.0000
Epoch [18/20], Step [850/1503], Loss: 0.0002
Epoch [18/20], Step [900/1503], Loss: 0.0000
Epoch [18/20], Step [950/1503], Loss: 0.0000
Epoch [18/20], Step [1000/1503], Loss: 0.0000
Epoch [18/20], Step [1050/1503], Loss: 0.0000
Epoch [18/20], Step [1100/1503], Loss: 0.0008
Epoch [18/20], Step [1150/1503], Loss: 0.0002
Epoch [18/20], Step [1200/1503], Loss: 0.0000
Epoch [18/20], Step [1250/1503], Loss: 0.0002
Epoch [18/20], Step [1300/1503], Loss: 0.0018
Epoch [18/20], Step [1350/1503], Loss: 0.0002
Epoch [18/20], Step [1400/1503], Loss: 0.0000
Epoch [18/20], Step [1450/1503], Loss: 0.0001
Epoch [18/20], Step [1500/1503], Loss: 0.0001
Epoch [19/20], Step [ 50/1503], Loss: 0.0002
Epoch [19/20], Step [100/1503], Loss: 0.0002
Epoch [19/20], Step [150/1503], Loss: 0.0000
Epoch [19/20], Step [200/1503], Loss: 0.0001
Epoch [19/20], Step [250/1503], Loss: 0.0001
Epoch [19/20], Step [300/1503], Loss: 0.0000
Epoch [19/20], Step [350/1503], Loss: 0.0001
Epoch [19/20], Step [400/1503], Loss: 0.0000
Epoch [19/20], Step [450/1503], Loss: 0.0000
Epoch [19/20], Step [500/1503], Loss: 0.0002
Epoch [19/20], Step [550/1503], Loss: 0.0001
```

```
Epoch [19/20], Step [600/1503], Loss: 0.0001
Epoch [19/20], Step [650/1503], Loss: 0.0000
Epoch [19/20], Step [700/1503], Loss: 0.0003
Epoch [19/20], Step [750/1503], Loss: 0.0001
Epoch [19/20], Step [800/1503], Loss: 0.0000
Epoch [19/20], Step [850/1503], Loss: 0.0006
Epoch [19/20], Step [900/1503], Loss: 0.0000
Epoch [19/20], Step [950/1503], Loss: 0.0002
Epoch [19/20], Step [1000/1503], Loss: 0.0000
Epoch [19/20], Step [1050/1503], Loss: 0.0001
Epoch [19/20], Step [1100/1503], Loss: 0.0000
Epoch [19/20], Step [1150/1503], Loss: 0.0001
Epoch [19/20], Step [1200/1503], Loss: 0.0000
Epoch [19/20], Step [1250/1503], Loss: 0.0001
Epoch [19/20], Step [1300/1503], Loss: 0.0001
Epoch [19/20], Step [1350/1503], Loss: 0.0001
Epoch [19/20], Step [1400/1503], Loss: 0.0003
Epoch [19/20], Step [1450/1503], Loss: 0.0001
Epoch [19/20], Step [1500/1503], Loss: 0.0000
Epoch [20/20], Step [ 50/1503], Loss: 0.0002
Epoch [20/20], Step [100/1503], Loss: 0.0001
Epoch [20/20], Step [150/1503], Loss: 0.0000
Epoch [20/20], Step [200/1503], Loss: 0.0000
Epoch [20/20], Step [250/1503], Loss: 0.0000
Epoch [20/20], Step [300/1503], Loss: 0.0000
Epoch [20/20], Step [350/1503], Loss: 0.0000
Epoch [20/20], Step [400/1503], Loss: 0.0000
Epoch [20/20], Step [450/1503], Loss: 0.0005
Epoch [20/20], Step [500/1503], Loss: 0.0005
Epoch [20/20], Step [550/1503], Loss: 0.0000
Epoch [20/20], Step [600/1503], Loss: 0.0001
Epoch [20/20], Step [650/1503], Loss: 0.0002
Epoch [20/20], Step [700/1503], Loss: 0.0000
Epoch [20/20], Step [750/1503], Loss: 0.0000
Epoch [20/20], Step [800/1503], Loss: 0.0000
Epoch [20/20], Step [850/1503], Loss: 0.0001
Epoch [20/20], Step [900/1503], Loss: 0.0001
Epoch [20/20], Step [950/1503], Loss: 0.0001
Epoch [20/20], Step [1000/1503], Loss: 0.0000
Epoch [20/20], Step [1050/1503], Loss: 0.0000
Epoch [20/20], Step [1100/1503], Loss: 0.0001
Epoch [20/20], Step [1150/1503], Loss: 0.0001
Epoch [20/20], Step [1200/1503], Loss: 0.0004
Epoch [20/20], Step [1250/1503], Loss: 0.0001
Epoch [20/20], Step [1300/1503], Loss: 0.0000
Epoch [20/20], Step [1350/1503], Loss: 0.0000
Epoch [20/20], Step [1400/1503], Loss: 0.0001
Epoch [20/20], Step [1450/1503], Loss: 0.0000
```

```
Epoch [20/20], Step [1500/1503], Loss: 0.0000
Total Time of Training 27187.922503232955932617 1875s
```

[23]:
```
#Visualizing model performance
plt.figure(figsize=(12, 4))
plt.plot(losses)
plt.xlabel('Iteration')
plt.ylabel('loss')
plt.title('Cross Entropy Loss');
plt.show()
```



[24]:
```
#Evaluate model performance
def evaluate_model(model, dataloader):
    # for batch normalization layers
    model.eval()
    corrects = 0
    for inputs, targets in dataloader:
        inputs, targets = to_var(inputs, True), to_var(targets, True)
#         targets = to_var(targets)

        inputs1=inputs[:,0,:,:]
        inputs1=inputs1.resize(inputs1.shape[0],1,64,32)
        inputs2=inputs[:,1,:,:]
        inputs2=inputs1.resize(inputs2.shape[0],1,64,32)
        inputs3=inputs[:,2,:,:]
        inputs3=inputs1.resize(inputs3.shape[0],1,64,32)

        outputs = model(inputs1,inputs2,inputs3)
        _, preds = torch.max(outputs.data, 1)
        corrects += (preds == targets.data).sum()

    zz=len(dataloader.dataset)
```

```
    print('accuracy: {:.2f}'.format(100. * corrects / len(dataloader.dataset)))
    print('corrects: {:.2f}'.format(corrects))
    print('Toatal: {:.2f}'.format(zz))
```

[ ]:
```
#Save model
evaluate_model(model, train_dl)
evaluate_model(model, test_dl)
torch.save(model.state_dict(), 'ECNN_network_wights.pth')
```

```
/tmp/ipykernel_44780/2390227020.py:4: UserWarning: volatile was removed and now
has no effect. Use `with torch.no_grad():` instead.
  return Variable(x, volatile=volatile)

accuracy: 100.00
corrects: 24050.00
Toatal: 24050.00
```

[ ]:
```

```