Non CNN Classification Models

December 16, 2022

[21]:
```
import numpy as np
import sklearn
import torch
import torchvision
from torchvision import transforms
from torch.utils.data import DataLoader, random_split
from sklearn.model_selection import train_test_split
from torchvision.datasets import ImageFolder
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve
```

[2]:
```
data_transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

data=ImageFolder("./archive/Dataset/", transform=data_transform)
```

[3]:
```
c0=0    # mild
c1=0    # moderate
c2=0    # non
c3=0    # very mild
for i,d in enumerate(data):
    if data[i][1]==0:   c0+=1
    elif data[i][1]==1: c1+=1
    elif data[i][1]==2: c2+=1
    elif data[i][1]==3: c3+=1
print(c0,c1,c2,c3)
```

```
896 64 3200 2240
```

[4]:
```
n=len(data)
n_test=int(0.2*n)   # 20% for test
train_data,test_data=random_split(data,[n-n_test,n_test],torch.Generator().
 ↪manual_seed(42))
```

[5]:
```
c0=0    # mild
c1=0    # moderate
```

```
c2=0    # non
c3=0    # very mild
for i,d in enumerate(test_data):
    if test_data[i][1]==0:   c0+=1
    elif test_data[i][1]==1: c1+=1
    elif test_data[i][1]==2: c2+=1
    elif test_data[i][1]==3: c3+=1
print(c0,c1,c2,c3)
```

```
179 13 634 454
```

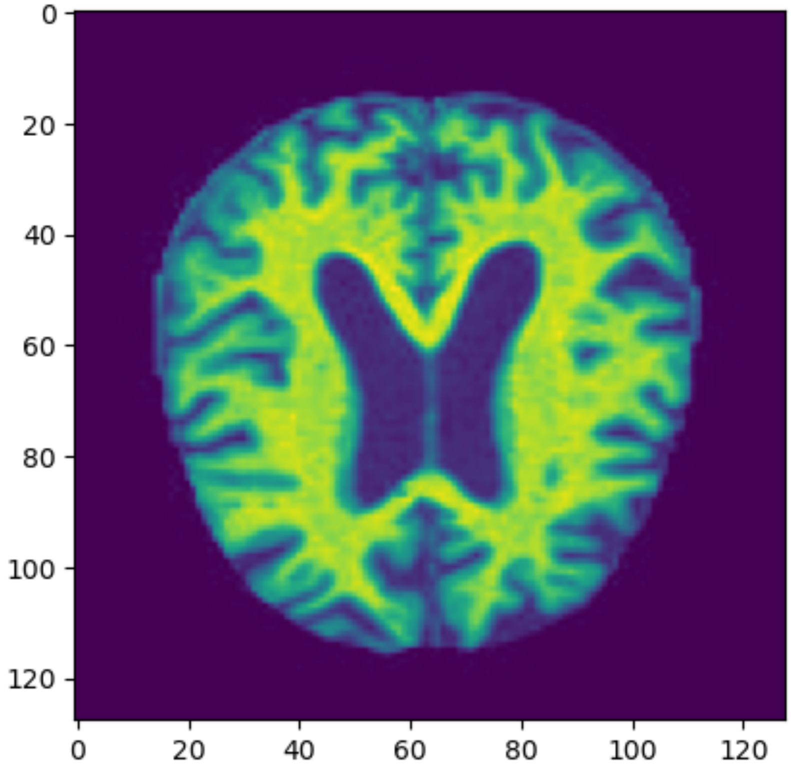
[6]:
```
trainloader=DataLoader(train_data,batch_size=64,shuffle=True)
testloader=DataLoader(test_data,batch_size=64,shuffle=False)
```

[7]:
```
for i,data in enumerate(trainloader):
    imgs, targets=data
    if i<10: print(imgs.shape)
    else: break
```

```
torch.Size([64, 3, 128, 128])
torch.Size([64, 3, 128, 128])
torch.Size([64, 3, 128, 128])
torch.Size([64, 3, 128, 128])
torch.Size([64, 3, 128, 128])
torch.Size([64, 3, 128, 128])
torch.Size([64, 3, 128, 128])
torch.Size([64, 3, 128, 128])
torch.Size([64, 3, 128, 128])
torch.Size([64, 3, 128, 128])
```

[8]:
```
from PIL import Image
fig=Image.open("./archive/Dataset/Mild_Demented/mild.jpg")
plt.imshow(fig)
print(fig)
```

```
<PIL.JpegImagePlugin.JpegImageFile image mode=L size=128x128 at 0x12E93AB3D08>
```



[ ]:

[9]:
```
train_data_cls=[]
train_label_cls=[]
test_data_cls=[]
test_label_cls=[]
for i,data in enumerate(train_data):
    train_data_cls.append(np.array(data[0].view(1,-1)[0]))
    train_label_cls.append(data[1])

for i,data in enumerate(test_data):
    test_data_cls.append(np.array(data[0].view(1,-1)[0]))
    test_label_cls.append(data[1])

train_data_cls=np.array(train_data_cls)
test_data_cls=np.array(test_data_cls)
```

[10]:
```
n_train=len(train_data_cls)
n_test=len(test_data_cls)
print(n_train,n_test)
```

```
5120 1280
```

[26]:
```
data_cls=np.concatenate((train_data_cls,test_data_cls),axis=0)
label_cls=np.concatenate((train_label_cls,test_label_cls),axis=0)
```

1 KNN

[12]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(train_data_cls,train_label_cls)
knn_pred=knn.predict(test_data_cls)
print(np.mean(knn_pred==test_label_cls))
```

```
0.971875
```

[31]:
```
knn2=KNeighborsClassifier(n_neighbors=10)
knn2.fit(train_data_cls,train_label_cls)
knn2_pred=knn2.predict(test_data_cls)
print(np.mean(knn2_pred==test_label_cls))
```

```
0.89296875
```

2 Naive Bayes

[13]:
```
train_c0=896-179
train_c1=64-13
train_c2=3200-634
train_c3=2240-454
total=train_c0+train_c1+train_c2+train_c3
```

[14]:
```
from sklearn.naive_bayes import GaussianNB
nb=GaussianNB(priors=[train_c0/total,train_c1/total,train_c2/total,train_c3/
 ↪total])
nb.fit(train_data_cls,train_label_cls)
nb_pred=nb.predict(test_data_cls)
print(np.mean(nb_pred==test_label_cls))
```

```
0.4859375
```

[29]:
```
nb2=GaussianNB(priors=[0.25,0.25,0.25,0.25])
nb2.fit(train_data_cls,train_label_cls)
nb2_pred=nb2.predict(test_data_cls)
print(np.mean(nb2_pred==test_label_cls))
```

```
0.4859375
```

3 Logistic Regression

[15]:
```
from sklearn.linear_model import LogisticRegression
log=LogisticRegression()
log.fit(train_data_cls,train_label_cls)
log_pred=log.predict(test_data_cls)
print(np.mean(log_pred==test_label_cls))
```

```
0.89375

c:\ProgramData\Anaconda3\envs\pytorch\lib\site-
packages\sklearn\linear_model\_logistic.py:818: ConvergenceWarning: lbfgs failed
to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

4 Decision Tree

[16]:
```
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
dt.fit(train_data_cls,train_label_cls)
dt_pred=dt.predict(test_data_cls)
print(np.mean(dt_pred==test_label_cls))
```

```
0.684375
```

5 Random Forest

[17]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
rf.fit(train_data_cls,train_label_cls)
rf_pred=rf.predict(test_data_cls)
print(np.mean(rf_pred==test_label_cls))
```

```
0.921875
```

6 SVM

[18]:
```
from sklearn.svm import SVC
svc=SVC()
svc.fit(train_data_cls,train_label_cls)
```

```
svc_pred=svc.predict(test_data_cls)
print(np.mean(svc_pred==test_label_cls))
```

0.7671875

[20]:
```
from sklearn.svm import SVC

svc_linear=SVC(kernel='linear')
svc_linear.fit(train_data_cls,train_label_cls)
svc_linear_pred=svc.predict(test_data_cls)
print(np.mean(svc_linear_pred==test_label_cls))
```

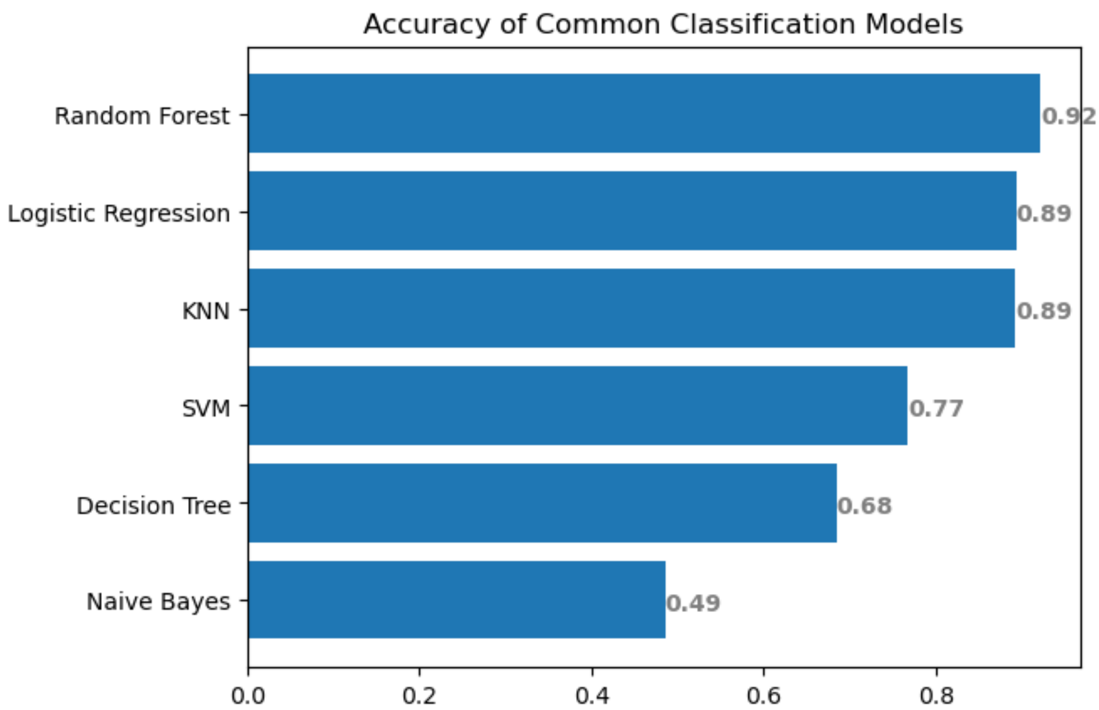
0.7671875

7 Comparison

[37]:
```
acc=[knn2_pred,nb_pred,log_pred,dt_pred,rf_pred,svc_pred]
acc=[np.mean(i==test_label_cls) for i in acc]
labels=["KNN","Naive Bayes","Logistic Regression","Decision Tree", "Random␣
↪Forest","SVM"]
print(acc)
```

[0.89296875, 0.4859375, 0.89375, 0.684375, 0.921875, 0.7671875]

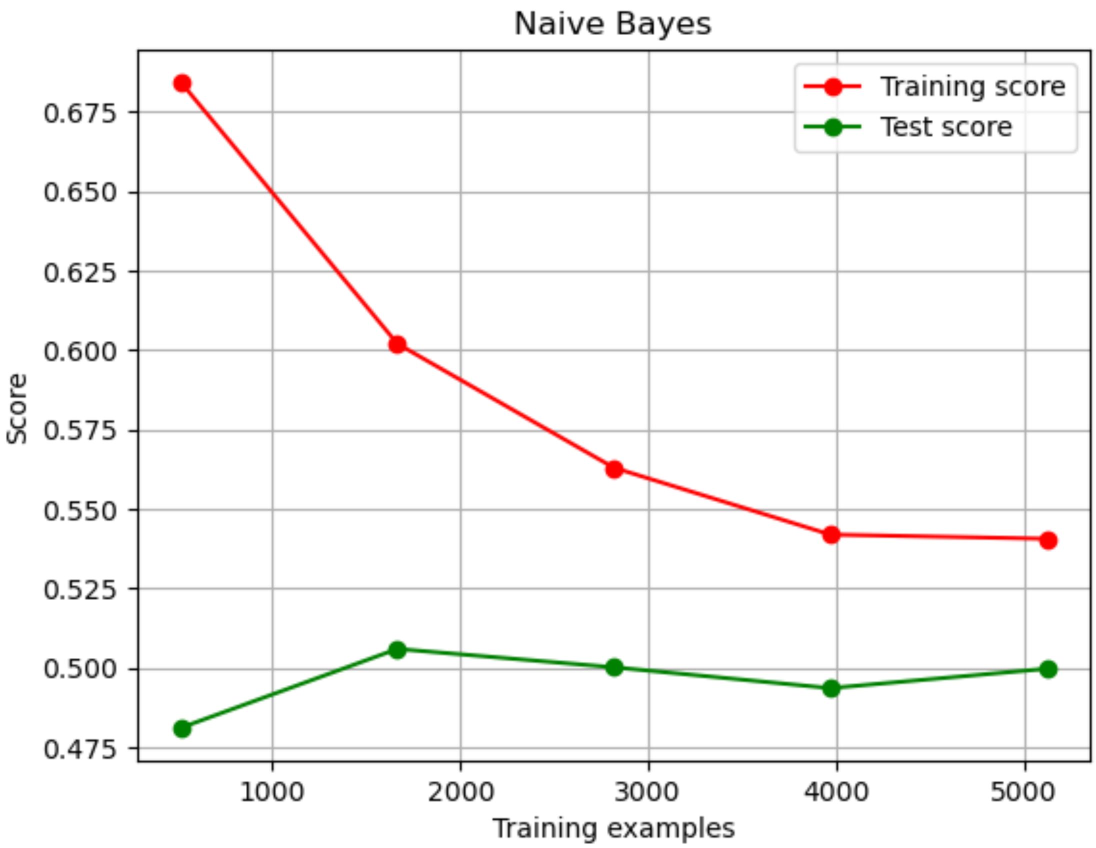
[46]:
```
fig, ax = plt.subplots()
indexes = np.argsort(-np.array(acc))
acc_sorted = [acc[i] for i in indexes]
labels_sorted = [labels[i] for i in indexes]
y_pos = np.arange(len(labels))
ax.barh(y_pos,acc_sorted,align='center')
ax.set_yticks(y_pos)
ax.set_yticklabels(labels_sorted)
ax.invert_yaxis()
ax.set_title("Accuracy of Common Classification Models")
for i in ax.patches:
    plt.text(i.get_width(), i.get_y()+0.5, str(round((i.get_width()), 2)),␣
↪fontsize=10, fontweight='bold', color='grey')
```



8 Learning Curve

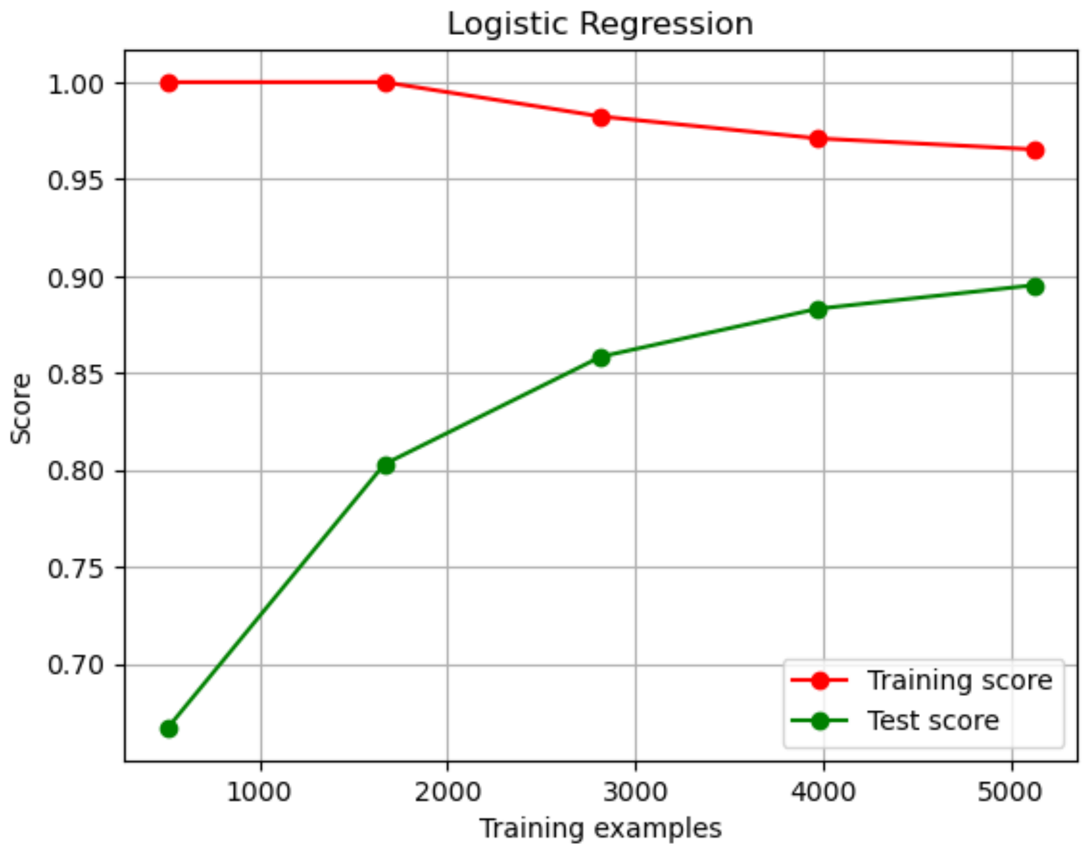
[27]:
```
def plot_learning_curve(estimator,title, X, y,ax,ylim=None,cv=None,n_jobs=None):
    train_sizes, train_scores, test_scores = learning_curve(estimator, X,␣
↪y,cv=cv,n_jobs=n_jobs)
    ax.set_title(title)
    if ylim is not None:
        ax.set_ylim(*ylim)
    ax.set_xlabel("Training examples")
    ax.set_ylabel("Score")
    ax.grid()
    ax.plot(train_sizes, np.mean(train_scores, axis=1), 'o-'
            , color="r",label="Training score")
    ax.plot(train_sizes, np.mean(test_scores, axis=1), 'o-'
            , color="g",label="Test score")
    ax.legend(loc="best")
    return ax

plt.figure()
plot_learning_curve(nb, "Naive Bayes", data_cls, label_cls, ax=plt.
↪gca(),n_jobs=4, cv=5);
```



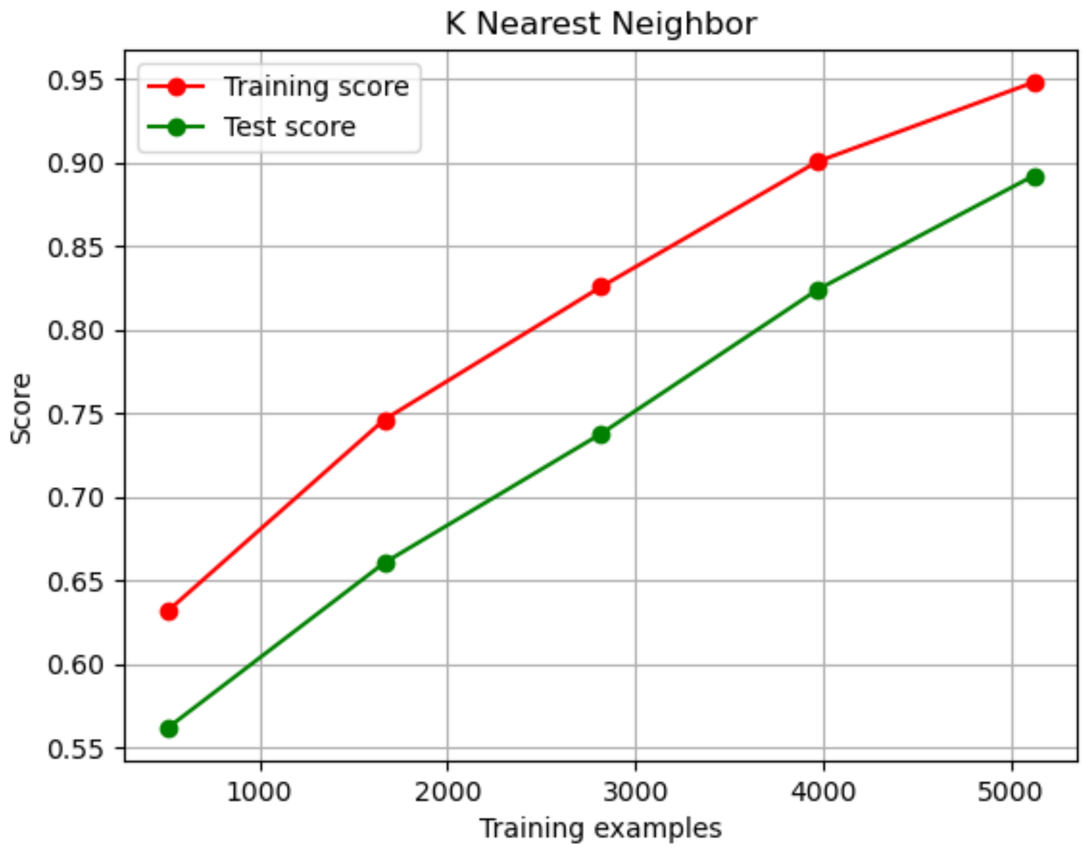
[28]:
```
plt.figure()
plot_learning_curve(log,"Logistic Regression",data_cls,label_cls,ax=plt.
↪gca(),n_jobs=4,cv=5)
```

[28]:
```
<AxesSubplot:title={'center':'Logistic Regression'}, xlabel='Training examples',
ylabel='Score'>
```



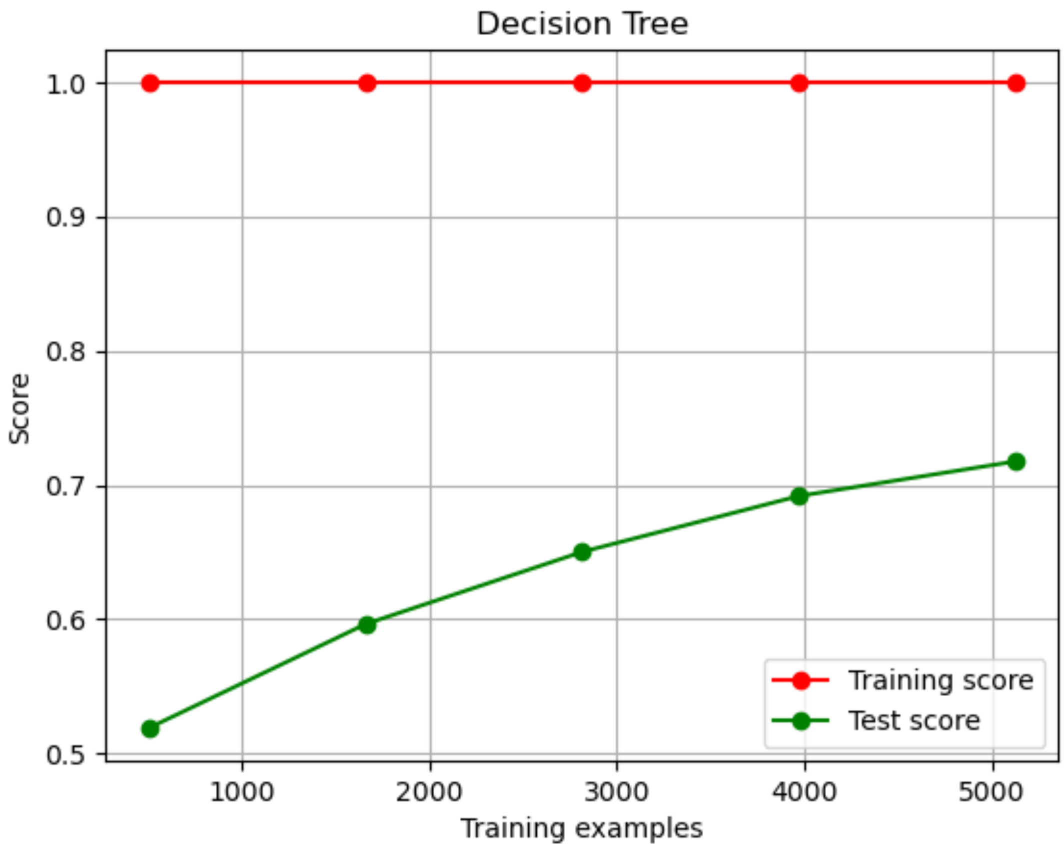
[32]:
```
plt.figure()
plot_learning_curve(knn2,"K Nearest Neighbor",data_cls,label_cls,ax=plt.
↪gca(),n_jobs=4,cv=5)
```

[32]:
```
<AxesSubplot:title={'center':'K Nearest Neighbor'}, xlabel='Training examples',
ylabel='Score'>
```



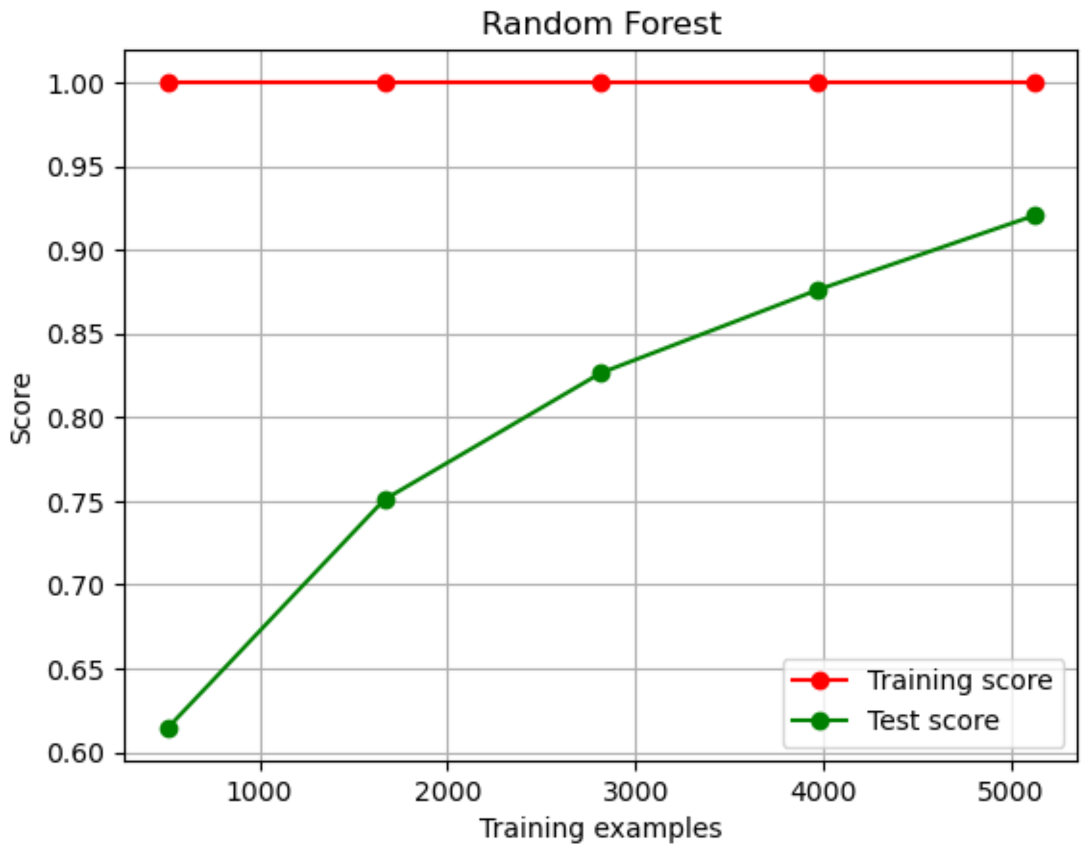
[33]:
```
plt.figure()
plot_learning_curve(dt,"Decision Tree",data_cls,label_cls,ax=plt.
 ↪gca(),n_jobs=4,cv=5)
```

[33]:
```
<AxesSubplot:title={'center':'Decision Tree'}, xlabel='Training examples',
ylabel='Score'>
```



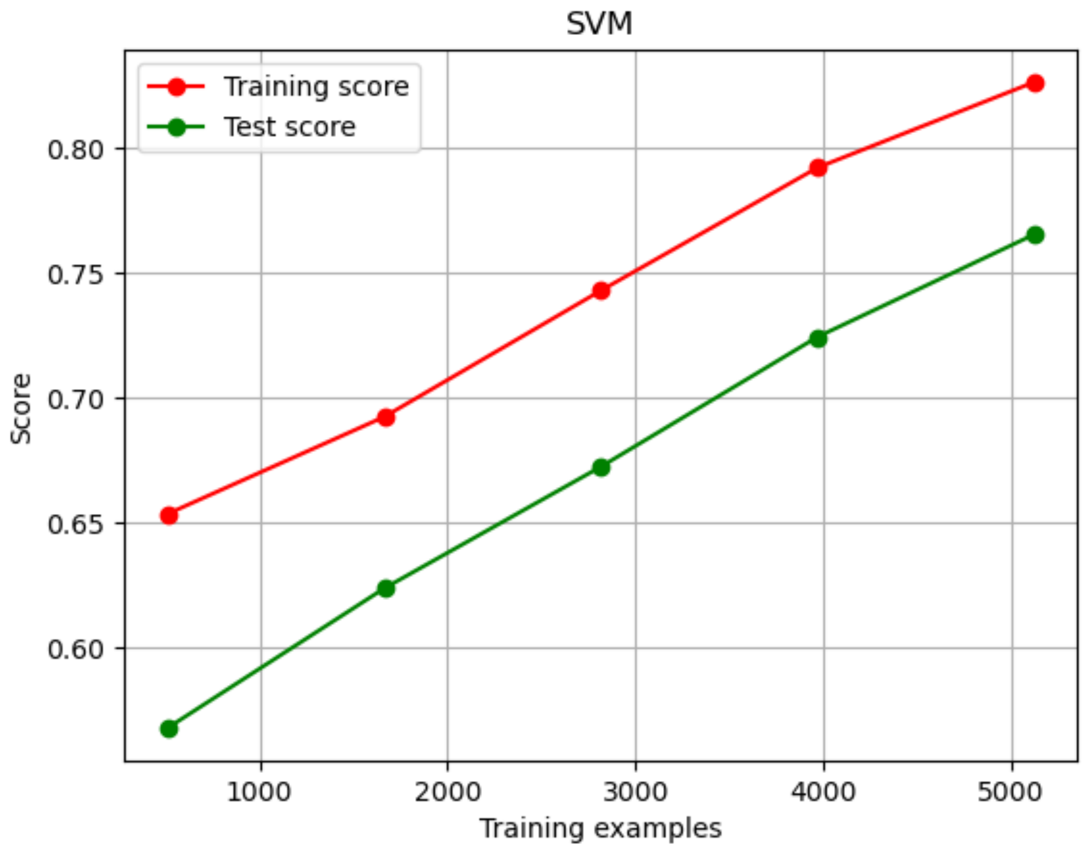
[34]:
```
plt.figure()
plot_learning_curve(rf,"Random Forest",data_cls,label_cls,ax=plt.
↪gca(),n_jobs=4,cv=5)
```

[34]: <AxesSubplot:title={'center':'Random Forest'}, xlabel='Training examples', ylabel='Score'>




```
[47]: plt.figure()
      plot_learning_curve(svc,"SVM",data_cls,label_cls,ax=plt.gca(),n_jobs=4,cv=5)
```

[47]: <AxesSubplot:title={'center':'SVM'}, xlabel='Training examples', ylabel='Score'>


SVM
Training score
Test score
0.80
0.75
0.70
0.65
0.60
Score
1000
2000
3000
4000
5000
Training examples

```
[ ]:
```

CNN Model

December 16, 2022

[18]:
```
import numpy as np
import sklearn
import torch
import torchvision
from torchvision import transforms
from torch.utils.data import DataLoader, random_split
from sklearn.model_selection import train_test_split
from torchvision.datasets import ImageFolder
import matplotlib.pyplot as plt
```

[19]:
```
data_transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))  # Any Reason for␣
 ↪this?
])

data=ImageFolder("./archive/Dataset/", transform=data_transform)
```

[20]:
```
# c0=0    # mild
# c1=0    # moderate
# c2=0    # non
# c3=0    # very mild
# for i,d in enumerate(data):
#     if data[i][1]==0:   c0+=1
#     elif data[i][1]==1: c1+=1
#     elif data[i][1]==2: c2+=1
#     elif data[i][1]==3: c3+=1
# print(c0,c1,c2,c3)
```

[21]:
```
n=len(data)
n_test=int(0.2*n)   # 20% for test
train_data,test_data=random_split(data,[n-n_test,n_test],torch.Generator().
 ↪manual_seed(42))
```

[22]:
```
# c0=0    # mild
# c1=0    # moderate
# c2=0    # non
# c3=0    # very mild
```

```
# for i,d in enumerate(test_data):
#     if test_data[i][1]==0:   c0+=1
#     elif test_data[i][1]==1: c1+=1
#     elif test_data[i][1]==2: c2+=1
#     elif test_data[i][1]==3: c3+=1
# print(c0,c1,c2,c3)
```

[23]:
```
trainloader=DataLoader(train_data,batch_size=128,drop_last=False,shuffle=True)
testloader=DataLoader(test_data,batch_size=128,drop_last=False,shuffle=False)
```

[24]:
```
# for i,data in enumerate(trainloader):
#     imgs, targets=data
#     if i<10: print(imgs.shape)
#     else: break
```

[25]:
```
# from PIL import Image
# fig=Image.open("./archive/Dataset/Mild_Demented/mild.jpg")
# plt.imshow(fig)
# print(fig)
```

[26]:
```
import torch.nn.functional as F

class ConvNet(torch.nn.Module):
    def __init__(self):
        super(ConvNet, self).__init__()
        # Set for convolution operation
        self.conv1 = torch.nn.Sequential(
            torch.nn.Conv2d(3, 16, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
        )
        self.conv2 = torch.nn.Sequential(
            torch.nn.Conv2d(16, 32, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
        )
        self.conv3 = torch.nn.Sequential(
            torch.nn.Conv2d(32, 64, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
        )

        self.conv4 = torch.nn.Sequential(
            torch.nn.Conv2d(64, 128, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
        )
```

```
        self.dp = torch.nn.Dropout(p=0.5)

        self.fc1 = torch.nn.Sequential(
            torch.nn.Linear(128*8*8, 32),
            torch.nn.ReLU()
        )
        self.fc2 = torch.nn.Linear(32, 4)

    def forward(self, x):
        # Three-layer convolutional network (Conv -> ReLU -> MaxPool)
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.conv3(x)
        x = self.conv4(x)
        x = self.dp(x)
        x = x.view(-1, 128*8*8)
        x = self.fc1(x) # Fully connected layer -> ReLU
        x = self.fc2(x)
        out = F.log_softmax(x, dim=1) # Softmax probability
        return out

net_cpu = ConvNet()
net_gpu = net_cpu.cuda()
```

[27]:
```
from torch import optim
from torch.utils.tensorboard import SummaryWriter

summaryWriter = SummaryWriter("logs/lyf_cnn_3")

optimizer = optim.Adam(
    net_gpu.parameters(),
    lr = 0.001,
    betas = (0.9, 0.999),
    eps = 1e-08,
    weight_decay = 0,
    amsgrad = False
)

loss_func = torch.nn.CrossEntropyLoss()

for epoch in range(50):
    running_loss_train = 0
    for i, data in enumerate(trainloader, 0):
        inputs_cpu, targets_cpu = data
        inputs_gpu = inputs_cpu.cuda()
        targets_gpu = targets_cpu.cuda()
```

```
        optimizer.zero_grad()
        outputs_gpu = net_gpu.train()(inputs_gpu)
        loss = loss_func(outputs_gpu, targets_gpu)
        running_loss_train += loss.item()
        loss.backward()
        optimizer.step()
        if i % 8 == 7:
            print('Train Epoch: %d [%d/5000] Loss: %.6f' %(epoch, i*64, loss.
 ↪item()))
    running_loss_train /= len(trainloader)
    print(running_loss_train)
    summaryWriter.add_scalar("loss", running_loss_train, epoch)

    # Step 4 Predict
    correct = 0
    total = 0
    running_loss = 0
    for data in testloader:
        images_cpu, targets_cpu = data
        images_gpu = images_cpu.cuda()
        targets_gpu = targets_cpu.cuda()
        outputs_gpu = net_gpu.eval()(images_gpu)
        _, predicted = torch.max(outputs_gpu, 1)
        loss = loss_func(outputs_gpu, targets_gpu)
        total += targets_gpu.size(0)
        running_loss += loss.item()
        correct += (predicted == targets_gpu).sum().item()

    running_loss = running_loss / 10000

    print('Test set: Average loss: %.4f, Accuracy: %d/10000 (%d%%)'␣
 ↪%(running_loss, correct, correct*100/total))
    summaryWriter.add_scalar("accuracy", correct/total, epoch)

print('Train and predict complete!')
```

```
Train Epoch: 0 [448/5000] Loss: 1.234264
Train Epoch: 0 [960/5000] Loss: 1.192123
Train Epoch: 0 [1472/5000] Loss: 1.028847
Train Epoch: 0 [1984/5000] Loss: 1.158864
Train Epoch: 0 [2496/5000] Loss: 0.968348
1.176887346804142
Test set: Average loss: 0.0010, Accuracy: 651/10000 (50%)
Train Epoch: 1 [448/5000] Loss: 1.055168
Train Epoch: 1 [960/5000] Loss: 0.980253
Train Epoch: 1 [1472/5000] Loss: 0.993573
Train Epoch: 1 [1984/5000] Loss: 1.018553
```

```
Train Epoch: 1 [2496/5000] Loss: 0.973411
1.0067195862531662
Test set: Average loss: 0.0010, Accuracy: 656/10000 (51%)
Train Epoch: 2 [448/5000] Loss: 0.977366
Train Epoch: 2 [960/5000] Loss: 1.022532
Train Epoch: 2 [1472/5000] Loss: 0.884617
Train Epoch: 2 [1984/5000] Loss: 1.027266
Train Epoch: 2 [2496/5000] Loss: 0.961592
0.9522234946489334
Test set: Average loss: 0.0009, Accuracy: 702/10000 (54%)
Train Epoch: 3 [448/5000] Loss: 0.939151
Train Epoch: 3 [960/5000] Loss: 0.932697
Train Epoch: 3 [1472/5000] Loss: 0.946233
Train Epoch: 3 [1984/5000] Loss: 0.860118
Train Epoch: 3 [2496/5000] Loss: 0.873317
0.9258226558566094
Test set: Average loss: 0.0009, Accuracy: 670/10000 (52%)
Train Epoch: 4 [448/5000] Loss: 0.777402
Train Epoch: 4 [960/5000] Loss: 0.938945
Train Epoch: 4 [1472/5000] Loss: 1.006959
Train Epoch: 4 [1984/5000] Loss: 0.916795
Train Epoch: 4 [2496/5000] Loss: 0.890977
0.9201558992266655
Test set: Average loss: 0.0009, Accuracy: 737/10000 (57%)
Train Epoch: 5 [448/5000] Loss: 0.907668
Train Epoch: 5 [960/5000] Loss: 0.877484
Train Epoch: 5 [1472/5000] Loss: 0.936767
Train Epoch: 5 [1984/5000] Loss: 0.973712
Train Epoch: 5 [2496/5000] Loss: 0.879805
0.8996330931782722
Test set: Average loss: 0.0010, Accuracy: 637/10000 (49%)
Train Epoch: 6 [448/5000] Loss: 0.909240
Train Epoch: 6 [960/5000] Loss: 0.852635
Train Epoch: 6 [1472/5000] Loss: 0.936893
Train Epoch: 6 [1984/5000] Loss: 0.839148
Train Epoch: 6 [2496/5000] Loss: 0.933353
0.9009059056639671
Test set: Average loss: 0.0009, Accuracy: 757/10000 (59%)
Train Epoch: 7 [448/5000] Loss: 0.957539
Train Epoch: 7 [960/5000] Loss: 0.906467
Train Epoch: 7 [1472/5000] Loss: 0.893200
Train Epoch: 7 [1984/5000] Loss: 0.855887
Train Epoch: 7 [2496/5000] Loss: 0.884819
0.8666952133178711
Test set: Average loss: 0.0009, Accuracy: 760/10000 (59%)
Train Epoch: 8 [448/5000] Loss: 0.750773
Train Epoch: 8 [960/5000] Loss: 0.957538
Train Epoch: 8 [1472/5000] Loss: 0.809177
```

```
Train Epoch: 8 [1984/5000] Loss: 0.844509
Train Epoch: 8 [2496/5000] Loss: 0.952795
0.8555545806884766
Test set: Average loss: 0.0009, Accuracy: 760/10000 (59%)
Train Epoch: 9 [448/5000] Loss: 0.792276
Train Epoch: 9 [960/5000] Loss: 0.868337
Train Epoch: 9 [1472/5000] Loss: 0.800438
Train Epoch: 9 [1984/5000] Loss: 0.810928
Train Epoch: 9 [2496/5000] Loss: 0.875803
0.8360996454954147
Test set: Average loss: 0.0009, Accuracy: 741/10000 (57%)
Train Epoch: 10 [448/5000] Loss: 0.825281
Train Epoch: 10 [960/5000] Loss: 0.803332
Train Epoch: 10 [1472/5000] Loss: 0.848510
Train Epoch: 10 [1984/5000] Loss: 0.935005
Train Epoch: 10 [2496/5000] Loss: 0.832994
0.826998870074749
Test set: Average loss: 0.0009, Accuracy: 776/10000 (60%)
Train Epoch: 11 [448/5000] Loss: 0.832698
Train Epoch: 11 [960/5000] Loss: 0.820723
Train Epoch: 11 [1472/5000] Loss: 0.739659
Train Epoch: 11 [1984/5000] Loss: 0.850299
Train Epoch: 11 [2496/5000] Loss: 0.760378
0.811349019408226
Test set: Average loss: 0.0008, Accuracy: 812/10000 (63%)
Train Epoch: 12 [448/5000] Loss: 0.750534
Train Epoch: 12 [960/5000] Loss: 0.716712
Train Epoch: 12 [1472/5000] Loss: 0.690921
Train Epoch: 12 [1984/5000] Loss: 0.635195
Train Epoch: 12 [2496/5000] Loss: 0.769534
0.7572867766022682
Test set: Average loss: 0.0008, Accuracy: 846/10000 (66%)
Train Epoch: 13 [448/5000] Loss: 0.754180
Train Epoch: 13 [960/5000] Loss: 0.734480
Train Epoch: 13 [1472/5000] Loss: 0.622708
Train Epoch: 13 [1984/5000] Loss: 0.744337
Train Epoch: 13 [2496/5000] Loss: 0.707321
0.7231426939368248
Test set: Average loss: 0.0007, Accuracy: 900/10000 (70%)
Train Epoch: 14 [448/5000] Loss: 0.734310
Train Epoch: 14 [960/5000] Loss: 0.678680
Train Epoch: 14 [1472/5000] Loss: 0.551226
Train Epoch: 14 [1984/5000] Loss: 0.693768
Train Epoch: 14 [2496/5000] Loss: 0.635008
0.6345520570874215
Test set: Average loss: 0.0006, Accuracy: 941/10000 (73%)
Train Epoch: 15 [448/5000] Loss: 0.483630
Train Epoch: 15 [960/5000] Loss: 0.609912
```

```
Train Epoch: 15 [1472/5000] Loss: 0.590572
Train Epoch: 15 [1984/5000] Loss: 0.676287
Train Epoch: 15 [2496/5000] Loss: 0.551793
0.5830947093665599
Test set: Average loss: 0.0006, Accuracy: 968/10000 (75%)
Train Epoch: 16 [448/5000] Loss: 0.508013
Train Epoch: 16 [960/5000] Loss: 0.477788
Train Epoch: 16 [1472/5000] Loss: 0.372923
Train Epoch: 16 [1984/5000] Loss: 0.480078
Train Epoch: 16 [2496/5000] Loss: 0.454914
0.504715372622013
Test set: Average loss: 0.0005, Accuracy: 1040/10000 (81%)
Train Epoch: 17 [448/5000] Loss: 0.398368
Train Epoch: 17 [960/5000] Loss: 0.435356
Train Epoch: 17 [1472/5000] Loss: 0.443595
Train Epoch: 17 [1984/5000] Loss: 0.392702
Train Epoch: 17 [2496/5000] Loss: 0.504588
0.4229027919471264
Test set: Average loss: 0.0005, Accuracy: 1040/10000 (81%)
Train Epoch: 18 [448/5000] Loss: 0.319731
Train Epoch: 18 [960/5000] Loss: 0.348549
Train Epoch: 18 [1472/5000] Loss: 0.395762
Train Epoch: 18 [1984/5000] Loss: 0.452708
Train Epoch: 18 [2496/5000] Loss: 0.358595
0.36359197050333025
Test set: Average loss: 0.0004, Accuracy: 1103/10000 (86%)
Train Epoch: 19 [448/5000] Loss: 0.450720
Train Epoch: 19 [960/5000] Loss: 0.271245
Train Epoch: 19 [1472/5000] Loss: 0.345136
Train Epoch: 19 [1984/5000] Loss: 0.289784
Train Epoch: 19 [2496/5000] Loss: 0.344468
0.32578014098107816
Test set: Average loss: 0.0003, Accuracy: 1107/10000 (86%)
Train Epoch: 20 [448/5000] Loss: 0.280024
Train Epoch: 20 [960/5000] Loss: 0.309936
Train Epoch: 20 [1472/5000] Loss: 0.269713
Train Epoch: 20 [1984/5000] Loss: 0.273366
Train Epoch: 20 [2496/5000] Loss: 0.310289
0.26586394011974335
Test set: Average loss: 0.0004, Accuracy: 1109/10000 (86%)
Train Epoch: 21 [448/5000] Loss: 0.367013
Train Epoch: 21 [960/5000] Loss: 0.224212
Train Epoch: 21 [1472/5000] Loss: 0.163470
Train Epoch: 21 [1984/5000] Loss: 0.330192
Train Epoch: 21 [2496/5000] Loss: 0.309836
0.2253912039101124
Test set: Average loss: 0.0002, Accuracy: 1181/10000 (92%)
Train Epoch: 22 [448/5000] Loss: 0.173861
```

```
Train Epoch: 22 [960/5000] Loss: 0.186253
Train Epoch: 22 [1472/5000] Loss: 0.135730
Train Epoch: 22 [1984/5000] Loss: 0.209299
Train Epoch: 22 [2496/5000] Loss: 0.179331
0.1902927180752158
Test set: Average loss: 0.0002, Accuracy: 1190/10000 (92%)
Train Epoch: 23 [448/5000] Loss: 0.161341
Train Epoch: 23 [960/5000] Loss: 0.158263
Train Epoch: 23 [1472/5000] Loss: 0.096367
Train Epoch: 23 [1984/5000] Loss: 0.127988
Train Epoch: 23 [2496/5000] Loss: 0.142110
0.14379832353442906
Test set: Average loss: 0.0002, Accuracy: 1191/10000 (93%)
Train Epoch: 24 [448/5000] Loss: 0.110942
Train Epoch: 24 [960/5000] Loss: 0.094547
Train Epoch: 24 [1472/5000] Loss: 0.108469
Train Epoch: 24 [1984/5000] Loss: 0.120813
Train Epoch: 24 [2496/5000] Loss: 0.167865
0.14141426105052232
Test set: Average loss: 0.0002, Accuracy: 1177/10000 (91%)
Train Epoch: 25 [448/5000] Loss: 0.100436
Train Epoch: 25 [960/5000] Loss: 0.111592
Train Epoch: 25 [1472/5000] Loss: 0.074073
Train Epoch: 25 [1984/5000] Loss: 0.155453
Train Epoch: 25 [2496/5000] Loss: 0.091681
0.11854634564369917
Test set: Average loss: 0.0002, Accuracy: 1206/10000 (94%)
Train Epoch: 26 [448/5000] Loss: 0.043842
Train Epoch: 26 [960/5000] Loss: 0.079522
Train Epoch: 26 [1472/5000] Loss: 0.084302
Train Epoch: 26 [1984/5000] Loss: 0.116561
Train Epoch: 26 [2496/5000] Loss: 0.082015
0.09839530503377318
Test set: Average loss: 0.0002, Accuracy: 1207/10000 (94%)
Train Epoch: 27 [448/5000] Loss: 0.087024
Train Epoch: 27 [960/5000] Loss: 0.101705
Train Epoch: 27 [1472/5000] Loss: 0.033478
Train Epoch: 27 [1984/5000] Loss: 0.103518
Train Epoch: 27 [2496/5000] Loss: 0.079083
0.09506505713798105
Test set: Average loss: 0.0002, Accuracy: 1206/10000 (94%)
Train Epoch: 28 [448/5000] Loss: 0.143516
Train Epoch: 28 [960/5000] Loss: 0.082981
Train Epoch: 28 [1472/5000] Loss: 0.082114
Train Epoch: 28 [1984/5000] Loss: 0.147961
Train Epoch: 28 [2496/5000] Loss: 0.097162
0.09331057369709014
Test set: Average loss: 0.0001, Accuracy: 1239/10000 (96%)
```

```
Train Epoch: 29 [448/5000] Loss: 0.060058
Train Epoch: 29 [960/5000] Loss: 0.116239
Train Epoch: 29 [1472/5000] Loss: 0.077780
Train Epoch: 29 [1984/5000] Loss: 0.058171
Train Epoch: 29 [2496/5000] Loss: 0.095297
0.07924023657105864
Test set: Average loss: 0.0001, Accuracy: 1231/10000 (96%)
Train Epoch: 30 [448/5000] Loss: 0.059153
Train Epoch: 30 [960/5000] Loss: 0.046657
Train Epoch: 30 [1472/5000] Loss: 0.081288
Train Epoch: 30 [1984/5000] Loss: 0.118025
Train Epoch: 30 [2496/5000] Loss: 0.087080
0.0692436930257827
Test set: Average loss: 0.0001, Accuracy: 1224/10000 (95%)
Train Epoch: 31 [448/5000] Loss: 0.051034
Train Epoch: 31 [960/5000] Loss: 0.085172
Train Epoch: 31 [1472/5000] Loss: 0.030492
Train Epoch: 31 [1984/5000] Loss: 0.104602
Train Epoch: 31 [2496/5000] Loss: 0.098336
0.06956297219730914
Test set: Average loss: 0.0001, Accuracy: 1238/10000 (96%)
Train Epoch: 32 [448/5000] Loss: 0.072822
Train Epoch: 32 [960/5000] Loss: 0.005907
Train Epoch: 32 [1472/5000] Loss: 0.063191
Train Epoch: 32 [1984/5000] Loss: 0.051788
Train Epoch: 32 [2496/5000] Loss: 0.028525
0.0600310274399817
Test set: Average loss: 0.0001, Accuracy: 1249/10000 (97%)
Train Epoch: 33 [448/5000] Loss: 0.040661
Train Epoch: 33 [960/5000] Loss: 0.030763
Train Epoch: 33 [1472/5000] Loss: 0.109777
Train Epoch: 33 [1984/5000] Loss: 0.016732
Train Epoch: 33 [2496/5000] Loss: 0.043012
0.059706470975652334
Test set: Average loss: 0.0001, Accuracy: 1242/10000 (97%)
Train Epoch: 34 [448/5000] Loss: 0.028411
Train Epoch: 34 [960/5000] Loss: 0.010468
Train Epoch: 34 [1472/5000] Loss: 0.059043
Train Epoch: 34 [1984/5000] Loss: 0.102021
Train Epoch: 34 [2496/5000] Loss: 0.041372
0.043871473707258704
Test set: Average loss: 0.0001, Accuracy: 1242/10000 (97%)
Train Epoch: 35 [448/5000] Loss: 0.025873
Train Epoch: 35 [960/5000] Loss: 0.036909
Train Epoch: 35 [1472/5000] Loss: 0.044132
Train Epoch: 35 [1984/5000] Loss: 0.014974
Train Epoch: 35 [2496/5000] Loss: 0.061383
0.045211103605106474
```

```
Test set: Average loss: 0.0001, Accuracy: 1252/10000 (97%)
Train Epoch: 36 [448/5000] Loss: 0.007037
Train Epoch: 36 [960/5000] Loss: 0.075854
Train Epoch: 36 [1472/5000] Loss: 0.026957
Train Epoch: 36 [1984/5000] Loss: 0.030669
Train Epoch: 36 [2496/5000] Loss: 0.022871
0.03289215210825205
Test set: Average loss: 0.0001, Accuracy: 1243/10000 (97%)
Train Epoch: 37 [448/5000] Loss: 0.045723
Train Epoch: 37 [960/5000] Loss: 0.058530
Train Epoch: 37 [1472/5000] Loss: 0.046558
Train Epoch: 37 [1984/5000] Loss: 0.085211
Train Epoch: 37 [2496/5000] Loss: 0.011027
0.038931540213525295
Test set: Average loss: 0.0001, Accuracy: 1251/10000 (97%)
Train Epoch: 38 [448/5000] Loss: 0.022292
Train Epoch: 38 [960/5000] Loss: 0.009152
Train Epoch: 38 [1472/5000] Loss: 0.068549
Train Epoch: 38 [1984/5000] Loss: 0.021246
Train Epoch: 38 [2496/5000] Loss: 0.020517
0.040106918173842133
Test set: Average loss: 0.0001, Accuracy: 1244/10000 (97%)
Train Epoch: 39 [448/5000] Loss: 0.008950
Train Epoch: 39 [960/5000] Loss: 0.031474
Train Epoch: 39 [1472/5000] Loss: 0.077165
Train Epoch: 39 [1984/5000] Loss: 0.028147
Train Epoch: 39 [2496/5000] Loss: 0.074615
0.039570213132537904
Test set: Average loss: 0.0001, Accuracy: 1251/10000 (97%)
Train Epoch: 40 [448/5000] Loss: 0.029533
Train Epoch: 40 [960/5000] Loss: 0.055235
Train Epoch: 40 [1472/5000] Loss: 0.039130
Train Epoch: 40 [1984/5000] Loss: 0.040581
Train Epoch: 40 [2496/5000] Loss: 0.036086
0.03883770573884249
Test set: Average loss: 0.0001, Accuracy: 1250/10000 (97%)
Train Epoch: 41 [448/5000] Loss: 0.005679
Train Epoch: 41 [960/5000] Loss: 0.050045
Train Epoch: 41 [1472/5000] Loss: 0.026413
Train Epoch: 41 [1984/5000] Loss: 0.041197
Train Epoch: 41 [2496/5000] Loss: 0.041869
0.03722620097687468
Test set: Average loss: 0.0001, Accuracy: 1252/10000 (97%)
Train Epoch: 42 [448/5000] Loss: 0.021655
Train Epoch: 42 [960/5000] Loss: 0.013844
Train Epoch: 42 [1472/5000] Loss: 0.027733
Train Epoch: 42 [1984/5000] Loss: 0.008161
Train Epoch: 42 [2496/5000] Loss: 0.033062
```

```
0.031211514491587877
Test set: Average loss: 0.0001, Accuracy: 1242/10000 (97%)
Train Epoch: 43 [448/5000] Loss: 0.019223
Train Epoch: 43 [960/5000] Loss: 0.015118
Train Epoch: 43 [1472/5000] Loss: 0.021714
Train Epoch: 43 [1984/5000] Loss: 0.095972
Train Epoch: 43 [2496/5000] Loss: 0.014421
0.03802575923036784
Test set: Average loss: 0.0001, Accuracy: 1249/10000 (97%)
Train Epoch: 44 [448/5000] Loss: 0.053302
Train Epoch: 44 [960/5000] Loss: 0.015452
Train Epoch: 44 [1472/5000] Loss: 0.049132
Train Epoch: 44 [1984/5000] Loss: 0.028816
Train Epoch: 44 [2496/5000] Loss: 0.033576
0.04524776458274573
Test set: Average loss: 0.0001, Accuracy: 1256/10000 (98%)
Train Epoch: 45 [448/5000] Loss: 0.060601
Train Epoch: 45 [960/5000] Loss: 0.023622
Train Epoch: 45 [1472/5000] Loss: 0.006234
Train Epoch: 45 [1984/5000] Loss: 0.040829
Train Epoch: 45 [2496/5000] Loss: 0.044075
0.02673335822764784
Test set: Average loss: 0.0001, Accuracy: 1252/10000 (97%)
Train Epoch: 46 [448/5000] Loss: 0.057646
Train Epoch: 46 [960/5000] Loss: 0.032832
Train Epoch: 46 [1472/5000] Loss: 0.020663
Train Epoch: 46 [1984/5000] Loss: 0.010912
Train Epoch: 46 [2496/5000] Loss: 0.057533
0.027547846478410066
Test set: Average loss: 0.0001, Accuracy: 1255/10000 (98%)
Train Epoch: 47 [448/5000] Loss: 0.015542
Train Epoch: 47 [960/5000] Loss: 0.087516
Train Epoch: 47 [1472/5000] Loss: 0.037386
Train Epoch: 47 [1984/5000] Loss: 0.044959
Train Epoch: 47 [2496/5000] Loss: 0.049140
0.032653901563026014
Test set: Average loss: 0.0001, Accuracy: 1242/10000 (97%)
Train Epoch: 48 [448/5000] Loss: 0.031235
Train Epoch: 48 [960/5000] Loss: 0.057731
Train Epoch: 48 [1472/5000] Loss: 0.023574
Train Epoch: 48 [1984/5000] Loss: 0.051722
Train Epoch: 48 [2496/5000] Loss: 0.020835
0.029008962900843472
Test set: Average loss: 0.0001, Accuracy: 1249/10000 (97%)
Train Epoch: 49 [448/5000] Loss: 0.017079
Train Epoch: 49 [960/5000] Loss: 0.007227
Train Epoch: 49 [1472/5000] Loss: 0.009994
Train Epoch: 49 [1984/5000] Loss: 0.046724
```

```
Train Epoch: 49 [2496/5000] Loss: 0.027271
0.029323481302708388
Test set: Average loss: 0.0001, Accuracy: 1258/10000 (98%)
Train and predict complete!
```

EX1: Conv2D-Conv2D-Conv2D-MaxPooling-Flatten-Dense 98.05% EX2: Conv2D-MaxPooling-Conv2D-MaxPooling-Conv2D-MaxPooling-Flatten-Dense 92% EX3: Conv2D-MaxPooling-Conv2D-MaxPooling-DropOut-Conv2D-MaxPooling-DropOut-Flatten-Dense-Dense-Dense 99%

[28]:
```
all_preds=[]
all_labels=[]

for i,data in enumerate(testloader):
    imgs,labels=data
    imgs_gpu=imgs.cuda()
    labels_gpu=labels.cuda()
    all_labels=np.append(all_labels,labels_gpu.cpu().numpy())

    output=net_gpu(imgs_gpu)
    _, preds = torch.max(output, 1)
    all_preds=np.append(all_preds,preds.tolist())

print(all_preds[:10])
print(all_labels[:10])
```

```
[2. 2. 3. 3. 2. 2. 0. 3. 2. 3.]
[2. 2. 3. 3. 2. 2. 0. 3. 2. 3.]
```

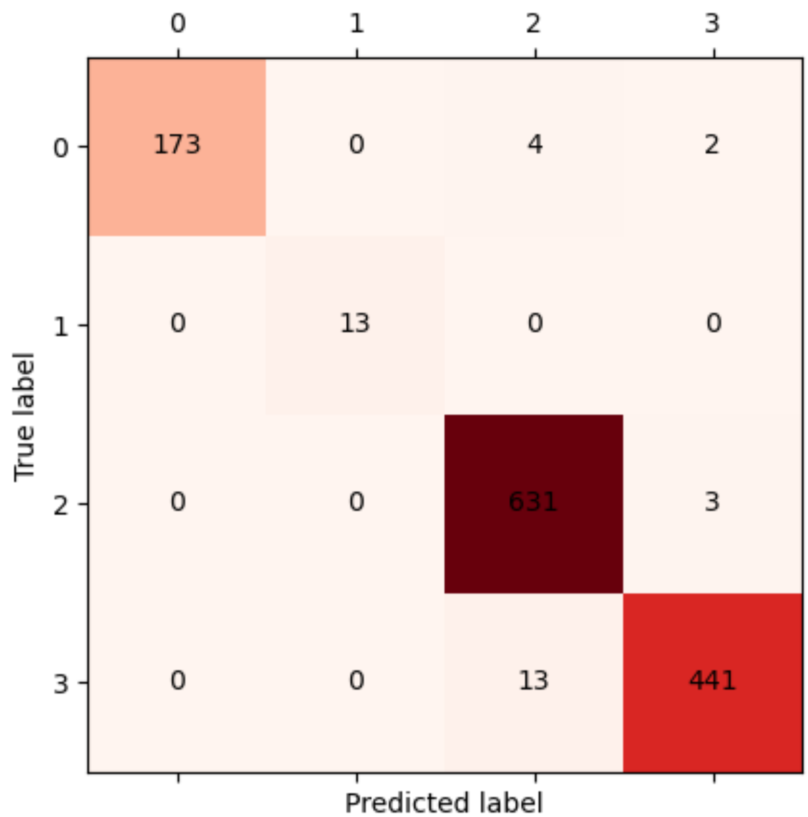
[29]:
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

y_pred = all_preds
y_true = all_labels

C = confusion_matrix(y_true, y_pred, labels=[0,1,2,3])
plt.matshow(C, cmap=plt.cm.Reds)

for i in range(len(C)):
    for j in range(len(C)):
        plt.annotate(C[j, i], xy=(i, j), horizontalalignment='center',␣
 ↪verticalalignment='center')

plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



[30]:
```
from sklearn.metrics import precision_score,recall_score,f1_score
precision = precision_score(all_labels,all_preds,average='weighted')
recall = recall_score(all_labels,all_preds,average='weighted')
f1 = f1_score(all_labels,all_preds,average='weighted')
print(precision,recall,f1)
```

```
0.9830293733218458 0.9828125 0.9827953656285455
```

[ ]:
```

```

GAN Model

December 16, 2022

[1]:
```python
import torch
from torchvision import transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
```

[2]:
```python
data_transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))  # Any Reason for␣
 ↪this?
])

data=ImageFolder("./archive/Dataset/", transform=data_transform)
```

[3]:
```python
BATCH_SIZE = 128
CHANNELS = 3
RAND_DIM = 100
IMG_DIM = 128
NGPU = 1

device = torch.device("cuda:0")

dataloader=DataLoader(data,batch_size=BATCH_SIZE,drop_last=False,shuffle=True)
```

[4]:
```python
import torch.nn.functional as F

class DNet(torch.nn.Module): # Discriminator Net (Same as training net)
    def __init__(self):
        super(DNet, self).__init__()
        # Set for convolution operation
        self.conv1 = torch.nn.Sequential(
            torch.nn.Conv2d(CHANNELS, 16, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
        )
        self.conv2 = torch.nn.Sequential(
            torch.nn.Conv2d(16, 32, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
```

```
        )
        self.conv3 = torch.nn.Sequential(
            torch.nn.Conv2d(32, 64, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
        )
        self.conv4 = torch.nn.Sequential(
            torch.nn.Conv2d(64, 128, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
        )
        self.fc1 = torch.nn.Sequential(
            torch.nn.Linear(128*8*8, 32),
            torch.nn.ReLU()
        )
        self.fc2 = torch.nn.Sequential(
            torch.nn.Linear(32, 1),
            torch.nn.Sigmoid()
        )

    def forward(self, x):
        # Three-layer convolutional network (Conv -> ReLU -> MaxPool)
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.conv3(x)
        x = self.conv4(x)
        x = x.view(-1, 128*8*8)
        x = self.fc1(x) # Fully connected layer -> ReLU
        x = self.fc2(x)
        return x

dnet_gpu = DNet().to(device)
```

[5]:
```
class GNet(torch.nn.Module): # Generator Net
    def __init__(self):
        super(GNet, self).__init__()
        # Set for convolution operation
        self.conv0 = torch.nn.Sequential(
            torch.nn.ConvTranspose2d(RAND_DIM, IMG_DIM * 16, kernel_size=4,␣
 ↪stride=1, padding=0, bias=False),
            torch.nn.BatchNorm2d(IMG_DIM * 16),
            torch.nn.ReLU(),
        )
        self.conv1 = torch.nn.Sequential(
            torch.nn.ConvTranspose2d(IMG_DIM * 16, IMG_DIM * 8, kernel_size=4,␣
 ↪stride=2, padding=1, bias=False),
            torch.nn.BatchNorm2d(IMG_DIM * 8),
```

```python
            torch.nn.ReLU(),
        )
        self.conv2 = torch.nn.Sequential(
            torch.nn.ConvTranspose2d(IMG_DIM * 8, IMG_DIM * 4, kernel_size=4, stride=2, padding=1, bias=False),
            torch.nn.BatchNorm2d(IMG_DIM * 4),
            torch.nn.ReLU(),
        )
        self.conv3 = torch.nn.Sequential(
            torch.nn.ConvTranspose2d(IMG_DIM * 4, IMG_DIM * 2, kernel_size=4, stride=2, padding=1, bias=False),
            torch.nn.BatchNorm2d(IMG_DIM * 2),
            torch.nn.ReLU(),
        )
        self.conv4 = torch.nn.Sequential(
            torch.nn.ConvTranspose2d(IMG_DIM * 2, IMG_DIM, kernel_size=4, stride=2, padding=1, bias=False),
            torch.nn.BatchNorm2d(IMG_DIM),
            torch.nn.ReLU(),
        )
        self.conv5 = torch.nn.Sequential(
            torch.nn.ConvTranspose2d(IMG_DIM, CHANNELS, kernel_size=4, stride=2, padding=1, bias=False),
            torch.nn.Tanh(),
        )

    def forward(self, x):
        # Three-layer convolutional network (Conv -> ReLU -> MaxPool)
        x = self.conv0(x)
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.conv3(x)
        x = self.conv4(x)
        x = self.conv5(x)
        return x

gnet_gpu = GNet().to(device)
#   128*128*3
```

[6]:

```python
from torch import optim

optimizer_d = optim.Adam(
    dnet_gpu.parameters(),
    lr = 0.001,
    betas = (0.9, 0.999),
    eps = 1e-08,
    weight_decay = 0,
```

```
    amsgrad = False
)

optimizer_g = optim.Adam(
    gnet_gpu.parameters(),
    lr = 0.001,
    betas = (0.9, 0.999),
    eps = 1e-08,
    weight_decay = 0,
    amsgrad = False
)

loss_func = torch.nn.BCELoss()

fixed_noise = torch.randn(64, RAND_DIM, 1, 1, device=device)
```

[7]:

```
from torch.utils.tensorboard import SummaryWriter
import matplotlib.pyplot as plt
import numpy as np
import torchvision.utils as vutils

summaryWriter = SummaryWriter("logs/GAN_recorded_4")

img_list = []
all_d_loss = []
all_g_loss = []


for epoch in range(100):
    errAll = 0
    errG = 0
    for i, data in enumerate(dataloader, 0):
        # Train with real batch
        optimizer_d.zero_grad()
        inputs_gpu = data[0].to(device)
        label = torch.full((BATCH_SIZE,), 1, dtype=torch.float, device=device)
        outputs_gpu = dnet_gpu(inputs_gpu).view(-1)
        d_loss_real = loss_func(outputs_gpu, label)
        d_loss_real.backward()
        D_x = outputs_gpu.mean().item()

        # Train with fake batch
        rand_noise = torch.randn(BATCH_SIZE, RAND_DIM, 1, 1, device=device)
        fake_images = gnet_gpu(rand_noise)
        label.fill_(0)
        outputs_fake = dnet_gpu(fake_images.detach()).view(-1)
        fake_loss = loss_func(outputs_fake, label)
```

```
        fake_loss.backward()
        D_G_z1 = outputs_fake.mean().item()
        errAll = d_loss_real + fake_loss
        optimizer_d.step()

        # Update G
        optimizer_g.zero_grad()
        label.fill_(1)
        outputs_g = dnet_gpu(fake_images).view(-1)
        errG = loss_func(outputs_g, label)
        errG.backward()
        # D_G_z2 = outputs_g.mean().item()
        optimizer_g.step()

        if i % 16 == 15:
            print('Train Epoch: %d [%d/5000] D-Loss: %.6f G-Loss: %.6f' ␣
↪%(epoch, i*128,errAll.item(), errG.item()))
    summaryWriter.add_scalar("d_loss", errAll, epoch)
    summaryWriter.add_scalar("g_loss", errG, epoch)
    all_d_loss = np.append(all_d_loss,errAll.item())
    # print(all_d_loss)
    all_g_loss = np.append(all_g_loss,errG.item())
    # print(all_g_loss)

    if (epoch % 5 == 0 or epoch > 90):
        with torch.no_grad():
            fake = gnet_gpu(fixed_noise).detach().cpu()
            img_list.append(vutils.make_grid(fake, padding=2, normalize=True))
        plt.subplot(1,2,2)
        plt.axis("off")
        plt.title("Fake")
        plt.imshow(np.transpose(img_list[-1],(1,2,0)))
        plt.show()

print('Train and predict complete!')
```

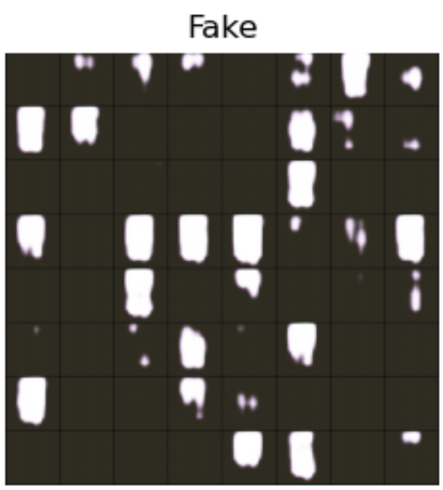
```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
c:
↪\ProgramData\Anaconda3\envs\pytorch\lib\site-packages\tensorboard\compat\__ini __.
↪py in tf()
     41     try:
---> 42         from tensorboard.compat import notf  # noqa: F401
     43     except ImportError:

ImportError: cannot import name 'notf' from 'tensorboard.compat' (c:
↪\ProgramData\Anaconda3\envs\pytorch\lib\site-packages\tensorboard\compat\__ini __.
↪py)
```

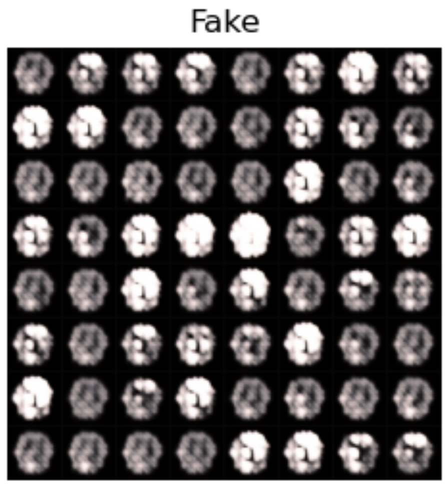
```
During handling of the above exception, another exception occurred:

RuntimeError                              Traceback (most recent call last)
RuntimeError: module compiled against API version 0xe but this version of numpy is␣
↪0xd
```

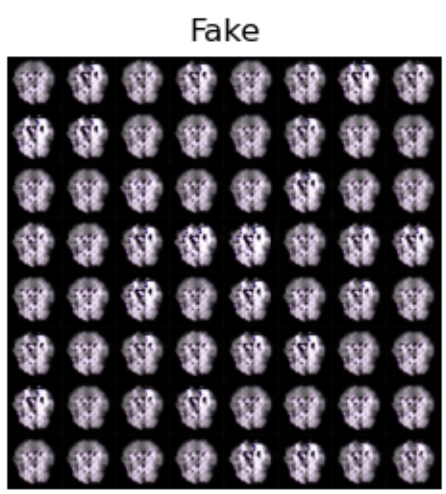
```
Train Epoch: 0 [1920/5000] D-Loss: 0.620631 G-Loss: 3.752686
Train Epoch: 0 [3968/5000] D-Loss: 0.323365 G-Loss: 2.897259
Train Epoch: 0 [6016/5000] D-Loss: 0.029723 G-Loss: 8.137974
```



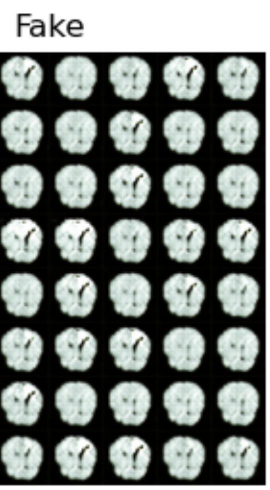

```
Train Epoch: 1 [1920/5000] D-Loss: 0.028731 G-Loss: 7.280564
Train Epoch: 1 [3968/5000] D-Loss: 0.036729 G-Loss: 11.439844
Train Epoch: 1 [6016/5000] D-Loss: 0.625033 G-Loss: 9.541747
Train Epoch: 2 [1920/5000] D-Loss: 0.605415 G-Loss: 3.386896
Train Epoch: 2 [3968/5000] D-Loss: 1.470398 G-Loss: 2.019850
Train Epoch: 2 [6016/5000] D-Loss: 1.160665 G-Loss: 1.605094
Train Epoch: 3 [1920/5000] D-Loss: 0.799886 G-Loss: 1.758806
Train Epoch: 3 [3968/5000] D-Loss: 0.698381 G-Loss: 2.404707
Train Epoch: 3 [6016/5000] D-Loss: 0.351882 G-Loss: 2.371909
Train Epoch: 4 [1920/5000] D-Loss: 0.500038 G-Loss: 3.949688
Train Epoch: 4 [3968/5000] D-Loss: 1.846416 G-Loss: 0.099139
Train Epoch: 4 [6016/5000] D-Loss: 0.495032 G-Loss: 5.128822
Train Epoch: 5 [1920/5000] D-Loss: 0.251757 G-Loss: 3.386813
Train Epoch: 5 [3968/5000] D-Loss: 0.297179 G-Loss: 4.978414
Train Epoch: 5 [6016/5000] D-Loss: 1.813661 G-Loss: 4.517722
```




```
Train Epoch: 6 [1920/5000] D-Loss: 0.697197 G-Loss: 2.558181
Train Epoch: 6 [3968/5000] D-Loss: 2.148966 G-Loss: 1.067138
Train Epoch: 6 [6016/5000] D-Loss: 0.826043 G-Loss: 2.669941
Train Epoch: 7 [1920/5000] D-Loss: 0.344904 G-Loss: 2.617375
Train Epoch: 7 [3968/5000] D-Loss: 0.737916 G-Loss: 2.819423
Train Epoch: 7 [6016/5000] D-Loss: 2.308386 G-Loss: 1.269044
Train Epoch: 8 [1920/5000] D-Loss: 0.286483 G-Loss: 2.478392
Train Epoch: 8 [3968/5000] D-Loss: 0.815064 G-Loss: 1.466595
Train Epoch: 8 [6016/5000] D-Loss: 0.786132 G-Loss: 1.878022
Train Epoch: 9 [1920/5000] D-Loss: 0.594064 G-Loss: 3.091740
Train Epoch: 9 [3968/5000] D-Loss: 0.632960 G-Loss: 1.804797
Train Epoch: 9 [6016/5000] D-Loss: 0.678797 G-Loss: 2.408049
Train Epoch: 10 [1920/5000] D-Loss: 0.520791 G-Loss: 2.356299
Train Epoch: 10 [3968/5000] D-Loss: 0.504598 G-Loss: 1.949342
Train Epoch: 10 [6016/5000] D-Loss: 0.952953 G-Loss: 2.442424
```

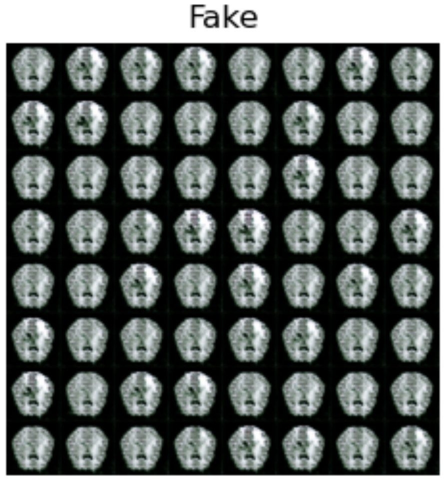


```
Train Epoch: 11 [1920/5000] D-Loss: 0.747288 G-Loss: 1.321373
Train Epoch: 11 [3968/5000] D-Loss: 0.217975 G-Loss: 3.468381
Train Epoch: 11 [6016/5000] D-Loss: 1.121833 G-Loss: 3.947107
Train Epoch: 12 [1920/5000] D-Loss: 0.301901 G-Loss: 4.665808
Train Epoch: 12 [3968/5000] D-Loss: 0.790332 G-Loss: 2.797332
Train Epoch: 12 [6016/5000] D-Loss: 0.777384 G-Loss: 1.147980
Train Epoch: 13 [1920/5000] D-Loss: 0.426973 G-Loss: 3.162313
Train Epoch: 13 [3968/5000] D-Loss: 2.018709 G-Loss: 2.006040
Train Epoch: 13 [6016/5000] D-Loss: 1.706564 G-Loss: 1.183115
Train Epoch: 14 [1920/5000] D-Loss: 0.976977 G-Loss: 1.544701
Train Epoch: 14 [3968/5000] D-Loss: 0.183519 G-Loss: 2.503435
Train Epoch: 14 [6016/5000] D-Loss: 0.234319 G-Loss: 3.637300
Train Epoch: 15 [1920/5000] D-Loss: 2.160779 G-Loss: 2.263383
Train Epoch: 15 [3968/5000] D-Loss: 0.613857 G-Loss: 2.966665
Train Epoch: 15 [6016/5000] D-Loss: 0.116188 G-Loss: 3.391873
```

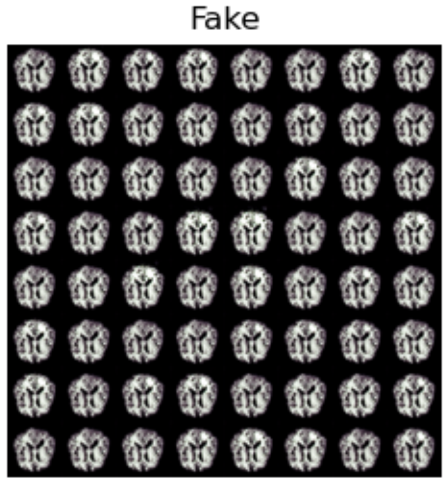



```
Train Epoch: 16 [1920/5000] D-Loss: 1.226496 G-Loss: 1.939894
Train Epoch: 16 [3968/5000] D-Loss: 0.511762 G-Loss: 2.786935
Train Epoch: 16 [6016/5000] D-Loss: 0.512526 G-Loss: 3.337615
Train Epoch: 17 [1920/5000] D-Loss: 0.237639 G-Loss: 4.138672
Train Epoch: 17 [3968/5000] D-Loss: 1.572751 G-Loss: 2.334280
Train Epoch: 17 [6016/5000] D-Loss: 0.083890 G-Loss: 3.295315
Train Epoch: 18 [1920/5000] D-Loss: 0.073469 G-Loss: 3.084571
Train Epoch: 18 [3968/5000] D-Loss: 1.073678 G-Loss: 2.440577
Train Epoch: 18 [6016/5000] D-Loss: 0.224322 G-Loss: 3.437482
Train Epoch: 19 [1920/5000] D-Loss: 0.426915 G-Loss: 3.981864
Train Epoch: 19 [3968/5000] D-Loss: 0.346252 G-Loss: 2.870422
Train Epoch: 19 [6016/5000] D-Loss: 0.415284 G-Loss: 3.943544
```

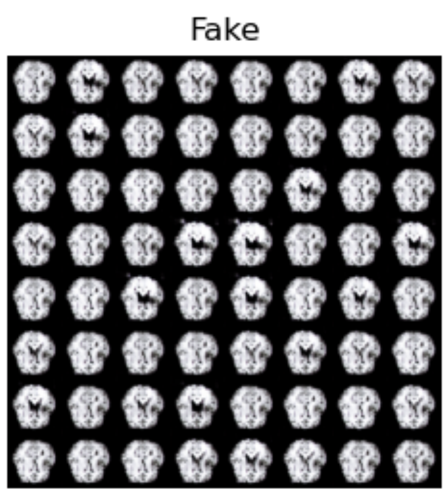
```
Train Epoch: 20 [1920/5000] D-Loss: 0.339909 G-Loss: 3.632792
Train Epoch: 20 [3968/5000] D-Loss: 0.761192 G-Loss: 4.201749
Train Epoch: 20 [6016/5000] D-Loss: 1.170472 G-Loss: 3.148186
```



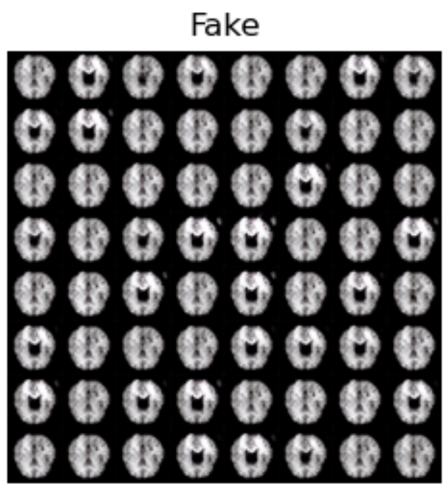

```
Train Epoch: 21 [1920/5000] D-Loss: 0.471630 G-Loss: 5.876425
Train Epoch: 21 [3968/5000] D-Loss: 0.410605 G-Loss: 4.369297
Train Epoch: 21 [6016/5000] D-Loss: 0.644118 G-Loss: 2.656657
Train Epoch: 22 [1920/5000] D-Loss: 0.226817 G-Loss: 4.529696
Train Epoch: 22 [3968/5000] D-Loss: 0.362913 G-Loss: 3.613520
Train Epoch: 22 [6016/5000] D-Loss: 0.908346 G-Loss: 4.751914
Train Epoch: 23 [1920/5000] D-Loss: 0.278952 G-Loss: 6.420827
Train Epoch: 23 [3968/5000] D-Loss: 0.467144 G-Loss: 4.078788
Train Epoch: 23 [6016/5000] D-Loss: 0.311568 G-Loss: 0.368239
Train Epoch: 24 [1920/5000] D-Loss: 0.812857 G-Loss: 5.152391
Train Epoch: 24 [3968/5000] D-Loss: 0.065257 G-Loss: 6.432172
Train Epoch: 24 [6016/5000] D-Loss: 0.179623 G-Loss: 5.239373
Train Epoch: 25 [1920/5000] D-Loss: 0.709756 G-Loss: 1.933937
Train Epoch: 25 [3968/5000] D-Loss: 0.474705 G-Loss: 3.050380
Train Epoch: 25 [6016/5000] D-Loss: 0.399784 G-Loss: 3.023294
```




```
Train Epoch: 26 [1920/5000] D-Loss: 0.324032 G-Loss: 3.872917
Train Epoch: 26 [3968/5000] D-Loss: 0.239058 G-Loss: 2.194970
Train Epoch: 26 [6016/5000] D-Loss: 0.526818 G-Loss: 1.905610
Train Epoch: 27 [1920/5000] D-Loss: 0.563949 G-Loss: 6.254292
Train Epoch: 27 [3968/5000] D-Loss: 0.669935 G-Loss: 2.771668
Train Epoch: 27 [6016/5000] D-Loss: 0.448923 G-Loss: 5.000283
Train Epoch: 28 [1920/5000] D-Loss: 0.080021 G-Loss: 6.041643
Train Epoch: 28 [3968/5000] D-Loss: 0.627078 G-Loss: 4.388107
Train Epoch: 28 [6016/5000] D-Loss: 0.429771 G-Loss: 4.632345
Train Epoch: 29 [1920/5000] D-Loss: 0.569584 G-Loss: 2.282325
Train Epoch: 29 [3968/5000] D-Loss: 0.655299 G-Loss: 2.039386
Train Epoch: 29 [6016/5000] D-Loss: 0.373528 G-Loss: 3.650390
Train Epoch: 30 [1920/5000] D-Loss: 0.893325 G-Loss: 5.131932
Train Epoch: 30 [3968/5000] D-Loss: 0.418272 G-Loss: 5.854839
Train Epoch: 30 [6016/5000] D-Loss: 0.163301 G-Loss: 1.771946
```

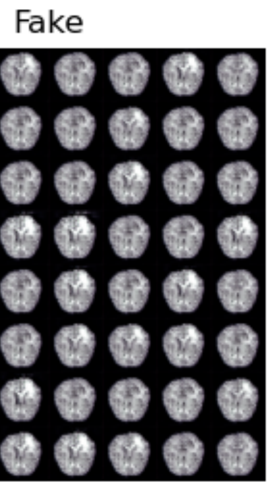


```
Train Epoch: 31 [1920/5000] D-Loss: 0.671122 G-Loss: 4.044340
Train Epoch: 31 [3968/5000] D-Loss: 0.328745 G-Loss: 2.411425
Train Epoch: 31 [6016/5000] D-Loss: 1.031084 G-Loss: 7.007136
Train Epoch: 32 [1920/5000] D-Loss: 0.535564 G-Loss: 2.910047
Train Epoch: 32 [3968/5000] D-Loss: 0.453885 G-Loss: 2.659522
Train Epoch: 32 [6016/5000] D-Loss: 0.470690 G-Loss: 2.651394
Train Epoch: 33 [1920/5000] D-Loss: 0.190090 G-Loss: 3.872406
Train Epoch: 33 [3968/5000] D-Loss: 0.379193 G-Loss: 3.997615
Train Epoch: 33 [6016/5000] D-Loss: 0.463718 G-Loss: 4.301066
Train Epoch: 34 [1920/5000] D-Loss: 0.269728 G-Loss: 3.047270
Train Epoch: 34 [3968/5000] D-Loss: 0.358603 G-Loss: 2.665477
Train Epoch: 34 [6016/5000] D-Loss: 0.493482 G-Loss: 3.197266
Train Epoch: 35 [1920/5000] D-Loss: 1.535028 G-Loss: 8.424935
Train Epoch: 35 [3968/5000] D-Loss: 1.652283 G-Loss: 6.296321
Train Epoch: 35 [6016/5000] D-Loss: 0.397874 G-Loss: 2.168586
```

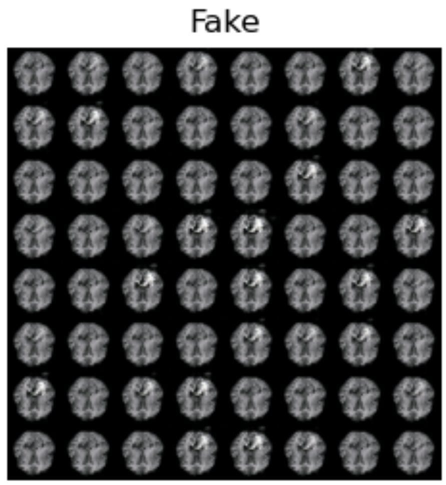



```
Train Epoch: 36 [1920/5000] D-Loss: 0.091191 G-Loss: 5.349733
Train Epoch: 36 [3968/5000] D-Loss: 0.526846 G-Loss: 3.767495
Train Epoch: 36 [6016/5000] D-Loss: 0.414172 G-Loss: 3.140286
Train Epoch: 37 [1920/5000] D-Loss: 0.455696 G-Loss: 3.021224
Train Epoch: 37 [3968/5000] D-Loss: 0.407753 G-Loss: 3.078928
Train Epoch: 37 [6016/5000] D-Loss: 0.477401 G-Loss: 3.090838
Train Epoch: 38 [1920/5000] D-Loss: 0.591929 G-Loss: 2.232069
Train Epoch: 38 [3968/5000] D-Loss: 0.420592 G-Loss: 2.907135
Train Epoch: 38 [6016/5000] D-Loss: 0.497808 G-Loss: 3.974330
Train Epoch: 39 [1920/5000] D-Loss: 0.195395 G-Loss: 3.277156
Train Epoch: 39 [3968/5000] D-Loss: 0.535334 G-Loss: 4.609308
Train Epoch: 39 [6016/5000] D-Loss: 0.793924 G-Loss: 6.156008
```

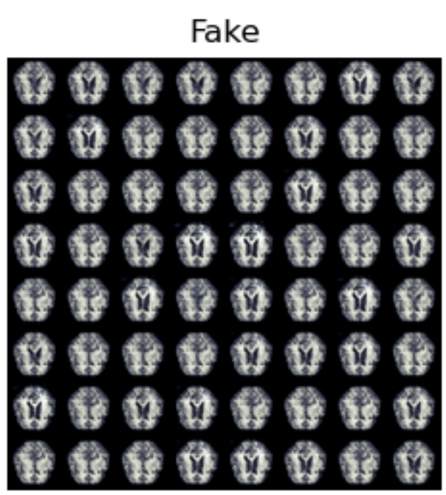
```
Train Epoch: 40 [1920/5000] D-Loss: 0.253465 G-Loss: 5.681975
Train Epoch: 40 [3968/5000] D-Loss: 0.372263 G-Loss: 3.653792
Train Epoch: 40 [6016/5000] D-Loss: 0.206606 G-Loss: 3.528642
```




```
Train Epoch: 41 [1920/5000] D-Loss: 0.422421 G-Loss: 3.069598
Train Epoch: 41 [3968/5000] D-Loss: 0.393572 G-Loss: 3.686675
Train Epoch: 41 [6016/5000] D-Loss: 0.620056 G-Loss: 3.437696
Train Epoch: 42 [1920/5000] D-Loss: 0.400316 G-Loss: 3.903860
Train Epoch: 42 [3968/5000] D-Loss: 0.320969 G-Loss: 5.309864
Train Epoch: 42 [6016/5000] D-Loss: 0.857732 G-Loss: 4.088079
Train Epoch: 43 [1920/5000] D-Loss: 0.454659 G-Loss: 2.122256
Train Epoch: 43 [3968/5000] D-Loss: 0.553265 G-Loss: 3.349783
Train Epoch: 43 [6016/5000] D-Loss: 1.112058 G-Loss: 1.925044
Train Epoch: 44 [1920/5000] D-Loss: 0.562896 G-Loss: 2.611276
Train Epoch: 44 [3968/5000] D-Loss: 0.333349 G-Loss: 3.661124
Train Epoch: 44 [6016/5000] D-Loss: 0.346085 G-Loss: 5.472315
Train Epoch: 45 [1920/5000] D-Loss: 0.310228 G-Loss: 3.882528
Train Epoch: 45 [3968/5000] D-Loss: 0.463913 G-Loss: 3.167953
Train Epoch: 45 [6016/5000] D-Loss: 0.222674 G-Loss: 4.035709
```

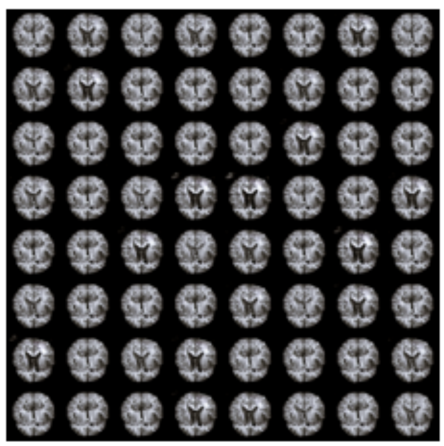


```
Train Epoch: 46 [1920/5000] D-Loss: 1.105503 G-Loss: 0.628190
Train Epoch: 46 [3968/5000] D-Loss: 0.897577 G-Loss: 5.384284
Train Epoch: 46 [6016/5000] D-Loss: 1.143742 G-Loss: 3.261919
Train Epoch: 47 [1920/5000] D-Loss: 0.244393 G-Loss: 3.333646
Train Epoch: 47 [3968/5000] D-Loss: 0.360163 G-Loss: 2.991303
Train Epoch: 47 [6016/5000] D-Loss: 0.467425 G-Loss: 3.157298
Train Epoch: 48 [1920/5000] D-Loss: 0.554853 G-Loss: 3.152487
Train Epoch: 48 [3968/5000] D-Loss: 0.511649 G-Loss: 3.615672
Train Epoch: 48 [6016/5000] D-Loss: 0.350106 G-Loss: 4.072041
Train Epoch: 49 [1920/5000] D-Loss: 0.547387 G-Loss: 4.647380
Train Epoch: 49 [3968/5000] D-Loss: 0.409953 G-Loss: 4.687531
Train Epoch: 49 [6016/5000] D-Loss: 0.375135 G-Loss: 7.027030
Train Epoch: 50 [1920/5000] D-Loss: 0.828060 G-Loss: 3.288312
Train Epoch: 50 [3968/5000] D-Loss: 0.579508 G-Loss: 3.301752
Train Epoch: 50 [6016/5000] D-Loss: 0.137874 G-Loss: 3.658839
```



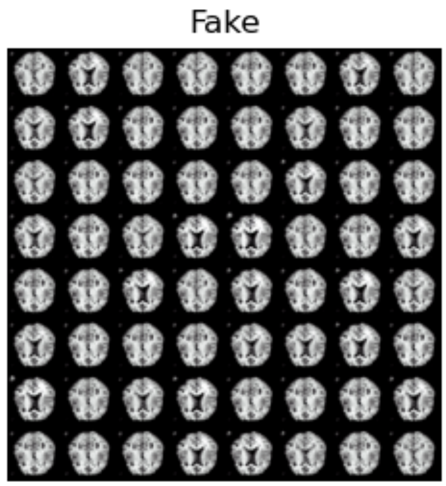
```
Train Epoch: 51 [1920/5000] D-Loss: 0.433462 G-Loss: 3.734538
Train Epoch: 51 [3968/5000] D-Loss: 0.338235 G-Loss: 3.501250
Train Epoch: 51 [6016/5000] D-Loss: 0.183905 G-Loss: 7.584962
Train Epoch: 52 [1920/5000] D-Loss: 1.312748 G-Loss: 3.879839
Train Epoch: 52 [3968/5000] D-Loss: 0.282865 G-Loss: 2.203984
Train Epoch: 52 [6016/5000] D-Loss: 0.455327 G-Loss: 2.740016
Train Epoch: 53 [1920/5000] D-Loss: 0.274456 G-Loss: 4.119394
Train Epoch: 53 [3968/5000] D-Loss: 0.426200 G-Loss: 3.742188
Train Epoch: 53 [6016/5000] D-Loss: 0.492660 G-Loss: 2.289934
Train Epoch: 54 [1920/5000] D-Loss: 0.683404 G-Loss: 4.316451
Train Epoch: 54 [3968/5000] D-Loss: 0.444979 G-Loss: 2.287118
Train Epoch: 54 [6016/5000] D-Loss: 0.461759 G-Loss: 3.623129
Train Epoch: 55 [1920/5000] D-Loss: 0.428947 G-Loss: 5.296576
Train Epoch: 55 [3968/5000] D-Loss: 0.592178 G-Loss: 3.621313
Train Epoch: 55 [6016/5000] D-Loss: 0.715334 G-Loss: 2.838659
```

Fake

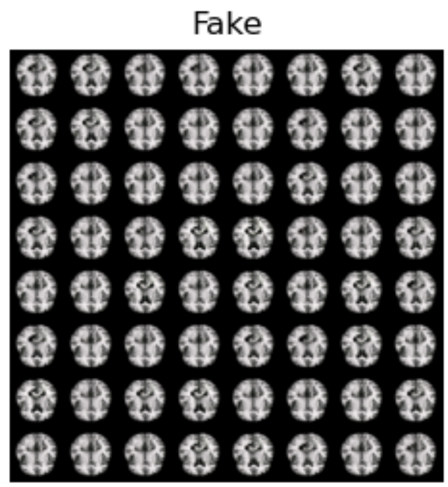


```
Train Epoch: 56 [1920/5000] D-Loss: 0.602180 G-Loss: 2.922879
Train Epoch: 56 [3968/5000] D-Loss: 0.748578 G-Loss: 2.055559
Train Epoch: 56 [6016/5000] D-Loss: 0.682449 G-Loss: 3.620935
Train Epoch: 57 [1920/5000] D-Loss: 0.123806 G-Loss: 4.566719
Train Epoch: 57 [3968/5000] D-Loss: 0.735957 G-Loss: 1.936661
Train Epoch: 57 [6016/5000] D-Loss: 0.262648 G-Loss: 6.794894
Train Epoch: 58 [1920/5000] D-Loss: 0.328910 G-Loss: 3.692234
Train Epoch: 58 [3968/5000] D-Loss: 0.602013 G-Loss: 4.049365
Train Epoch: 58 [6016/5000] D-Loss: 0.697813 G-Loss: 4.068985
Train Epoch: 59 [1920/5000] D-Loss: 0.218673 G-Loss: 4.970861
Train Epoch: 59 [3968/5000] D-Loss: 0.208075 G-Loss: 4.716311
Train Epoch: 59 [6016/5000] D-Loss: 0.121100 G-Loss: 6.131151
```

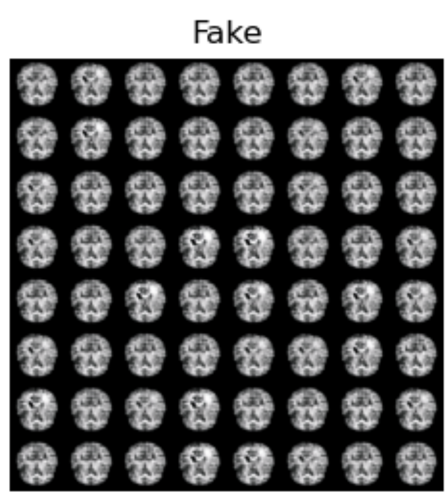
```
Train Epoch: 60 [1920/5000] D-Loss: 0.178941 G-Loss: 4.564821
Train Epoch: 60 [3968/5000] D-Loss: 2.051748 G-Loss: 4.503987
Train Epoch: 60 [6016/5000] D-Loss: 0.229077 G-Loss: 3.034917
```



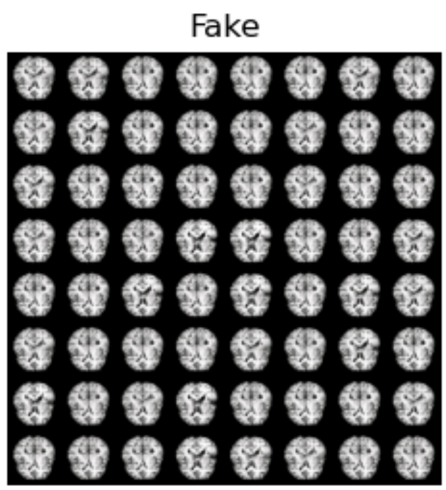

```
Train Epoch: 61 [1920/5000] D-Loss: 0.294386 G-Loss: 4.903909
Train Epoch: 61 [3968/5000] D-Loss: 0.506195 G-Loss: 3.793943
Train Epoch: 61 [6016/5000] D-Loss: 0.360867 G-Loss: 3.407473
Train Epoch: 62 [1920/5000] D-Loss: 0.470193 G-Loss: 3.406794
Train Epoch: 62 [3968/5000] D-Loss: 0.218482 G-Loss: 4.071246
Train Epoch: 62 [6016/5000] D-Loss: 0.415637 G-Loss: 4.034445
Train Epoch: 63 [1920/5000] D-Loss: 1.349407 G-Loss: 11.452553
Train Epoch: 63 [3968/5000] D-Loss: 0.498367 G-Loss: 1.958487
Train Epoch: 63 [6016/5000] D-Loss: 0.451440 G-Loss: 4.889888
Train Epoch: 64 [1920/5000] D-Loss: 0.635080 G-Loss: 3.049534
Train Epoch: 64 [3968/5000] D-Loss: 0.517202 G-Loss: 3.979232
Train Epoch: 64 [6016/5000] D-Loss: 0.668467 G-Loss: 3.128254
Train Epoch: 65 [1920/5000] D-Loss: 0.258526 G-Loss: 5.562471
Train Epoch: 65 [3968/5000] D-Loss: 0.258311 G-Loss: 5.226520
Train Epoch: 65 [6016/5000] D-Loss: 0.833445 G-Loss: 2.271490
```




```
Train Epoch: 66 [1920/5000] D-Loss: 0.462170 G-Loss: 2.577061
Train Epoch: 66 [3968/5000] D-Loss: 0.829438 G-Loss: 2.561436
Train Epoch: 66 [6016/5000] D-Loss: 0.379216 G-Loss: 3.626442
Train Epoch: 67 [1920/5000] D-Loss: 0.566051 G-Loss: 3.835438
Train Epoch: 67 [3968/5000] D-Loss: 0.312500 G-Loss: 4.869793
Train Epoch: 67 [6016/5000] D-Loss: 1.237827 G-Loss: 6.728691
Train Epoch: 68 [1920/5000] D-Loss: 0.760846 G-Loss: 3.553924
Train Epoch: 68 [3968/5000] D-Loss: 0.221489 G-Loss: 4.302754
Train Epoch: 68 [6016/5000] D-Loss: 0.295149 G-Loss: 3.926167
Train Epoch: 69 [1920/5000] D-Loss: 0.303913 G-Loss: 4.079391
Train Epoch: 69 [3968/5000] D-Loss: 0.450124 G-Loss: 4.902692
Train Epoch: 69 [6016/5000] D-Loss: 0.307388 G-Loss: 3.738111
Train Epoch: 70 [1920/5000] D-Loss: 0.499332 G-Loss: 3.788627
Train Epoch: 70 [3968/5000] D-Loss: 1.000288 G-Loss: 3.621817
Train Epoch: 70 [6016/5000] D-Loss: 0.345774 G-Loss: 3.633478
```

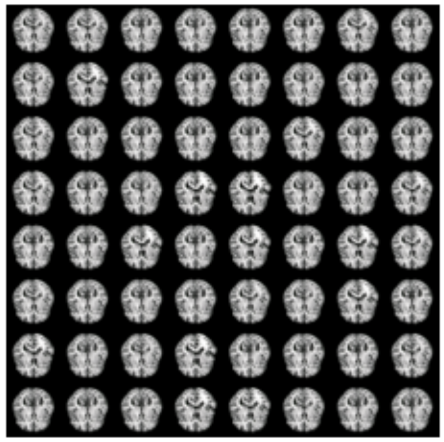


```
Train Epoch: 71 [1920/5000] D-Loss: 0.456077 G-Loss: 4.180548
Train Epoch: 71 [3968/5000] D-Loss: 0.470955 G-Loss: 2.590944
Train Epoch: 71 [6016/5000] D-Loss: 0.320709 G-Loss: 3.051486
Train Epoch: 72 [1920/5000] D-Loss: 0.323433 G-Loss: 4.303418
Train Epoch: 72 [3968/5000] D-Loss: 0.688678 G-Loss: 5.063155
Train Epoch: 72 [6016/5000] D-Loss: 0.608464 G-Loss: 8.791400
Train Epoch: 73 [1920/5000] D-Loss: 0.307055 G-Loss: 3.718576
Train Epoch: 73 [3968/5000] D-Loss: 0.253538 G-Loss: 4.655527
Train Epoch: 73 [6016/5000] D-Loss: 0.297835 G-Loss: 4.410706
Train Epoch: 74 [1920/5000] D-Loss: 0.599724 G-Loss: 2.549782
Train Epoch: 74 [3968/5000] D-Loss: 0.467650 G-Loss: 4.826925
Train Epoch: 74 [6016/5000] D-Loss: 0.117892 G-Loss: 4.276830
Train Epoch: 75 [1920/5000] D-Loss: 0.882187 G-Loss: 6.009505
Train Epoch: 75 [3968/5000] D-Loss: 0.330788 G-Loss: 4.177440
Train Epoch: 75 [6016/5000] D-Loss: 0.142243 G-Loss: 4.769705
```

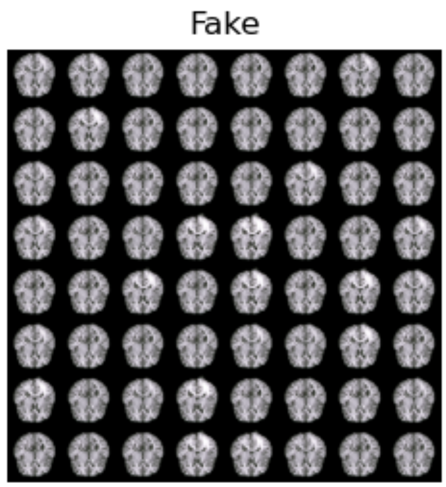



```
Train Epoch: 76 [1920/5000] D-Loss: 0.930533 G-Loss: 1.496459
Train Epoch: 76 [3968/5000] D-Loss: 0.123270 G-Loss: 4.600114
Train Epoch: 76 [6016/5000] D-Loss: 0.279817 G-Loss: 4.005948
Train Epoch: 77 [1920/5000] D-Loss: 0.494490 G-Loss: 5.825949
Train Epoch: 77 [3968/5000] D-Loss: 1.461688 G-Loss: 7.251351
Train Epoch: 77 [6016/5000] D-Loss: 0.391305 G-Loss: 8.335816
Train Epoch: 78 [1920/5000] D-Loss: 0.310106 G-Loss: 7.101126
Train Epoch: 78 [3968/5000] D-Loss: 0.245831 G-Loss: 3.854228
Train Epoch: 78 [6016/5000] D-Loss: 0.117588 G-Loss: 4.326218
Train Epoch: 79 [1920/5000] D-Loss: 0.353043 G-Loss: 3.377371
Train Epoch: 79 [3968/5000] D-Loss: 0.814906 G-Loss: 3.292036
Train Epoch: 79 [6016/5000] D-Loss: 0.290173 G-Loss: 3.774748
```

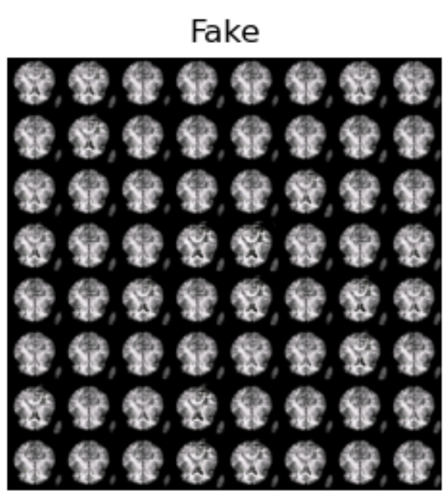
```
Train Epoch: 80 [1920/5000] D-Loss: 0.193964 G-Loss: 4.107721
Train Epoch: 80 [3968/5000] D-Loss: 0.601599 G-Loss: 4.363950
Train Epoch: 80 [6016/5000] D-Loss: 0.724586 G-Loss: 8.054670
```




```
Train Epoch: 81 [1920/5000] D-Loss: 1.834056 G-Loss: 5.708974
Train Epoch: 81 [3968/5000] D-Loss: 0.499039 G-Loss: 2.288345
Train Epoch: 81 [6016/5000] D-Loss: 0.454982 G-Loss: 3.396752
Train Epoch: 82 [1920/5000] D-Loss: 0.518975 G-Loss: 4.194656
Train Epoch: 82 [3968/5000] D-Loss: 0.853817 G-Loss: 2.857837
Train Epoch: 82 [6016/5000] D-Loss: 0.600645 G-Loss: 5.888038
Train Epoch: 83 [1920/5000] D-Loss: 0.600347 G-Loss: 4.196224
Train Epoch: 83 [3968/5000] D-Loss: 0.617052 G-Loss: 2.993189
Train Epoch: 83 [6016/5000] D-Loss: 0.736300 G-Loss: 2.681528
Train Epoch: 84 [1920/5000] D-Loss: 0.373562 G-Loss: 4.070043
Train Epoch: 84 [3968/5000] D-Loss: 0.448438 G-Loss: 4.540978
Train Epoch: 84 [6016/5000] D-Loss: 0.643787 G-Loss: 5.481877
Train Epoch: 85 [1920/5000] D-Loss: 0.076039 G-Loss: 4.833040
Train Epoch: 85 [3968/5000] D-Loss: 0.595265 G-Loss: 3.259089
Train Epoch: 85 [6016/5000] D-Loss: 0.322511 G-Loss: 3.400573
```

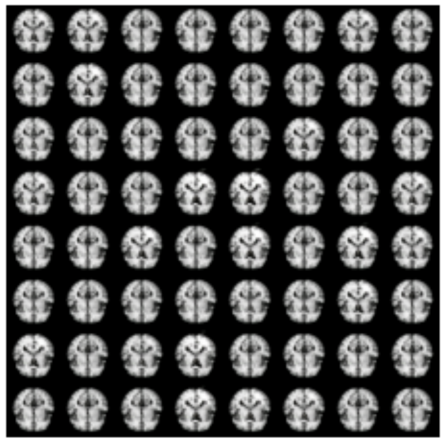


```
Train Epoch: 86 [1920/5000] D-Loss: 0.617445 G-Loss: 3.555153
Train Epoch: 86 [3968/5000] D-Loss: 2.096704 G-Loss: 2.017271
Train Epoch: 86 [6016/5000] D-Loss: 0.368166 G-Loss: 3.262436
Train Epoch: 87 [1920/5000] D-Loss: 0.580030 G-Loss: 4.141155
Train Epoch: 87 [3968/5000] D-Loss: 0.393258 G-Loss: 4.528088
Train Epoch: 87 [6016/5000] D-Loss: 0.603971 G-Loss: 4.511047
Train Epoch: 88 [1920/5000] D-Loss: 0.337258 G-Loss: 4.630699
Train Epoch: 88 [3968/5000] D-Loss: 0.487438 G-Loss: 3.954466
Train Epoch: 88 [6016/5000] D-Loss: 0.263009 G-Loss: 2.974656
Train Epoch: 89 [1920/5000] D-Loss: 0.315792 G-Loss: 4.855607
Train Epoch: 89 [3968/5000] D-Loss: 0.047506 G-Loss: 6.404248
Train Epoch: 89 [6016/5000] D-Loss: 0.093613 G-Loss: 6.918266
Train Epoch: 90 [1920/5000] D-Loss: 1.140128 G-Loss: 4.861363
Train Epoch: 90 [3968/5000] D-Loss: 0.362410 G-Loss: 4.246180
Train Epoch: 90 [6016/5000] D-Loss: 0.393055 G-Loss: 3.921970
```



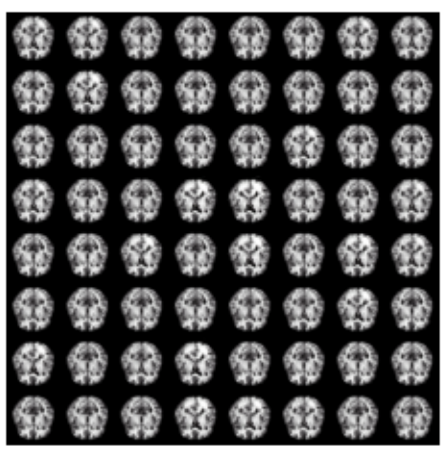
```
Train Epoch: 91 [1920/5000] D-Loss: 0.211894 G-Loss: 4.002560
Train Epoch: 91 [3968/5000] D-Loss: 0.675562 G-Loss: 2.097031
Train Epoch: 91 [6016/5000] D-Loss: 0.523734 G-Loss: 3.677448
```

Fake

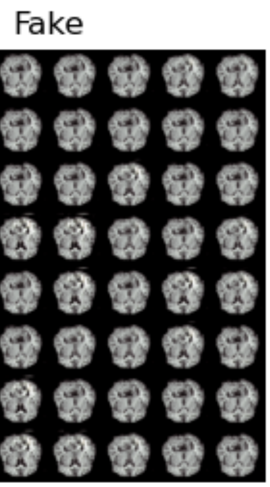


```
Train Epoch: 92 [1920/5000] D-Loss: 0.729768 G-Loss: 2.195317
Train Epoch: 92 [3968/5000] D-Loss: 0.387683 G-Loss: 3.353424
Train Epoch: 92 [6016/5000] D-Loss: 0.851723 G-Loss: 3.480939
```

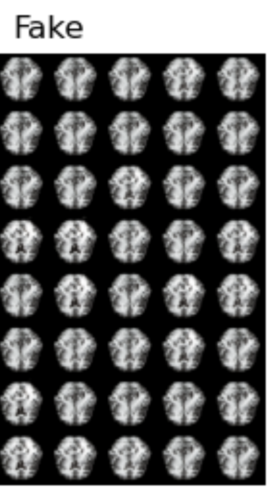
Fake



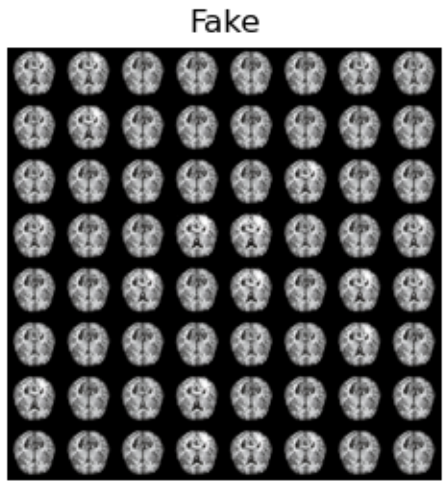
```
Train Epoch: 93 [1920/5000] D-Loss: 0.400232 G-Loss: 2.964131
Train Epoch: 93 [3968/5000] D-Loss: 0.432827 G-Loss: 5.224907
Train Epoch: 93 [6016/5000] D-Loss: 0.356236 G-Loss: 3.687192
```



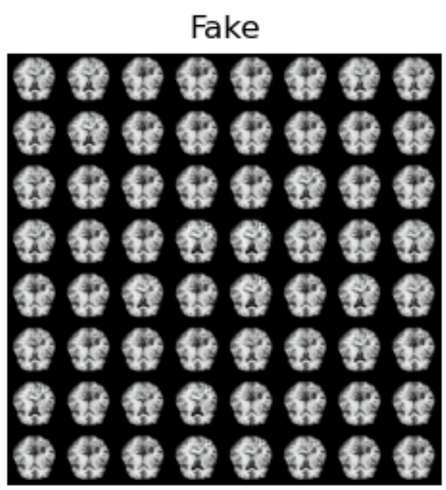

```
Train Epoch: 94 [1920/5000] D-Loss: 1.034054 G-Loss: 3.571905
Train Epoch: 94 [3968/5000] D-Loss: 0.972740 G-Loss: 5.441704
Train Epoch: 94 [6016/5000] D-Loss: 0.473469 G-Loss: 2.993345
```



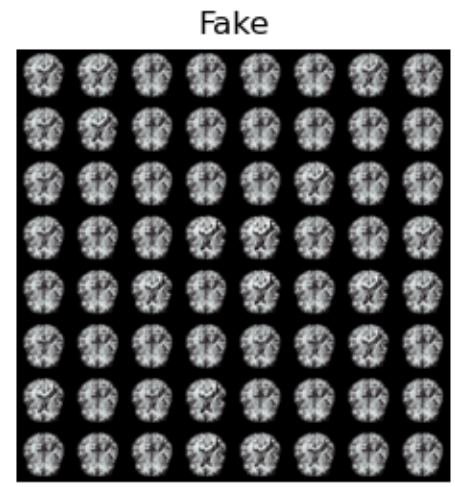

```
Train Epoch: 95 [1920/5000] D-Loss: 0.693970 G-Loss: 2.932011
Train Epoch: 95 [3968/5000] D-Loss: 0.781657 G-Loss: 2.631963
Train Epoch: 95 [6016/5000] D-Loss: 0.265479 G-Loss: 3.621550
```



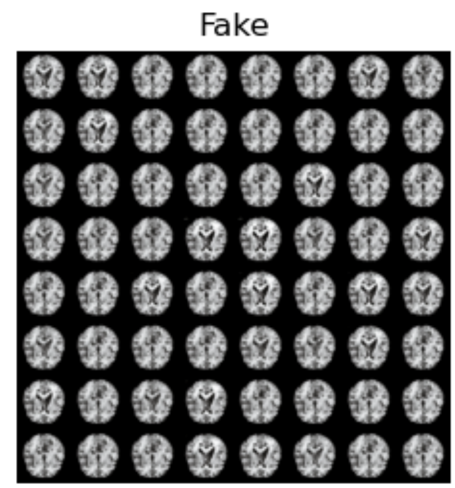
```
Train Epoch: 96 [1920/5000] D-Loss: 0.386407 G-Loss: 4.298824
Train Epoch: 96 [3968/5000] D-Loss: 0.581149 G-Loss: 4.737599
Train Epoch: 96 [6016/5000] D-Loss: 0.389731 G-Loss: 5.231346
```



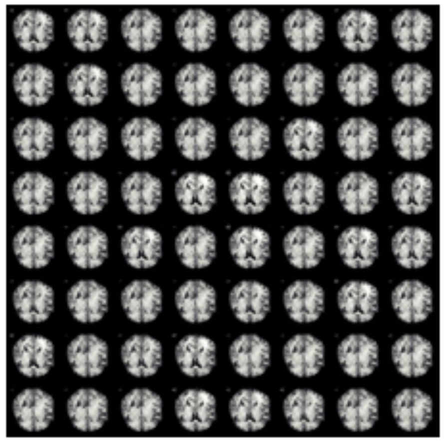
```
Train Epoch: 97 [1920/5000] D-Loss: 0.670395 G-Loss: 3.106375
Train Epoch: 97 [3968/5000] D-Loss: 0.156082 G-Loss: 3.479836
Train Epoch: 97 [6016/5000] D-Loss: 0.322117 G-Loss: 3.619271
```




```
Train Epoch: 98 [1920/5000] D-Loss: 0.301556 G-Loss: 4.272038
Train Epoch: 98 [3968/5000] D-Loss: 0.200067 G-Loss: 3.007439
Train Epoch: 98 [6016/5000] D-Loss: 0.292570 G-Loss: 5.499386
```




```
Train Epoch: 99 [1920/5000] D-Loss: 0.138774 G-Loss: 4.464682
Train Epoch: 99 [3968/5000] D-Loss: 0.405471 G-Loss: 4.986885
Train Epoch: 99 [6016/5000] D-Loss: 0.425403 G-Loss: 5.056683
```

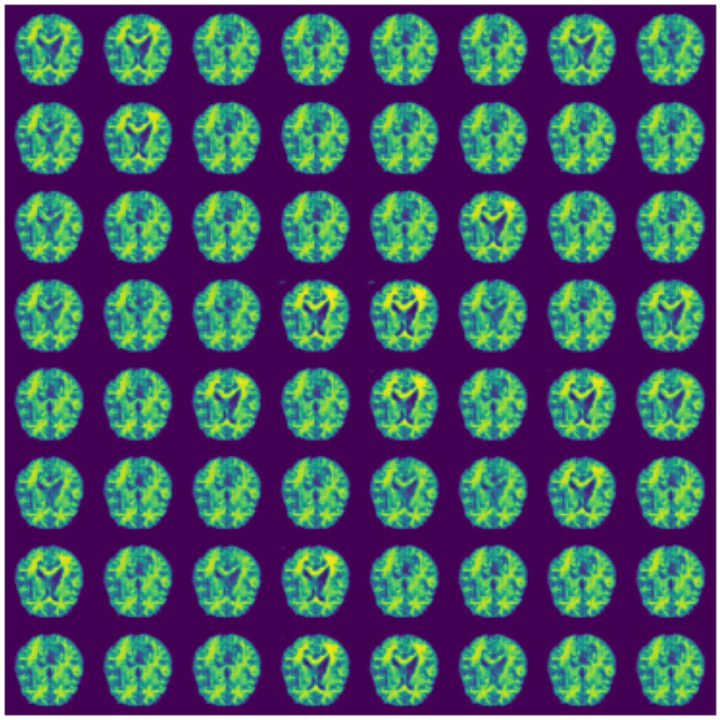



```
Train and predict complete!
```

[63]:
```
plt.axis("off")
plt.title("Fake")
pics = np.transpose(img_list[26],(1,2,0))
plt.imshow(pics[:,:,-1])
print(pics.size())
plt.show()
```

```
torch.Size([1042, 1042, 3])
```

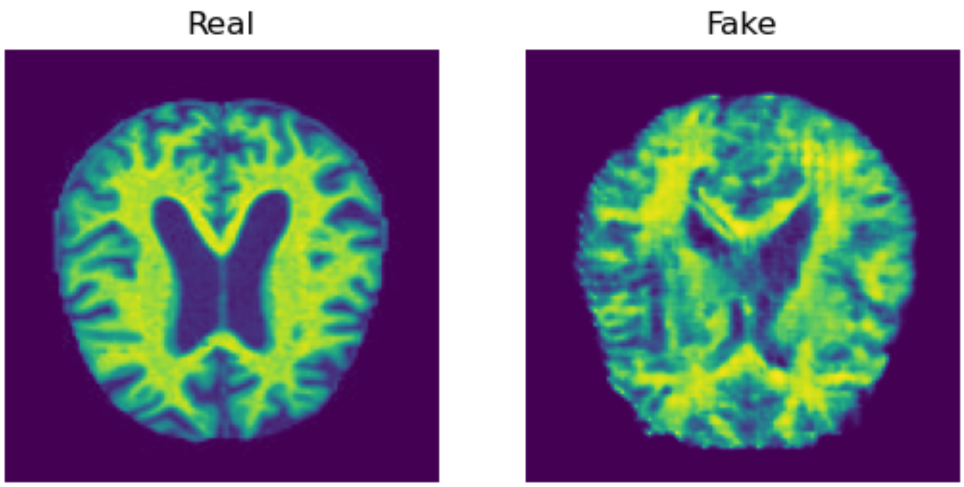
Fake



[50]:
```
from PIL import Image
fig=Image.open("./archive/Dataset/Mild_Demented/mild.jpg")
plt.subplot(1,2,1)
plt.axis("off")
plt.title("Real")
plt.imshow(fig)

plt.subplot(1,2,2)
plt.axis("off")
plt.title("Fake")
pics = np.transpose(img_list[26],(1,2,0))
plt.imshow(pics[0:128,0:128,-1])
print(pics.size())
plt.show()
```

```
torch.Size([1042, 1042, 3])
```



1 CNN

[41]:
```
import numpy as np
import sklearn
import torch
import torchvision
from torchvision import transforms
from torch.utils.data import DataLoader, random_split
from sklearn.model_selection import train_test_split
from torchvision.datasets import ImageFolder
import matplotlib.pyplot as plt

data_transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))  # Any Reason for␣
 ↪this?
])

data=ImageFolder("./archive/Dataset/", transform=data_transform)

n=len(data)
n_test=int(0.2*n)    # 20% for test
train_data,test_data=random_split(data,[n-n_test,n_test],torch.Generator().
 ↪manual_seed(42))

trainloader=DataLoader(train_data,batch_size=128,drop_last=False,shuffle=True)
testloader=DataLoader(test_data,batch_size=128,drop_last=False,shuffle=False)

import torch.nn.functional as F
```

```
class ConvNet(torch.nn.Module):
    def __init__(self):
        super(ConvNet, self).__init__()
        # Set for convolution operation
        self.conv1 = torch.nn.Sequential(
            torch.nn.Conv2d(3, 16, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
        )
        self.conv2 = torch.nn.Sequential(
            torch.nn.Conv2d(16, 32, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
        )
        self.conv3 = torch.nn.Sequential(
            torch.nn.Conv2d(32, 64, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
        )

        self.conv4 = torch.nn.Sequential(
            torch.nn.Conv2d(64, 128, 3, padding=1),
            torch.nn.ReLU(),
            torch.nn.MaxPool2d(2, 2)
        )

        self.dp = torch.nn.Dropout(p=0.5)

        self.fc1 = torch.nn.Sequential(
            torch.nn.Linear(128*8*8, 32),
            torch.nn.ReLU()
        )
        self.fc2 = torch.nn.Linear(32, 4)

    def forward(self, x):
        # Three-layer convolutional network (Conv -> ReLU -> MaxPool)
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.conv3(x)
        x = self.conv4(x)
        x = self.dp(x)
        x = x.view(-1, 128*8*8)
        x = self.fc1(x) # Fully connected layer -> ReLU
        x = self.fc2(x)
        out = F.log_softmax(x, dim=1) # Softmax probability
        return out
```

```python
net_cpu = ConvNet()
net_gpu = net_cpu.cuda()

from torch import optim
from torch.utils.tensorboard import SummaryWriter

summaryWriter = SummaryWriter("logs/lyf_cnn_gan")

optimizer = optim.Adam(
    net_gpu.parameters(),
    lr = 0.001,
    betas = (0.9, 0.999),
    eps = 1e-08,
    weight_decay = 0,
    amsgrad = False
)

loss_func = torch.nn.CrossEntropyLoss()

for epoch in range(50):
    running_loss_train = 0
    for i, data in enumerate(trainloader, 0):
        inputs_cpu, targets_cpu = data
        inputs_gpu = inputs_cpu.cuda()
        targets_gpu = targets_cpu.cuda()
        optimizer.zero_grad()
        outputs_gpu = net_gpu.train()(inputs_gpu)
        loss = loss_func(outputs_gpu, targets_gpu)
        running_loss_train += loss.item()
        loss.backward()
        optimizer.step()
        if i % 8 == 7:
            print('Train Epoch: %d [%d/5000] Loss: %.6f' %(epoch, i*64, loss.
↪item()))
    running_loss_train /= len(trainloader)
    print(running_loss_train)
    summaryWriter.add_scalar("loss", running_loss_train, epoch)

    # Step 4 Predict
    correct = 0
    total = 0
    running_loss = 0
    for data in testloader:
        images_cpu, targets_cpu = data
        images_gpu = images_cpu.cuda()
        targets_gpu = targets_cpu.cuda()
```

```
        outputs_gpu = net_gpu.eval()(images_gpu)
        _, predicted = torch.max(outputs_gpu, 1)
        loss = loss_func(outputs_gpu, targets_gpu)
        total += targets_gpu.size(0)
        running_loss += loss.item()
        correct += (predicted == targets_gpu).sum().item()

    running_loss = running_loss / 10000

    print('Test set: Average loss: %.4f, Accuracy: %d/10000 (%d%%)'␣
 ↪%(running_loss, correct, correct*100/total))
    summaryWriter.add_scalar("accuracy", correct/total, epoch)

print('Train and predict complete!')
```

```
Train Epoch: 0 [448/5000] Loss: 1.184302
Train Epoch: 0 [960/5000] Loss: 0.984301
Train Epoch: 0 [1472/5000] Loss: 1.056733
Train Epoch: 0 [1984/5000] Loss: 0.995560
Train Epoch: 0 [2496/5000] Loss: 1.166978
1.059310682117939
Test set: Average loss: 0.0010, Accuracy: 660/10000 (51%)
Train Epoch: 1 [448/5000] Loss: 0.945711
Train Epoch: 1 [960/5000] Loss: 1.042292
Train Epoch: 1 [1472/5000] Loss: 0.899025
Train Epoch: 1 [1984/5000] Loss: 1.041971
Train Epoch: 1 [2496/5000] Loss: 0.946131
0.9780660063028336
Test set: Average loss: 0.0010, Accuracy: 669/10000 (52%)
Train Epoch: 2 [448/5000] Loss: 0.912323
Train Epoch: 2 [960/5000] Loss: 0.926343
Train Epoch: 2 [1472/5000] Loss: 0.899762
Train Epoch: 2 [1984/5000] Loss: 0.842545
Train Epoch: 2 [2496/5000] Loss: 0.981674
0.9380198255181312
Test set: Average loss: 0.0009, Accuracy: 691/10000 (53%)
Train Epoch: 3 [448/5000] Loss: 0.864307
Train Epoch: 3 [960/5000] Loss: 0.909338
Train Epoch: 3 [1472/5000] Loss: 0.874002
Train Epoch: 3 [1984/5000] Loss: 0.944111
Train Epoch: 3 [2496/5000] Loss: 0.851864
0.9200391560792923
Test set: Average loss: 0.0009, Accuracy: 709/10000 (55%)
Train Epoch: 4 [448/5000] Loss: 0.837277
Train Epoch: 4 [960/5000] Loss: 1.058367
Train Epoch: 4 [1472/5000] Loss: 0.912683
Train Epoch: 4 [1984/5000] Loss: 0.760248
Train Epoch: 4 [2496/5000] Loss: 0.877468
```

```
0.9060287520289421
Test set: Average loss: 0.0009, Accuracy: 721/10000 (56%)
Train Epoch: 5 [448/5000] Loss: 0.833565
Train Epoch: 5 [960/5000] Loss: 0.918321
Train Epoch: 5 [1472/5000] Loss: 0.896223
Train Epoch: 5 [1984/5000] Loss: 0.902787
Train Epoch: 5 [2496/5000] Loss: 0.912368
0.8880569741129876
Test set: Average loss: 0.0009, Accuracy: 736/10000 (57%)
Train Epoch: 6 [448/5000] Loss: 0.864245
Train Epoch: 6 [960/5000] Loss: 0.883355
Train Epoch: 6 [1472/5000] Loss: 0.863943
Train Epoch: 6 [1984/5000] Loss: 0.955054
Train Epoch: 6 [2496/5000] Loss: 0.758850
0.8699227303266526
Test set: Average loss: 0.0009, Accuracy: 749/10000 (58%)
Train Epoch: 7 [448/5000] Loss: 0.888258
Train Epoch: 7 [960/5000] Loss: 0.811115
Train Epoch: 7 [1472/5000] Loss: 0.863958
Train Epoch: 7 [1984/5000] Loss: 0.812749
Train Epoch: 7 [2496/5000] Loss: 0.800268
0.8476858749985695
Test set: Average loss: 0.0009, Accuracy: 768/10000 (60%)
Train Epoch: 8 [448/5000] Loss: 0.799748
Train Epoch: 8 [960/5000] Loss: 0.989671
Train Epoch: 8 [1472/5000] Loss: 0.777110
Train Epoch: 8 [1984/5000] Loss: 0.809748
Train Epoch: 8 [2496/5000] Loss: 0.817938
0.8316218376159668
Test set: Average loss: 0.0008, Accuracy: 770/10000 (60%)
Train Epoch: 9 [448/5000] Loss: 0.760700
Train Epoch: 9 [960/5000] Loss: 0.760496
Train Epoch: 9 [1472/5000] Loss: 0.904626
Train Epoch: 9 [1984/5000] Loss: 0.830887
Train Epoch: 9 [2496/5000] Loss: 0.707649
0.8040266409516335
Test set: Average loss: 0.0008, Accuracy: 800/10000 (62%)
Train Epoch: 10 [448/5000] Loss: 0.757495
Train Epoch: 10 [960/5000] Loss: 0.808893
Train Epoch: 10 [1472/5000] Loss: 0.805673
Train Epoch: 10 [1984/5000] Loss: 0.750111
Train Epoch: 10 [2496/5000] Loss: 0.764039
0.7662643864750862
Test set: Average loss: 0.0008, Accuracy: 825/10000 (64%)
Train Epoch: 11 [448/5000] Loss: 0.656272
Train Epoch: 11 [960/5000] Loss: 0.727260
Train Epoch: 11 [1472/5000] Loss: 0.721524
Train Epoch: 11 [1984/5000] Loss: 0.763966
```

```
Train Epoch: 11 [2496/5000] Loss: 0.728280
0.726995313167572
Test set: Average loss: 0.0007, Accuracy: 853/10000 (66%)
Train Epoch: 12 [448/5000] Loss: 0.632288
Train Epoch: 12 [960/5000] Loss: 0.731929
Train Epoch: 12 [1472/5000] Loss: 0.673158
Train Epoch: 12 [1984/5000] Loss: 0.532014
Train Epoch: 12 [2496/5000] Loss: 0.635704
0.7005811288952828
Test set: Average loss: 0.0008, Accuracy: 831/10000 (64%)
Train Epoch: 13 [448/5000] Loss: 0.611317
Train Epoch: 13 [960/5000] Loss: 0.699233
Train Epoch: 13 [1472/5000] Loss: 0.777902
Train Epoch: 13 [1984/5000] Loss: 0.589422
Train Epoch: 13 [2496/5000] Loss: 0.718228
0.6539036884903908
Test set: Average loss: 0.0006, Accuracy: 926/10000 (72%)
Train Epoch: 14 [448/5000] Loss: 0.631462
Train Epoch: 14 [960/5000] Loss: 0.637204
Train Epoch: 14 [1472/5000] Loss: 0.567665
Train Epoch: 14 [1984/5000] Loss: 0.596656
Train Epoch: 14 [2496/5000] Loss: 0.577692
0.6039270639419556
Test set: Average loss: 0.0006, Accuracy: 922/10000 (72%)
Train Epoch: 15 [448/5000] Loss: 0.616755
Train Epoch: 15 [960/5000] Loss: 0.609204
Train Epoch: 15 [1472/5000] Loss: 0.661993
Train Epoch: 15 [1984/5000] Loss: 0.542730
Train Epoch: 15 [2496/5000] Loss: 0.698361
0.5810859590768814
Test set: Average loss: 0.0006, Accuracy: 935/10000 (73%)
Train Epoch: 16 [448/5000] Loss: 0.553278
Train Epoch: 16 [960/5000] Loss: 0.489031
Train Epoch: 16 [1472/5000] Loss: 0.574645
Train Epoch: 16 [1984/5000] Loss: 0.517660
Train Epoch: 16 [2496/5000] Loss: 0.480100
0.5046333640813827
Test set: Average loss: 0.0005, Accuracy: 1005/10000 (78%)
Train Epoch: 17 [448/5000] Loss: 0.489037
Train Epoch: 17 [960/5000] Loss: 0.430146
Train Epoch: 17 [1472/5000] Loss: 0.542541
Train Epoch: 17 [1984/5000] Loss: 0.491605
Train Epoch: 17 [2496/5000] Loss: 0.487316
0.45310535058379175
Test set: Average loss: 0.0005, Accuracy: 1027/10000 (80%)
Train Epoch: 18 [448/5000] Loss: 0.417342
Train Epoch: 18 [960/5000] Loss: 0.323239
Train Epoch: 18 [1472/5000] Loss: 0.420276
```

```
Train Epoch: 18 [1984/5000] Loss: 0.492126
Train Epoch: 18 [2496/5000] Loss: 0.378779
0.41633934155106544
Test set: Average loss: 0.0004, Accuracy: 1075/10000 (83%)
Train Epoch: 19 [448/5000] Loss: 0.390002
Train Epoch: 19 [960/5000] Loss: 0.310708
Train Epoch: 19 [1472/5000] Loss: 0.403030
Train Epoch: 19 [1984/5000] Loss: 0.332799
Train Epoch: 19 [2496/5000] Loss: 0.360852
0.3614169344305992
Test set: Average loss: 0.0004, Accuracy: 1090/10000 (85%)
Train Epoch: 20 [448/5000] Loss: 0.309517
Train Epoch: 20 [960/5000] Loss: 0.307054
Train Epoch: 20 [1472/5000] Loss: 0.253412
Train Epoch: 20 [1984/5000] Loss: 0.401284
Train Epoch: 20 [2496/5000] Loss: 0.349469
0.325675667822361
Test set: Average loss: 0.0003, Accuracy: 1135/10000 (88%)
Train Epoch: 21 [448/5000] Loss: 0.191465
Train Epoch: 21 [960/5000] Loss: 0.351276
Train Epoch: 21 [1472/5000] Loss: 0.357560
Train Epoch: 21 [1984/5000] Loss: 0.199869
Train Epoch: 21 [2496/5000] Loss: 0.286298
0.2965821385383606
Test set: Average loss: 0.0003, Accuracy: 1139/10000 (88%)
Train Epoch: 22 [448/5000] Loss: 0.292842
Train Epoch: 22 [960/5000] Loss: 0.256483
Train Epoch: 22 [1472/5000] Loss: 0.211014
Train Epoch: 22 [1984/5000] Loss: 0.286241
Train Epoch: 22 [2496/5000] Loss: 0.210229
0.25308653749525545
Test set: Average loss: 0.0003, Accuracy: 1167/10000 (91%)
Train Epoch: 23 [448/5000] Loss: 0.258938
Train Epoch: 23 [960/5000] Loss: 0.219477
Train Epoch: 23 [1472/5000] Loss: 0.246418
Train Epoch: 23 [1984/5000] Loss: 0.223956
Train Epoch: 23 [2496/5000] Loss: 0.315431
0.24585194177925587
Test set: Average loss: 0.0003, Accuracy: 1134/10000 (88%)
Train Epoch: 24 [448/5000] Loss: 0.128648
Train Epoch: 24 [960/5000] Loss: 0.208040
Train Epoch: 24 [1472/5000] Loss: 0.251287
Train Epoch: 24 [1984/5000] Loss: 0.231845
Train Epoch: 24 [2496/5000] Loss: 0.210724
0.20744291078299285
Test set: Average loss: 0.0002, Accuracy: 1192/10000 (93%)
Train Epoch: 25 [448/5000] Loss: 0.124385
Train Epoch: 25 [960/5000] Loss: 0.139095
```

```
Train Epoch: 25 [1472/5000] Loss: 0.170643
Train Epoch: 25 [1984/5000] Loss: 0.161802
Train Epoch: 25 [2496/5000] Loss: 0.166897
0.1709763055667281
Test set: Average loss: 0.0002, Accuracy: 1199/10000 (93%)
Train Epoch: 26 [448/5000] Loss: 0.171886
Train Epoch: 26 [960/5000] Loss: 0.166273
Train Epoch: 26 [1472/5000] Loss: 0.181545
Train Epoch: 26 [1984/5000] Loss: 0.263531
Train Epoch: 26 [2496/5000] Loss: 0.175125
0.18599466886371374
Test set: Average loss: 0.0002, Accuracy: 1196/10000 (93%)
Train Epoch: 27 [448/5000] Loss: 0.100569
Train Epoch: 27 [960/5000] Loss: 0.171190
Train Epoch: 27 [1472/5000] Loss: 0.151292
Train Epoch: 27 [1984/5000] Loss: 0.092879
Train Epoch: 27 [2496/5000] Loss: 0.190890
0.1669236382469535
Test set: Average loss: 0.0002, Accuracy: 1213/10000 (94%)
Train Epoch: 28 [448/5000] Loss: 0.112136
Train Epoch: 28 [960/5000] Loss: 0.191809
Train Epoch: 28 [1472/5000] Loss: 0.097219
Train Epoch: 28 [1984/5000] Loss: 0.079168
Train Epoch: 28 [2496/5000] Loss: 0.127292
0.1419488213956356
Test set: Average loss: 0.0002, Accuracy: 1217/10000 (95%)
Train Epoch: 29 [448/5000] Loss: 0.159782
Train Epoch: 29 [960/5000] Loss: 0.120593
Train Epoch: 29 [1472/5000] Loss: 0.128640
Train Epoch: 29 [1984/5000] Loss: 0.134191
Train Epoch: 29 [2496/5000] Loss: 0.128223
0.136030288413167
Test set: Average loss: 0.0001, Accuracy: 1214/10000 (94%)
Train Epoch: 30 [448/5000] Loss: 0.143697
Train Epoch: 30 [960/5000] Loss: 0.112254
Train Epoch: 30 [1472/5000] Loss: 0.098951
Train Epoch: 30 [1984/5000] Loss: 0.075037
Train Epoch: 30 [2496/5000] Loss: 0.062157
0.11880059763789177
Test set: Average loss: 0.0002, Accuracy: 1224/10000 (95%)
Train Epoch: 31 [448/5000] Loss: 0.205229
Train Epoch: 31 [960/5000] Loss: 0.091153
Train Epoch: 31 [1472/5000] Loss: 0.079601
Train Epoch: 31 [1984/5000] Loss: 0.098154
Train Epoch: 31 [2496/5000] Loss: 0.141218
0.10056419987231494
Test set: Average loss: 0.0001, Accuracy: 1230/10000 (96%)
Train Epoch: 32 [448/5000] Loss: 0.082792
```

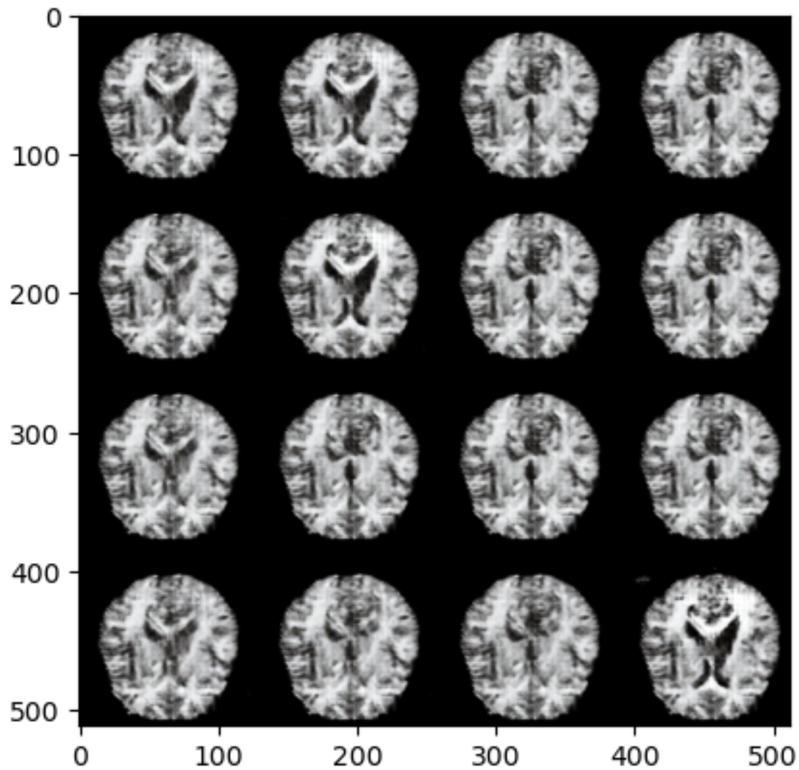
```
Train Epoch: 32 [960/5000] Loss: 0.069535
Train Epoch: 32 [1472/5000] Loss: 0.104985
Train Epoch: 32 [1984/5000] Loss: 0.039782
Train Epoch: 32 [2496/5000] Loss: 0.053772
0.0933999934233725
Test set: Average loss: 0.0001, Accuracy: 1228/10000 (95%)
Train Epoch: 33 [448/5000] Loss: 0.112814
Train Epoch: 33 [960/5000] Loss: 0.041989
Train Epoch: 33 [1472/5000] Loss: 0.086726
Train Epoch: 33 [1984/5000] Loss: 0.055735
Train Epoch: 33 [2496/5000] Loss: 0.101166
0.08994227480143309
Test set: Average loss: 0.0001, Accuracy: 1224/10000 (95%)
Train Epoch: 34 [448/5000] Loss: 0.137720
Train Epoch: 34 [960/5000] Loss: 0.082036
Train Epoch: 34 [1472/5000] Loss: 0.051996
Train Epoch: 34 [1984/5000] Loss: 0.102107
Train Epoch: 34 [2496/5000] Loss: 0.060881
0.09549545948393642
Test set: Average loss: 0.0001, Accuracy: 1241/10000 (96%)
Train Epoch: 35 [448/5000] Loss: 0.158613
Train Epoch: 35 [960/5000] Loss: 0.073481
Train Epoch: 35 [1472/5000] Loss: 0.096473
Train Epoch: 35 [1984/5000] Loss: 0.077460
Train Epoch: 35 [2496/5000] Loss: 0.073719
0.0923249644227326
Test set: Average loss: 0.0001, Accuracy: 1235/10000 (96%)
Train Epoch: 36 [448/5000] Loss: 0.061810
Train Epoch: 36 [960/5000] Loss: 0.101465
Train Epoch: 36 [1472/5000] Loss: 0.110336
Train Epoch: 36 [1984/5000] Loss: 0.081915
Train Epoch: 36 [2496/5000] Loss: 0.117267
0.092028793040663
Test set: Average loss: 0.0001, Accuracy: 1224/10000 (95%)
Train Epoch: 37 [448/5000] Loss: 0.067768
Train Epoch: 37 [960/5000] Loss: 0.108568
Train Epoch: 37 [1472/5000] Loss: 0.049320
Train Epoch: 37 [1984/5000] Loss: 0.088591
Train Epoch: 37 [2496/5000] Loss: 0.085546
0.07804194740019739
Test set: Average loss: 0.0001, Accuracy: 1234/10000 (96%)
Train Epoch: 38 [448/5000] Loss: 0.070747
Train Epoch: 38 [960/5000] Loss: 0.103866
Train Epoch: 38 [1472/5000] Loss: 0.086631
Train Epoch: 38 [1984/5000] Loss: 0.054096
Train Epoch: 38 [2496/5000] Loss: 0.052354
0.07120829951018096
Test set: Average loss: 0.0001, Accuracy: 1231/10000 (96%)
```

```
Train Epoch: 39 [448/5000] Loss: 0.030448
Train Epoch: 39 [960/5000] Loss: 0.068512
Train Epoch: 39 [1472/5000] Loss: 0.114550
Train Epoch: 39 [1984/5000] Loss: 0.111102
Train Epoch: 39 [2496/5000] Loss: 0.109439
0.06324709001928568
Test set: Average loss: 0.0001, Accuracy: 1242/10000 (97%)
Train Epoch: 40 [448/5000] Loss: 0.036648
Train Epoch: 40 [960/5000] Loss: 0.075790
Train Epoch: 40 [1472/5000] Loss: 0.116648
Train Epoch: 40 [1984/5000] Loss: 0.062506
Train Epoch: 40 [2496/5000] Loss: 0.111627
0.06868363907560707
Test set: Average loss: 0.0001, Accuracy: 1235/10000 (96%)
Train Epoch: 41 [448/5000] Loss: 0.065187
Train Epoch: 41 [960/5000] Loss: 0.077176
Train Epoch: 41 [1472/5000] Loss: 0.046705
Train Epoch: 41 [1984/5000] Loss: 0.109271
Train Epoch: 41 [2496/5000] Loss: 0.072352
0.09241930777207016
Test set: Average loss: 0.0001, Accuracy: 1231/10000 (96%)
Train Epoch: 42 [448/5000] Loss: 0.049061
Train Epoch: 42 [960/5000] Loss: 0.051872
Train Epoch: 42 [1472/5000] Loss: 0.055846
Train Epoch: 42 [1984/5000] Loss: 0.041589
Train Epoch: 42 [2496/5000] Loss: 0.054174
0.05810639970004559
Test set: Average loss: 0.0001, Accuracy: 1239/10000 (96%)
Train Epoch: 43 [448/5000] Loss: 0.050837
Train Epoch: 43 [960/5000] Loss: 0.021840
Train Epoch: 43 [1472/5000] Loss: 0.055730
Train Epoch: 43 [1984/5000] Loss: 0.038553
Train Epoch: 43 [2496/5000] Loss: 0.043159
0.05420882140751928
Test set: Average loss: 0.0001, Accuracy: 1240/10000 (96%)
Train Epoch: 44 [448/5000] Loss: 0.134087
Train Epoch: 44 [960/5000] Loss: 0.033199
Train Epoch: 44 [1472/5000] Loss: 0.060805
Train Epoch: 44 [1984/5000] Loss: 0.045961
Train Epoch: 44 [2496/5000] Loss: 0.062778
0.0632667808327824
Test set: Average loss: 0.0001, Accuracy: 1245/10000 (97%)
Train Epoch: 45 [448/5000] Loss: 0.040096
Train Epoch: 45 [960/5000] Loss: 0.040820
Train Epoch: 45 [1472/5000] Loss: 0.071376
Train Epoch: 45 [1984/5000] Loss: 0.042697
Train Epoch: 45 [2496/5000] Loss: 0.063618
0.05118083970155567
```

```
Test set: Average loss: 0.0001, Accuracy: 1245/10000 (97%)
Train Epoch: 46 [448/5000] Loss: 0.023300
Train Epoch: 46 [960/5000] Loss: 0.069528
Train Epoch: 46 [1472/5000] Loss: 0.030252
Train Epoch: 46 [1984/5000] Loss: 0.089136
Train Epoch: 46 [2496/5000] Loss: 0.065510
0.05270845163613558
Test set: Average loss: 0.0001, Accuracy: 1240/10000 (96%)
Train Epoch: 47 [448/5000] Loss: 0.038934
Train Epoch: 47 [960/5000] Loss: 0.053103
Train Epoch: 47 [1472/5000] Loss: 0.069452
Train Epoch: 47 [1984/5000] Loss: 0.044644
Train Epoch: 47 [2496/5000] Loss: 0.044490
0.057501562498509885
Test set: Average loss: 0.0001, Accuracy: 1246/10000 (97%)
Train Epoch: 48 [448/5000] Loss: 0.033504
Train Epoch: 48 [960/5000] Loss: 0.040560
Train Epoch: 48 [1472/5000] Loss: 0.045658
Train Epoch: 48 [1984/5000] Loss: 0.013463
Train Epoch: 48 [2496/5000] Loss: 0.018631
0.047174057306256144
Test set: Average loss: 0.0001, Accuracy: 1244/10000 (97%)
Train Epoch: 49 [448/5000] Loss: 0.072717
Train Epoch: 49 [960/5000] Loss: 0.019069
Train Epoch: 49 [1472/5000] Loss: 0.042559
Train Epoch: 49 [1984/5000] Loss: 0.049219
Train Epoch: 49 [2496/5000] Loss: 0.064374
0.05253603085875511
Test set: Average loss: 0.0001, Accuracy: 1251/10000 (97%)
Train and predict complete!
```

[61]:
```python
pics = np.transpose(img_list[26],(1,2,0))
pic1 = pics[:512,:512,:3]
plt.imshow(pic1)
print(pic1.shape)
```

```
torch.Size([512, 512, 3])
```



[96]:
```
# save figures
for i in range(8):
    for j in range(8):
        pic1=np.transpose(img_list[26],(1,2,0))
        pic1=pic1[128*i:128*i+128,128*j:128*j+128,:3]
        plt.imsave(f'./gan_pic/0/{8*i+j}.png',pic1.numpy())

        pic2=np.transpose(img_list[27],(1,2,0))
        pic2=pic2[128*i:128*i+128,128*j:128*j+128,:3]
        plt.imsave(f'./gan_pic/0/{8*i+j+64}.png',pic2.numpy())
```

[97]:
```
gan_data=ImageFolder("./gan_pic/", transform=data_transform)
gan_testloader=DataLoader(gan_data,batch_size=128,drop_last=False,shuffle=False)

types = ['Mild', 'Moderate', 'Non','Very Mild']
predicted = []
result = []
for data in gan_testloader:
    images_cpu, targets_cpu = data
    images_gpu = images_cpu.cuda()
    targets_gpu = targets_cpu.cuda()
    outputs_gpu = net_gpu.eval()(images_gpu)
```

```
    _, predicted = torch.max(outputs_gpu, 1)
    predicted=predicted.cpu().numpy()
    print(predicted)
    result = [types[i] for i in predicted]
    print(result)
```

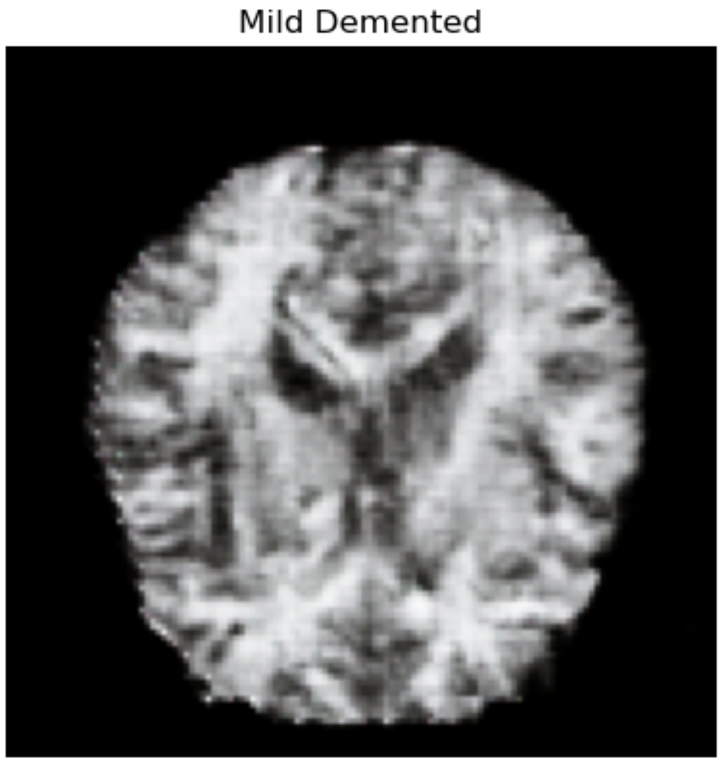
```
[2 2 2 0 2 3 3 3 2 2 2 0 2 2 2 2 0 2 3 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 3 2 2
 2 3 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 3 0 2 2 3 2 2 3 2]
['Non', 'Non', 'Non', 'Mild', 'Non', 'Very Mild', 'Very Mild', 'Very Mild',
'Non', 'Non', 'Non', 'Mild', 'Non', 'Non', 'Non', 'Non', 'Mild', 'Non', 'Very
Mild', 'Mild', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non',
'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non',
'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non',
'Non', 'Non', 'Very Mild', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non',
'Very Mild', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non',
'Very Mild', 'Non', 'Non', 'Non', 'Very Mild', 'Non', 'Non', 'Non', 'Non', 'Very
Mild', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non',
'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non',
'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Non',
'Non', 'Non', 'Non', 'Non', 'Non', 'Non', 'Very Mild', 'Mild', 'Non', 'Non',
'Very Mild', 'Non', 'Non', 'Very Mild', 'Non']
```

[98]:
```
print(np.where(predicted==0))
print(np.where(predicted==1))
print(np.where(predicted==2))
print(np.where(predicted==3))
```

```
(array([  3,  11,  16,  19, 120], dtype=int64),)
(array([], dtype=int64),)
(array([  0,   1,   2,   4,   8,   9,  10,  12,  13,  14,  15,  17,  20,
        21,  22,  23,  24,  25,  26,  27,  28,  29,  30,  31,  32,  33,
        34,  35,  36,  37,  38,  39,  40,  41,  42,  43,  44,  45,  46,
        47,  48,  49,  50,  51,  52,  54,  55,  56,  57,  58,  59,  60,
        62,  63,  64,  65,  66,  67,  68,  69,  70,  72,  73,  74,  76,
        77,  78,  79,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 121, 122, 124, 125, 127], dtype=int64),)
(array([  5,   6,   7,  18,  53,  61,  71,  75,  80, 119, 123, 126],
      dtype=int64),)
```

[106]:
```
fig=Image.open("./gan_pic/0/16.png")
plt.axis("off")
plt.title("Mild Demented")
plt.imshow(fig)
```

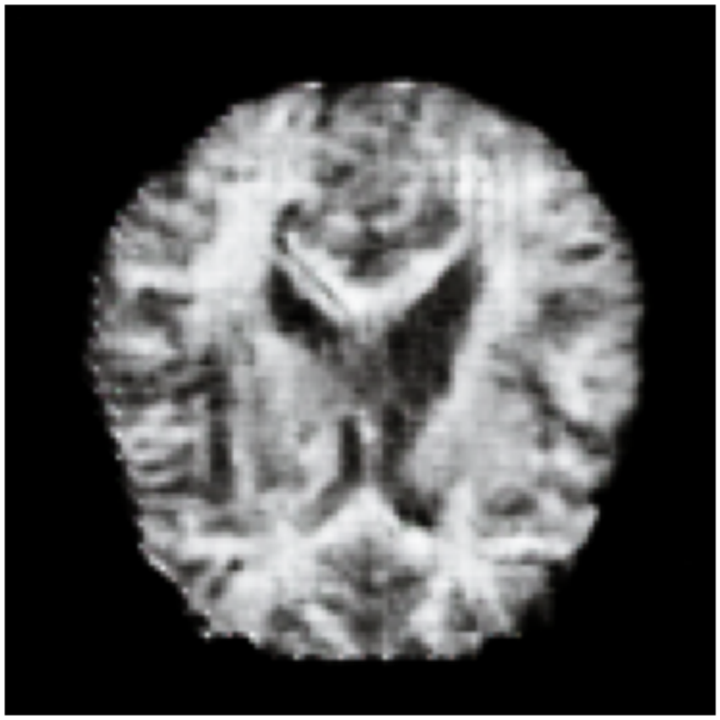
[106]: <matplotlib.image.AxesImage at 0x25d86615388>



[100]:
```
fig=Image.open("./gan_pic/0/0.png")
plt.axis("off")
plt.title("Non Demented")
plt.imshow(fig)
```

[100]: <matplotlib.image.AxesImage at 0x25d80d5c688>

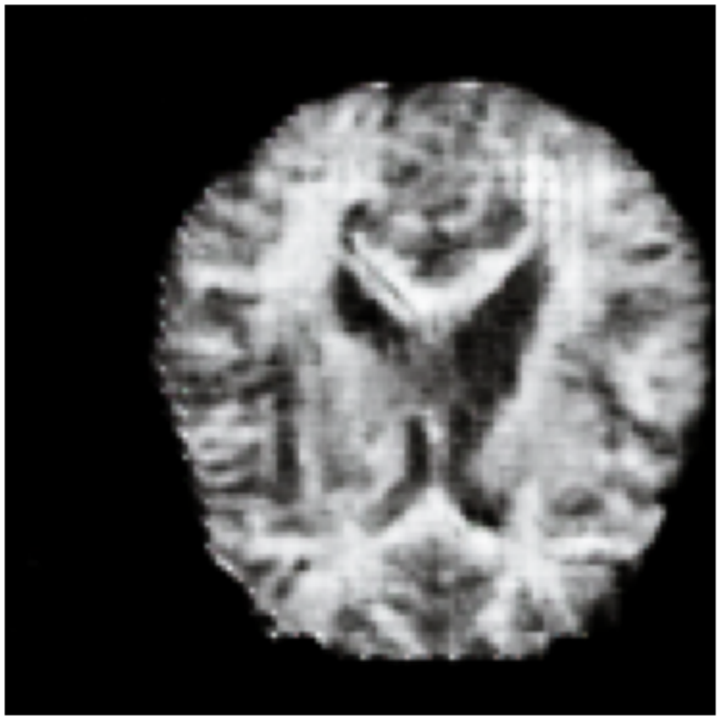
Non Demented



```
[109]: fig=Image.open("./gan_pic/0/6.png")
       plt.axis("off")
       plt.title("Very Mild Demented")
       plt.imshow(fig)
```

```
[109]: <matplotlib.image.AxesImage at 0x25d86463448>
```

Very Mild Demented



[ ]: