Implementation of famous research paper (SIAMESE NETWORK) in pytorch [PAPER TO PROJECT]

1) Importing important libraries

```
import torch.nn as nn
import torch .nn.functional as F
from torch.utils.data import DataLoader,TensorDataset
from sklearn.model_selection import train_test_split

!pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\chiranjeet\
anaconda3\lib\site-packages (4.9.0.80)
Requirement already satisfied: numpy>=1.21.2 in c:\users\chiranjeet\
anaconda3\lib\site-packages (from opencv-python) (1.24.3)
```

```
import numpy as np
import matplotlib.pyplot as plt
import random
import cv2
```

2) Creating Directories using "os"

```
POS_PATH=os.path.join('data','positives')
NEG_PATH=os.path.join('data','negatives')
ARCH_PATH=os.path.join('data','archs')

os.makedirs(POS_PATH,exist_ok=True)
os.makedirs(NEG_PATH,exist_ok=True)
os.makedirs(ARCH_PATH,exist_ok=True)

POS_PATH
```

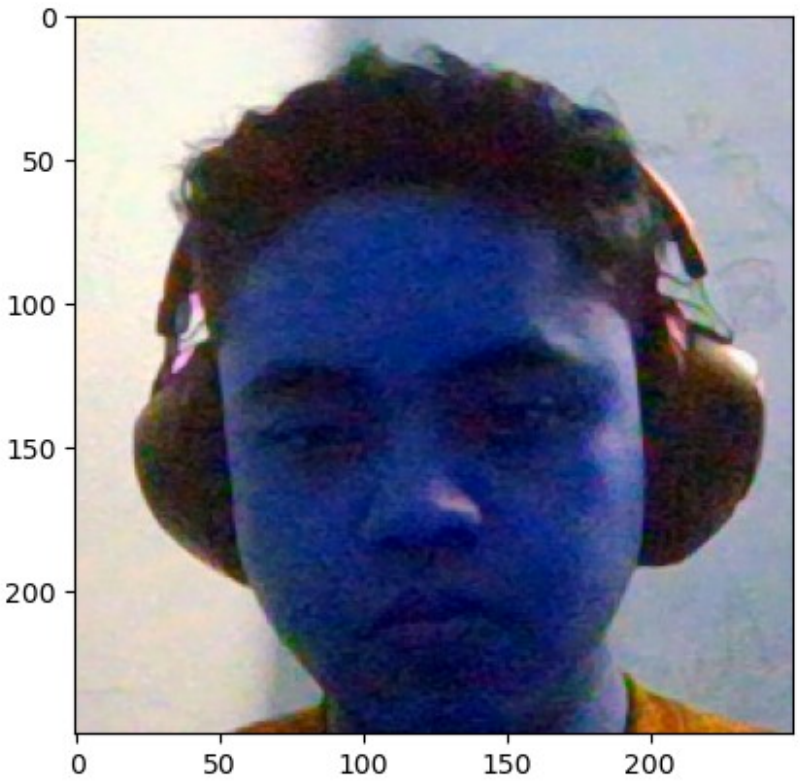
```
'data\\positives'
```

3) Giving universal unique identifier(uuid) to each image in files basically to identify each image uniquely and importing them into specific folders at the same time using different trigger keys

```
import uuid

cap=cv2.VideoCapture(0)
while(True):
```

```
    ret,frame=cap.read()
    cv2.imshow('Image',frame[150:150+250,220:220+250,:])
    if cv2.waitKey(1)&0XFF==ord('p'):
        imagename=os.path.join(POS_PATH,'{}.jpg'.format(uuid.uuid1()))
        cv2.imwrite(imagename,frame)
    if cv2.waitKey(1)&0XFF==ord('a'):

imagename=os.path.join(ARCH_PATH,'{}.jpg'.format(uuid.uuid1()))
        cv2.imwrite(imagename,frame)
    if cv2.waitKey(1)&0XFF==ord('q'):
        break
cap.release()
cv2.destroyAllWindows()

plt.imshow(frame[150:150+250,220:220+250,:])
```

```
<matplotlib.image.AxesImage at 0x1cffdcfc0d0>
```



4) Unzipping the negatives that we have downloaded and transferring to previously created files

```
!tar xf lfw.tgz

for directory in os.listdir('lfw'):
    directory_path=os.path.join('lfw',directory)
```

```
    for file in os.listdir(directory_path):
        EX_PATH=os.path.join('lfw',directory,file)
        NEW_PATH=os.path.join(NEG_PATH,file)
        os.replace(EX_PATH,NEW_PATH)
```

5) Importing data from "RESEARCH" directory

```
import torch
from torch.utils.data import DataLoader
from torchvision import datasets,transforms

target_size=(250,250)
path=r'D:\RESEARCH\data'

data=datasets.ImageFolder(root=path,transform=transforms.Compose([tran
sforms.Resize(target_size),transforms.ToTensor(),]))



data
```

```
Dataset ImageFolder
    Number of datapoints: 900
    Root location: D:\RESEARCH\data
    StandardTransform
Transform: Compose(
               Resize(size=(250, 250), interpolation=bilinear,
max_size=None, antialias=warn)
               ToTensor()
           )
```

```
class_labels = data.classes
class_names = [name.split("\\")[-1] for name in class_labels]  #
Extract class names from paths

# Print the available classes
print("Available Classes:", class_names)
```

```
Available Classes: ['archs', 'negatives', 'positives']
```

6) Partitioning the data and bacthcing it for training

```
info_anchor=[]
info_neg=[]
info_pos=[]
for x,y in data:
    if y==0:
        info_anchor.append(x[0])
    elif y==1:
        info_neg.append(x[0])
```

```
    else:
        info_pos.append(x[0])

len(info_anchor)
```

```
300
```

```
info_0_tensor = torch.zeros(300, 1, 250, 250)
info_1_tensor = torch.zeros(300, 1, 250, 250)
info_2_tensor = torch.zeros(300, 1, 250, 250)
label_0_tensor = torch.zeros(300)
label_1_tensor = torch.zeros(300)
label_2_tensor = torch.zeros(300)

for i in range(300):
    info_0_tensor[i, 0, :, :] = torch.tensor(info_anchor[i]).view(1,
250, 250)
    info_1_tensor[i, 0, :, :] = torch.tensor(info_neg[i]).view(1, 250,
250)
    info_2_tensor[i, 0, :, :] = torch.tensor(info_pos[i]).view(1, 250,
250)

    label_0_tensor[i] = torch.tensor([0])
    label_1_tensor[i] = torch.tensor([1])
    label_2_tensor[i] = torch.tensor([0])


train_data,test_data,train_label,test_label=train_test_split(info_0_te
nsor,label_0_tensor,train_size=0.9)
train_0_pytorch=TensorDataset(train_data,train_label)
test_0_pytorch=TensorDataset(test_data,test_label)
train_0_loader=DataLoader(train_0_pytorch,shuffle=True,batch_size=32,d
rop_last=True)
test_0_loader=DataLoader(train_0_pytorch,shuffle=True,batch_size=32,dr
op_last=True)

train_data,test_data,train_label,test_label=train_test_split(info_1_te
nsor,label_1_tensor,train_size=0.9)
train_1_pytorch=TensorDataset(train_data,train_label)
test_1_pytorch=TensorDataset(test_data,test_label)
train_1_loader=DataLoader(train_1_pytorch,shuffle=True,batch_size=32,d
rop_last=True)
test_1_loader=DataLoader(train_1_pytorch,shuffle=True,batch_size=32,dr
op_last=True)

train_data,test_data,train_label,test_label=train_test_split(info_2_te
nsor,label_2_tensor,train_size=0.9)
train_2_pytorch=TensorDataset(train_data,train_label)
test_2_pytorch=TensorDataset(test_data,test_label)
```

```
train_2_loader=DataLoader(train_2_pytorch,shuffle=True,batch_size=32,d
rop_last=True)
test_2_loader=DataLoader(train_2_pytorch,shuffle=True,batch_size=32,dr
op_last=True)
```

```
C:\Users\Chiranjeet\AppData\Local\Temp\
ipykernel_22188\2254438236.py:9: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  info_0_tensor[i, 0, :, :] = torch.tensor(info_anchor[i]).view(1,
250, 250)
C:\Users\Chiranjeet\AppData\Local\Temp\
ipykernel_22188\2254438236.py:10: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  info_1_tensor[i, 0, :, :] = torch.tensor(info_neg[i]).view(1, 250,
250)
C:\Users\Chiranjeet\AppData\Local\Temp\
ipykernel_22188\2254438236.py:11: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  info_2_tensor[i, 0, :, :] = torch.tensor(info_pos[i]).view(1, 250,
250)
```

```
for x,y in train_0_loader:
    print(x.shape,y.shape)
```

```
torch.Size([32, 1, 250, 250]) torch.Size([32])
torch.Size([32, 1, 250, 250]) torch.Size([32])
torch.Size([32, 1, 250, 250]) torch.Size([32])
torch.Size([32, 1, 250, 250]) torch.Size([32])
torch.Size([32, 1, 250, 250]) torch.Size([32])
torch.Size([32, 1, 250, 250]) torch.Size([32])
torch.Size([32, 1, 250, 250]) torch.Size([32])
torch.Size([32, 1, 250, 250]) torch.Size([32])
```

7) Creating user defined ContrastiveLoss mentioned in the paper

```
#loss fuction definition
class ContrastiveLoss(torch.nn.Module):
    def __init__ (self,margin=2.0):
        super(ContrastiveLoss,self).__init__()
        #defining parameters
        self.margin =margin

    def forward(self,output1,output2,label):
        #calculate the euclidean distance and calculate the contastive
```

```
loss
        euclidean_distance =
F.pairwise_distance(output1,output2,keepdim = True)
        loss_contrastive =torch.mean((1-
label)*torch.pow(euclidean_distance,2)+
        (label)*torch.pow(torch.clamp(self.margin-
euclidean_distance,min=0.0),2))

        return loss_contrastive
```

8) Creating working SIAMESE network architecture by getting ideas from the paper

```
#Saemese Network

import torch
import torch.nn as nn

class Siamese(nn.Module):
    class arc(nn.Module):
        def __init__(self):
            super().__init__()
            ### Convolutional layers
            self.cnn1 = nn.Sequential(
                nn.Conv2d(1, 96, kernel_size=3, stride=4),
                nn.ReLU(),
                nn.MaxPool2d(3, stride=2),
                nn.Conv2d(96, 256, kernel_size=5, stride=1),
                nn.ReLU(),
                nn.MaxPool2d(2, stride=2),
                nn.Conv2d(256, 384, kernel_size=3, stride=1),
                nn.ReLU(inplace=True)
            )
            # Calculate the output size after convolutional layers
            self.conv_output_size = self._get_conv_output_size()

            # Setting up the fully connected layers
            self.fc1 = nn.Sequential(
                nn.Linear(self.conv_output_size, 1024),  # Adjust
input features to match conv_output_size
                nn.ReLU(),
                nn.Linear(1024, 256),
                nn.ReLU(),
                nn.Linear(256, 2)
            )

        def _get_conv_output_size(self):
            # Compute the output size after convolutional layers
            with torch.no_grad():
                dummy_input = torch.zeros(1, 1, 250, 250)  # Assuming
```

```
input size is 250x250
                conv_output = self.cnn1(dummy_input)
                conv_output_size =
conv_output.view(conv_output.size(0), -1).size(1)
            return conv_output_size

        def forward_once(self, x):
            output = self.cnn1(x)
            output = output.view(output.size(0), -1)  # Flatten the
output tensor
            output = self.fc1(output)
            return output

        def forward(self, input1, input2):
            output1 = self.forward_once(input1)
            output2 = self.forward_once(input2)
            return output1, output2

    def __init__(self):
        super().__init__()
        self.siamese_network = self.arc()

    def forward(self, input1, input2):
        return self.siamese_network(input1, input2)


def network():
    net = Siamese()
    losfunc = ContrastiveLoss()
    optim = torch.optim.Adam(net.parameters(), lr=0.001)

    return net, losfunc, optim



net,losfunc,optim=network()

net
```

```
Siamese(
  (siamese_network): arc(
    (cnn1): Sequential(
      (0): Conv2d(1, 96, kernel_size=(3, 3), stride=(4, 4))
      (1): ReLU()
      (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
      (3): Conv2d(96, 256, kernel_size=(5, 5), stride=(1, 1))
      (4): ReLU()
      (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
```

```
      (6): Conv2d(256, 384, kernel_size=(3, 3), stride=(1, 1))
      (7): ReLU(inplace=True)
    )
    (fc1): Sequential(
      (0): Linear(in_features=46464, out_features=1024, bias=True)
      (1): ReLU()
      (2): Linear(in_features=1024, out_features=256, bias=True)
      (3): ReLU()
      (4): Linear(in_features=256, out_features=2, bias=True)
    )
  )
)
```

9) Creating an instance of fake data for passing through the network to check if everything is working right or not

```
x = torch.randn(1, 1, 250, 250)
y= torch.randn(1, 1, 250, 250)
# Example input tensor
output=net(x,y)
print(output)
```

```
(tensor([[-3.0215,  7.0274]], grad_fn=<AddmmBackward0>), tensor([[-
3.0176,  7.0446]], grad_fn=<AddmmBackward0>))
```

10)Creating a training function compatible to the multiple output architecture

```
#passing it through the values
def train(net, losfunc, optim, train_0_loader, train_2_loader,
test_0_loader, test_2_loader, epochs=100):
    loss_train = []
    loss_test = []

    for epoch in range(epochs):
        for (x_1, y_1), (x_2, y_2) in zip(train_0_loader,
train_2_loader):
                net.train()
                yhat_1, yhat_2 = net(x_1, x_2)
                loss = losfunc(yhat_1, yhat_2, y_1)


                # Backward pass
                optim.zero_grad()
                loss.backward()
                optim.step()
```

```
                loss_train.append(loss.item())

        for (x_1, y_1), (x_2, y_2) in zip(train_0_loader,
train_2_loader):
                net.eval()
                # Forward pass
                yhat_1, yhat_2 = net(x_1, x_2)
                loss = losfunc(yhat_1, yhat_2, y_1)

                loss_test.append(loss.item())

    return loss_train, loss_test,net

 loss_train, loss_test,net=train(net, losfunc, optim, train_0_loader,
train_2_loader, test_0_loader, test_2_loader, epochs=100)

loss_train
```

```
[5.527718016651306e-08,
 5.8577686701255516e-08,
 1.1540588218394987e-07,
 3.4740015308898364e-08,
 4.565072586615315e-08,
 6.930076779099181e-08,
 2.8790307027293238e-08,
 1.0859655930062218e-07,
 4.779775863994473e-08,
 7.755588171676209e-08,
 4.309327650275918e-08,
 1.9930853412120086e-08,
 2.663107245837182e-08,
 2.8717890288021408e-08,
 4.472492776130821e-08,
 2.833900403231837e-08,
 4.237619677383009e-08,
 3.8595548090825105e-08,
 6.910395455861362e-08,
 3.021778738343528e-08,
 5.192356056227254e-08,
 7.841487104087719e-08,
 4.6666002617712365e-08,
 7.973847715447846e-08,
 5.06299464575477e-08,
 7.127390944106082e-08,
 8.760829928178282e-08,
 7.38949736955874e-08,
 1.3433653123229305e-07,
 6.940953767298197e-08,
 1.0363532254586971e-07,
 6.678499886447753e-08,
```

```
8.333668688464968e-08,
1.031192482514598e-07,
3.0887083113384506e-08,
7.036577898134055e-08,
1.5182713752892596e-07,
2.607956446354365e-07,
1.0007743611595288e-07,
8.813145768726827e-08,
2.4786552899058734e-07,
1.5405485953579046e-07,
3.3616558425819676e-07,
1.9873547785209666e-07,
7.661589052077034e-08,
2.3095752510471357e-07,
9.868477945929044e-08,
1.1179623271573291e-07,
6.797387896995133e-08,
6.944832620092711e-08,
5.566906935428051e-08,
8.909143645041695e-08,
3.3562955081833934e-08,
4.785095342185741e-08,
5.395123281459746e-08,
2.5179557994192692e-08,
8.95373375442432e-08,
4.679483467384671e-08,
7.293136405905898e-08,
6.227345039633292e-08,
4.6882540516435256e-08,
6.258510865109201e-08,
9.192182659489845e-08,
5.4047745834395755e-08,
6.346743219864948e-08,
5.249540180329859e-08,
3.713179452802251e-08,
8.704558496219761e-08,
9.956914226449953e-08,
4.1478241286085904e-08,
3.4812135396578014e-08,
4.609038128933207e-08,
3.308613472086108e-08,
8.288936470535191e-08,
5.3279563871910796e-08,
1.0985699816501437e-07,
7.789781619749192e-08,
1.5248521378907753e-07,
6.049507561556311e-08,
4.9804093293914775e-08,
8.160428421888355e-08,
```

```
7.740990781712753e-08,
4.265312014695155e-08,
1.8896616893471219e-07,
8.841390553016026e-08,
1.5540916820100392e-07,
1.0879840317556955e-07,
7.339326657529455e-08,
4.714341059752769e-07,
5.271263603390253e-08,
3.1122047516873863e-07,
2.3844219754209917e-07,
3.994386759131885e-07,
4.2587041093611333e-07,
1.5120252783162869e-07,
5.052248184256314e-07,
8.758820513321552e-08,
1.8878591845350456e-07,
2.1137432781870302e-07,
1.44396821610826e-07,
2.289128531174356e-07,
2.158232206284083e-07,
9.60567874130902e-08,
1.187769456123533e-07,
1.0939577776980514e-07,
6.389345230672916e-08,
1.1472421590497106e-07,
2.981215985187191e-08,
5.353174614697309e-08,
3.22638271654796e-08,
5.592596963310825e-08,
6.858386569774666e-08,
4.159684863225266e-08,
3.6059301322666215e-08,
3.3765783058470333e-08,
2.0061296623907765e-08,
3.631220479860531e-08,
6.557675646945427e-08,
2.7217355480502192e-08,
4.253669771969726e-08,
3.640792201053955e-08,
3.087039956994886e-08,
3.4140779092695084e-08,
3.738000131647823e-08,
3.712213469952985e-08,
5.5706692592139007e-08,
3.6368305700307246e-08,
3.0132472517152564e-08,
3.367225431816223e-08,
2.9113859767448957e-08,
```

```
2.359948148011881e-08,
5.473334141470332e-08,
3.214745802893049e-08,
5.569077998757166e-08,
3.840191098447576e-08,
2.2174448943701464e-08,
2.060332349174132e-08,
2.39746640318117e-08,
3.8677409719412026e-08,
3.050718433428301e-08,
4.596296321324189e-08,
2.6908184125318257e-08,
3.939966575217113e-08,
3.714683316502487e-08,
3.001268211733077e-08,
4.9371099208883606e-08,
3.8634610177723516e-08,
2.6274561193417867e-08,
2.8093181114741128e-08,
3.34901955056921e-08,
3.78203850459613e-08,
7.12508878564222e-08,
6.662239115939883e-08,
1.1136615540863204e-07,
1.0582510867607198e-07,
7.838014681738059e-08,
1.455335336686403e-07,
5.4415757233527984e-08,
1.6005344605218852e-07,
9.904858444542697e-08,
2.9025366643509187e-07,
8.654987482259457e-08,
1.1176756942177235e-07,
8.947994700747586e-08,
4.541849207839732e-08,
8.789640304485147e-08,
4.587982971315796e-08,
4.223634064715043e-08,
4.4239580887506236e-08,
3.7894121618364807e-08,
1.0753979751143561e-07,
6.006572306205271e-08,
1.7960631737423682e-07,
1.399309752514455e-07,
1.5509694151205622e-07,
1.8092777054334874e-07,
8.58106545820192e-08,
1.5337995762365608e-07,
1.7699301224638475e-07,
```

```
7.848137073551698e-08,
1.7606144808723911e-07,
1.0891280055602692e-07,
2.453716376749071e-07,
1.3880618610073725e-07,
7.859738815341188e-08,
6.254833095908907e-08,
1.0561337404624282e-07,
1.934303242023816e-07,
9.237765397074327e-08,
5.029622229812958e-08,
1.752620022443807e-07,
6.793300144636305e-08,
6.495140070228445e-08,
4.413021414961804e-08,
7.554385206276493e-08,
1.1477476391519303e-07,
1.9454147093256324e-07,
1.2817757522043394e-07,
2.0851925341958122e-07,
8.50625880843836e-08,
4.703333615907468e-07,
9.058248906512745e-08,
1.831799494311781e-07,
5.350889864530473e-07,
1.5760393523578387e-07,
1.0900180313910823e-06,
2.477144391832553e-07,
8.232390769080666e-07,
3.9591827771801036e-07,
3.482078057004401e-07,
1.1719143913069274e-06,
3.4859365882766724e-07,
1.1049615977754002e-06,
1.8615885437611723e-06,
1.880046283986303e-06,
8.923021255213825e-07,
8.070101671364682e-07,
5.66603887364181e-07,
2.958666982522118e-07,
3.9784029581824143e-07,
2.512424828182702e-07,
1.825495132834476e-07,
2.8891724923596485e-07,
3.324977342344937e-07,
2.3757658595968678e-07,
2.075390881373096e-07,
3.133599193461123e-07,
3.785305295878061e-07,
```

```
1.8038673488263157e-07,
2.448703071422642e-07,
9.222832630939592e-08,
4.62145237634104e-07,
2.9040322147011466e-07,
3.008188969033654e-07,
2.2048529046969634e-07,
2.7641866040539753e-07,
1.495109245297499e-07,
1.9519100646903098e-07,
2.518803796647262e-07,
1.8972163218222704e-07,
1.8967864434671355e-07,
3.257921150634502e-07,
1.9300044584724674e-07,
9.708855230883273e-08,
2.2338784333442163e-07,
1.8871843110446207e-07,
3.9325502143583435e-07,
1.4733630848695611e-07,
2.2761972218177107e-07,
4.346402420196682e-07,
2.1896231316986814e-07,
3.7277644082678307e-07,
8.157183373214139e-08,
2.350978007825688e-07,
2.8082084213565395e-07,
2.0254813648534764e-07,
2.2500647389733786e-07,
4.3288815732012154e-07,
1.3942201348982053e-07,
2.983367437536799e-07,
8.500902026753465e-07,
1.4816538396189571e-07,
5.084832537249895e-07,
3.22419765552695e-07,
3.087064612827817e-07,
2.6574238631837943e-07,
4.1287060525974084e-07,
1.6536991154225689e-07,
1.4550532512203063e-07,
2.55736608778534e-07,
5.920182744034719e-08,
9.853260962700006e-08,
1.4924034985597245e-07,
1.3901340878419433e-07,
7.21910211609611e-08,
1.8340652729875728e-07,
6.926879336788261e-08,
```

```
1.1093583651700101e-07,
8.503315740426842e-08,
7.640385746299216e-08,
4.7947171566420366e-08,
6.664600249450814e-08,
1.5266870434516022e-07,
7.158804038454036e-08,
6.460914647732352e-08,
8.533655204701063e-08,
1.7819785114170372e-07,
5.009709980186017e-08,
1.7374605931763654e-07,
1.2673307026034308e-07,
9.443843396184093e-08,
1.3994564085351158e-07,
6.079903158706657e-08,
6.094229121345052e-08,
9.286065250080355e-08,
6.801793261956846e-08,
4.7316305540334724e-08,
5.267710889711452e-08,
7.331692586376448e-08,
5.0456375078056226e-08,
7.254435985259988e-08,
6.344765068888591e-08,
7.766342946524674e-08,
5.742310094092318e-08,
7.028709347878248e-08,
4.8645457439988604e-08,
6.430261123568926e-08,
3.80227938023836e-08,
3.89216019414107e-08,
5.7287429910957144e-08,
4.381573504019798e-08,
4.945744791484685e-08,
5.996678709152548e-08,
2.8461013101832577e-08,
2.5300547434881082e-08,
3.332871756356326e-08,
1.9343936230598047e-08,
1.8549753733054786e-08,
4.5235619694494744e-08,
1.295235563247843e-08,
3.5704321277307827e-08,
3.390340808095971e-08,
1.5522674701173855e-08,
4.6831189592921874e-08,
2.19830109671193e-08,
2.097456608396442e-08,
```

```
2.452657277274284e-08,
3.512375812420032e-08,
2.4197829517902392e-08,
1.250378289796572e-08,
1.5707492195815576e-08,
2.534106791074464e-08,
1.727251408567554e-08,
1.07709974273007e-08,
3.360662148566007e-08,
1.2388646553063154e-08,
2.061932491415064e-08,
1.2074636401848693e-08,
1.5398306629776926e-08,
1.3164318524161445e-08,
1.2381946135064936e-08,
1.925045189921093e-08,
1.0105316583519652e-08,
1.6995919338569365e-08,
1.494569978888194e-08,
1.4860971120356226e-08,
1.5168522438102627e-08,
1.782122360793892e-08,
1.2556816919584435e-08,
2.032440882260289e-08,
2.1677568184941265e-08,
1.7054617273970507e-08,
1.5740136305453234e-08,
1.4776077250644448e-08,
1.2162628237888384e-08,
1.5000658493136143e-08,
2.0136379674795535e-08,
8.432373022060347e-09,
1.4847786111715777e-08,
1.9262230921412993e-08,
1.597232568428808e-08,
1.1512562458904085e-08,
1.2915338132302168e-08,
1.9422673247504463e-08,
1.0623946167243048e-08,
9.51230649803847e-09,
1.6938015434675435e-08,
1.067772537055589e-08,
1.1756746687296982e-08,
7.148442726645499e-09,
6.162643284568503e-09,
9.267735912033004e-09,
8.554405184213465e-09,
1.0388230720081992e-08,
1.1389141185702556e-08,
```

```
7.920569977670766e-09,
1.1368296526370614e-08,
8.227882375422269e-09,
7.577415139792265e-09,
9.9855190782705e-09,
1.031641172488662e-08,
8.154320774167445e-09,
1.1052957660240281e-08,
8.401283224657163e-09,
1.0855729648540091e-08,
1.0929858795805103e-08,
7.777823718413401e-09,
1.2082337796925913e-08,
6.412228525931596e-09,
7.3607884232274046e-09,
1.1240835817716288e-08,
1.4136644743700799e-08,
7.303956550686053e-09,
8.101952886363506e-09,
1.0258624172365671e-08,
7.1300325643619544e-09,
7.658329970183786e-09,
8.263932649299477e-09,
1.081829648086341e-08,
8.426664699356934e-09,
1.1344479133867935e-08,
9.808217349416282e-09,
5.824353888073119e-09,
6.340534763893402e-09,
9.21227183425799e-09,
8.061903145062388e-09,
8.44672687350112e-09,
8.318476574231681e-09,
5.612694753409642e-09,
5.158941718264032e-09,
1.0307175557500159e-08,
9.422094215949528e-09,
4.836802958152475e-09,
1.337657717925822e-08,
1.1872348437691471e-08,
7.166618409826242e-09,
1.9574700971247694e-08,
8.268205675676654e-09,
1.8285239988813373e-08,
1.8807710944201972e-08,
7.876796992434265e-09,
9.096866371294254e-09,
1.2067659760361948e-08,
7.719731520694495e-09,
```

```
7.796483458832881e-09,
1.4320972852033265e-08,
7.249443711998538e-09,
7.927389411577224e-09,
8.985853838794355e-09,
1.4505162404532257e-08,
8.65207372413579e-09,
1.5990373469776387e-08,
1.118189008053605e-08,
8.41956815378353e-09,
9.597036942921022e-09,
1.1096276786304315e-08,
8.754752478523642e-09,
1.1811390976390612e-08,
7.374743926646943e-09,
7.680630353945617e-09,
1.3758352679360542e-08,
8.739543311264697e-09,
8.704239995438456e-09,
1.4655994640122572e-08,
6.176725797502058e-09,
9.057831817926854e-09,
1.5144449250215075e-08,
1.0396082217312141e-08,
8.622738967289933e-09,
7.312948024917887e-09,
1.1084686946105649e-08,
9.442192805408922e-09,
9.333071204764565e-09,
9.516116783458983e-09,
1.1659579080003368e-08,
7.345199559694038e-09,
6.471239100136472e-09,
8.544304819224635e-09,
1.0900769176203084e-08,
6.240081340536108e-09,
6.3882774625767524e-09,
8.749680979747154e-09,
9.32719146362615e-09,
6.506092109503925e-09,
5.545690573427464e-09,
9.405933809603084e-09,
6.5021983353119595e-09,
7.649628486205984e-09,
6.931174301172405e-09,
9.568368319889942e-09,
4.907926065556012e-09,
7.112209488013832e-09,
9.004540224566426e-09,
```

```
5.497378552377086e-09,
5.4370943303183594e-09,
5.904164268599743e-09,
6.001520613807543e-09,
6.697483900808265e-09,
4.685833498996317e-09,
8.149755537090186e-09,
7.126768064580347e-09,
5.072231079594758e-09,
7.531007817362934e-09,
5.6137050563620505e-09,
4.139864007157712e-09,
7.033789994892459e-09,
6.006167119210204e-09,
6.152419906868545e-09,
8.109558358171398e-09,
7.006071722770457e-09,
5.755975696075666e-09,
6.843083877328127e-09,
4.405826814490865e-09,
7.124945966552332e-09,
6.16380857465515e-09,
8.10958145081031e-09,
6.705707544796269e-09,
8.518299843274235e-09,
7.2721704214018246e-09,
8.62115001609709e-09,
5.616672460462269e-09,
6.565975763095366e-09,
8.371075388424742e-09,
6.68973942907769e-09,
5.899184696289694e-09,
1.1567094837516834e-08,
5.96983840139842e-09,
3.773129364503802e-09,
8.04682187549588e-09,
6.684110154253631e-09,
4.637030315279844e-09,
5.433799188381272e-09,
5.131375768741009e-09,
5.630993893390723e-09,
5.9265898855187515e-09,
6.2675620249308395e-09,
4.722947366531116e-09,
7.104311361416649e-09,
5.095082133976803e-09,
5.754359211351812e-09,
7.2480079715830925e-09,
5.6011408844369726e-09,
5.851021445124616e-09,
```

```
6.096676941069745e-09,
7.026251136466044e-09,
3.07971204094315e-09,
4.671420139601423e-09,
5.682877723955926e-09,
4.443620138516735e-09,
5.260260671491324e-09,
7.108991173510049e-09,
5.554355642090059e-09,
6.501974958439405e-09,
1.0828888008518334e-08,
4.924342267287329e-09,
5.678111314466605e-09,
7.847954286432923e-09,
4.2574797021188715e-09,
5.398827607194789e-09,
6.044277078842697e-09,
6.659337081771355e-09,
3.3839582247452427e-09,
3.458776376419337e-09,
3.904744527716275e-09,
5.60893820278352e-09,
4.605663406209715e-09,
4.14715550789424e-09,
4.0516563437620334e-09,
6.298218835354419e-09,
6.792405304878457e-09,
7.231538035057383e-09,
3.1821636437001644e-09,
3.5132428077844224e-09,
5.871818586911104e-09,
3.8079712716410086e-09,
3.3042260039195526e-09,
4.703227585167724e-09,
4.838988321154147e-09,
3.4434679552219905e-09,
4.261117236836753e-09,
7.007152635907232e-09,
3.7403200536800796e-09,
4.1321199795163466e-09,
5.097218647165391e-09,
5.361957988725408e-09,
5.338097519569374e-09,
3.938069426112634e-09,
2.643584906536489e-09,
3.283965543943168e-09,
5.43030909128106e-09,
4.1689429686186 94e-09,
4.146972543139782e-09,
```

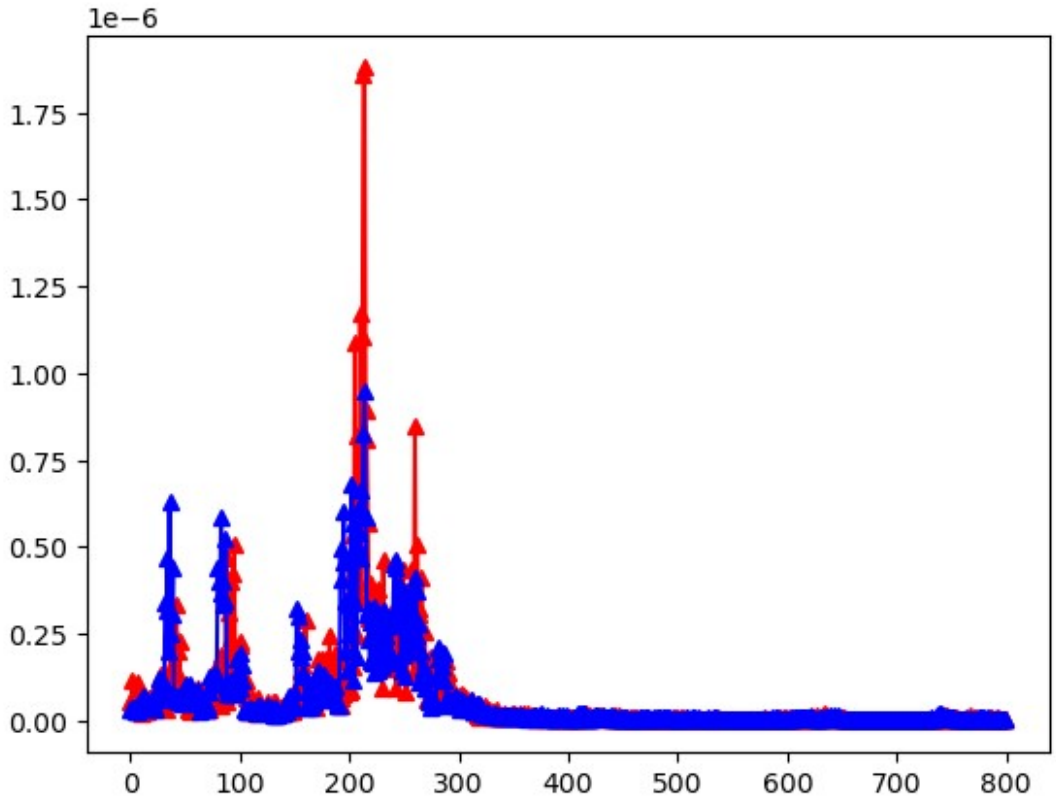
```
4.453987401120685e-09,
4.964617605907051e-09,
3.799618841782149e-09,
3.3633169582714117e-09,
6.330911350715951e-09,
2.262678711062449e-09,
4.521907737142783e-09,
8.079374502756309e-09,
3.2595297572157733e-09,
3.4349842970016198e-09,
6.491439830114132e-09,
6.4270477828642925e-09,
5.6435132123056064e-09,
5.1368029829745865e-09,
6.112888861764532e-09,
4.018680055395407e-09,
4.7758210719450744e-09,
5.862809349110876e-09,
5.244753076283359e-09,
6.329234469859557e-09,
9.148044988194215e-09,
6.712923106277913e-09,
8.186220590289395e-09,
6.066588564834774e-09,
1.0341699940852322e-08,
6.7009575666077126e-09,
9.501262887567918e-09,
5.850588014055802e-09,
6.122839124600432e-09,
5.930367308337736e-09,
8.152555075469081e-09,
5.234273459109318e-09,
6.181007705663433e-09,
1.245503256086522e-08,
4.0362988507069986e-09,
5.282348780610846e-09,
7.36524397026983e-09,
7.531100187918582e-09,
5.561148430643925e-09,
7.767682497217265e-09,
3.9211638380720615e-09,
5.400334401883811e-09,
1.1034551938848836e-08,
6.716731615341587e-09,
1.0268561112525276e-08,
5.330960117788663e-09,
5.753147735987341e-09,
1.1271756861219728e-08,
6.139873942601071e-09,
```

```
5.931802160574762e-09,
1.374209723792319e-08,
8.594924771898604e-09,
5.6706195294964346e-09,
8.178210997300539e-09,
7.664797685436042e-09,
1.5607419356911123e-08,
1.2218072775738165e-08,
6.425787013597528e-09,
9.447538751317097e-09,
1.2974732399584354e-08,
6.282363962384352e-09,
1.3507936991175029e-08,
2.1536992633741647e-08,
8.040756505067748e-09,
1.6850703943305234e-08,
9.899820518910474e-09,
1.0707709385826547e-08,
9.022826041871213e-09,
7.3354828877825184e-09,
8.036438181591166e-09,
1.1144079437030996e-08,
8.886289926124391e-09,
8.912738991284641e-09,
7.2202261947040824e-09,
6.840724875445403e-09,
5.4565028051456466e-09,
1.2630807511015973e-08,
9.437389536515184e-09,
7.873278029535413e-09,
7.828012904553816e-09,
4.80811790382063e-09,
6.5115663971937465e-09,
5.81339332228481e-09,
5.8085403153995685e-09,
5.222489996015156e-09,
3.8157939030725174e-09,
7.0598042967162655e-09,
4.848108137167628e-09,
5.384125145724283e-09,
5.410995651544681e-09,
5.1950088675312145e-09,
4.980424961331664e-09,
4.4081036598697665e-09,
3.883529942072528e-09,
4.3777212965778745e-09,
5.908233458029599e-09,
5.37344213569213e-09,
5.47988987520398e-09,
```

7.585144068400496e-09,
4.73549510715543e-09,
5.241261646915518e-09,
4.8280734965544525e-09,
5.869307262429402e-09,
3.4380769342590156e-09,
4.722072954876921e-09,
6.985991785057877e-09,
3.956844185637465e-09,
5.584844586792315e-09,
4.053138269455303e-09,
4.360929395375024e-09,
4.083941629318133e-09,
4.0720422589402006e-09,
3.901579059828464e-09,
4.076557313936746e-09,
4.377313178594022e-09,
2.8019206954610354e-09,
5.3608952832462364e-09,
4.307379342094464e-09,
6.021938947498029e-09,
5.810425918184592e-09,
4.559288502292702e-09,
6.916187622607595e-09,
6.253508377795924e-09,
3.4520124536641106e-09,
6.452907097553862e-09,
6.001162677904404e-09,
4.844086465283226e-09,
4.913008222473536e-09,
5.064201946680669e-09,
6.677516317665777e-09,
6.088115789282256e-09,
5.231372668390577e-09,
4.752270132968306e-09,
7.402536805756199e-09,
5.072005038186944e-09,
7.085724895716794e-09,
5.444459105774513e-09,
3.8470884256014415e-09,
5.041359329993611e-09,
4.999691327611799e-09,
3.4066578447067286e-09,
3.882540511312982e-09,
3.148693528132185e-09,
3.94832788686017e-09,
5.902117905520754e-09,
3.785025182168056e-09,
5.3096584906597855e-09,

```
6.5928125181358155e-09,
4.005230369585888e-09,
6.27375174033773e-09,
4.024286237580554e-09,
3.3225870943454083e-09,
2.755340622329072e-09,
2.8905984272853402e-09,
3.3124831766428997e-09,
3.5637930384524452e-09,
4.1532497441210126e-09,
3.3085798545329226e-09,
3.1107112441475238e-09,
5.080337928120571e-09,
3.4881730837099667e-09,
6.5711671659585136e-09,
4.868963898729817e-09,
4.448529988820837e-09,
6.693782417244165e-09,
3.2648164172144334e-09,
6.4192664517293e-09,
3.1536087075068053e-09,
3.5473983750478055e-09,
6.12915496134292e-09,
6.578665612266832e-09,
6.548140696338578e-09,
1.720673203919887e-08,
4.247928675482626e-09,
1.3769509088490395e-08,
5.268247615930477e-09,
5.3013238243693195e-09,
9.515617627187112e-09,
9.178998006120764e-09,
2.7746298592035146e-09,
8.775749016365353e-09,
8.780804527930286e-09,
4.1878469581035915e-09,
7.908725230265645e-09,
5.80542502959247e-09,
4.989676227751261e-09,
6.742482128174743e-09,
4.3579508890445595e-09,
4.2447969583747636e-09,
5.139070946569291e-09,
6.125474349971682e-09,
4.718085033772468e-09,
9.248640076009451e-09,
6.79900269417999e-09,
3.2000533334297643e-09,
1.505652669209212e-08,
```

```
 4.523315499938008e-09,
 6.4233067753605155e-09,
 8.917506733041591e-09,
 6.84434864339778e-09,
 7.90486165413995e-09,
 4.7263521985030366e-09,
 6.837229005185463e-09,
 8.172258425531709e-09,
 1.0055940968811683e-08,
 4.722528146317018e-09,
 5.057121832408029e-09,
 5.576758166370155e-09,
 4.298394973289987e-09,
 6.849528499941471e-09,
 4.665436037498694e-09,
 6.269563534999634e-09,
 6.2043947757217666e-09,
 1.000441329779278e-08,
 8.084067637526005e-09,
 6.204342373195004e-09,
 6.605390012737189e-09,
 5.518302259588381e-09,
 6.706067257056247e-09,
 9.523507316089308e-09,
 9.103005460531222e-09,
 5.4521223091796855e-09,
 4.435644740397038e-09,
 6.002533137206001e-09,
 5.320724305590829e-09,
 6.0373004373559525e-09,
 5.975379302469719e-09,
 7.700188042747413e-09]


plt.plot(loss_train,'r^-')
plt.plot(loss_test,'b^-')

[<matplotlib.lines.Line2D at 0x1f2e3f7e610>]
```

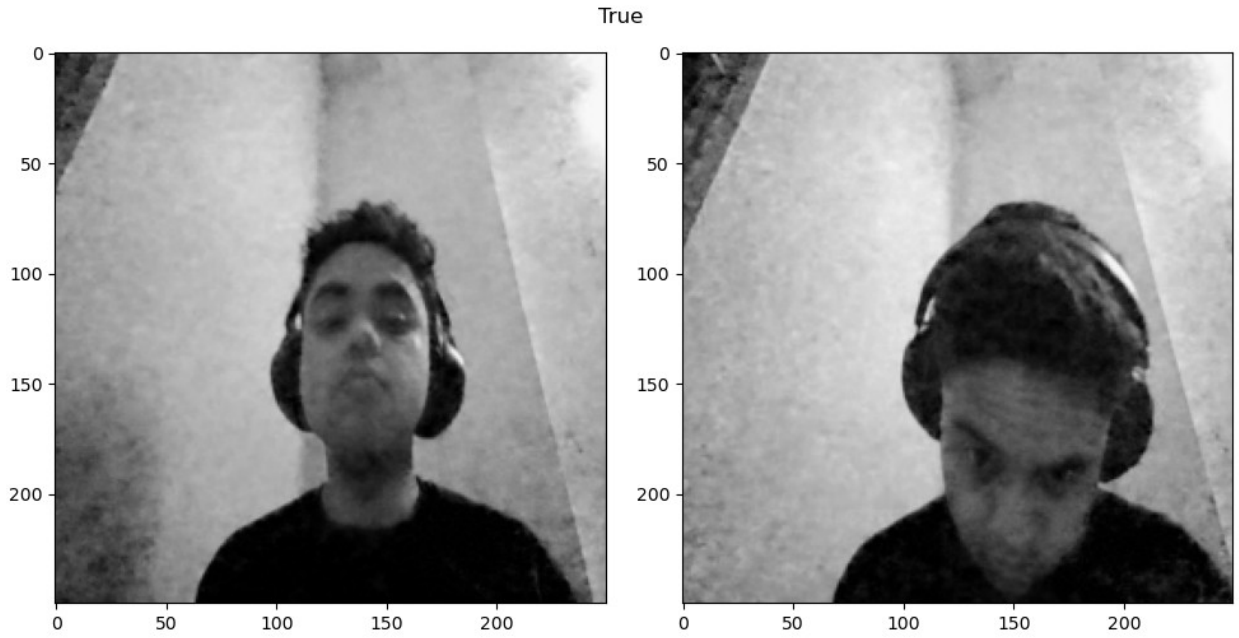


11) Using Euclidean distance to differentiate positive and negatives

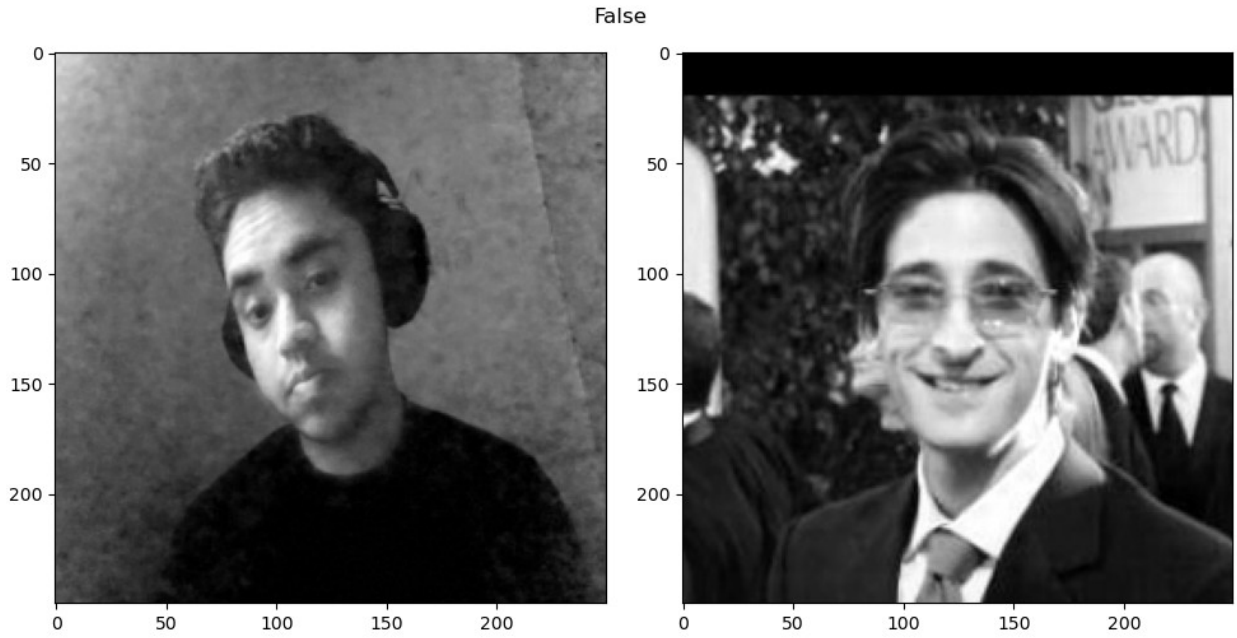
```
x,y=next(iter(train_0_loader))
x_1,y_1=next(iter(train_1_loader))
x_2,y_2=next(iter(train_2_loader))
output_1,output_2=net(x,x_1)
output_3,output_4=net(x,x_2)
similarity_1=F.pairwise_distance(output_3,output_4)
similarity=F.pairwise_distance(output_1,output_2)
print(similarity_1)
print(similarity)
```

```
tensor([1.6064e-04, 2.7940e-05, 3.1236e-05, 1.2562e-04, 1.6353e-06,
1.0673e-04,
        1.0659e-04, 8.0889e-05, 6.7923e-05, 6.4562e-05, 9.0838e-05,
8.0660e-05,
        4.7450e-05, 1.6541e-05, 1.5452e-05, 1.0662e-04, 7.7757e-05,
5.3569e-05,
        2.4170e-05, 1.1588e-05, 7.5811e-05, 3.0322e-05, 7.3756e-05,
5.0895e-05,
        1.2238e-04, 4.6117e-05, 6.2961e-05, 5.0414e-05, 5.1708e-05,
4.7672e-05,
        3.9189e-05, 1.0763e-04], grad_fn=<NormBackward1>)
tensor([0.0002, 0.0003, 0.0008, 0.0013, 0.0011, 0.0003, 0.0018,
0.0010, 0.0029,
```

```
        0.0011, 0.0010, 0.0008, 0.0008, 0.0020, 0.0012, 0.0002,
0.0004, 0.0011,
        0.0003, 0.0013, 0.0005, 0.0005, 0.0010, 0.0006, 0.0014,
0.0005, 0.0006,
        0.0001, 0.0011, 0.0002, 0.0013, 0.0004],
grad_fn=<NormBackward1>)

x_1,y_1=next(iter(train_0_loader))

x_3,y_3=next(iter(train_2_loader))


output_3,output_4=net(x_1,x_3)

x_plot_1 = np.transpose(x_1, (0, 2, 3, 1))
x_plot_3 = np.transpose(x_3, (0, 2, 3, 1))

x_rgb_1 = np.repeat(x_plot_1, 3 ,axis=3)

x_rgb_3 = np.repeat(x_plot_3, 3 ,axis=3)
fig,ax=plt.subplots(1,2,figsize=(10,5))

ax[0].imshow(x_rgb_1[0])
ax[1].imshow(x_rgb_3[0])

plt.suptitle(f'{(F.pairwise_distance(output_3[0],output_4[0])<1e-
04)}')

plt.tight_layout()
plt.show()
```




```
x_1,y_1=next(iter(train_0_loader))
x_2,y_2=next(iter(train_1_loader))
output_1,output_2=net(x_1,x_2)
x_plot_1 = np.transpose(x_1, (0, 2, 3, 1))
x_plot_2 = np.transpose(x_2, (0, 2, 3, 1))
x_rgb_1 = np.repeat(x_plot_1, 3 ,axis=3)
x_rgb_2 = np.repeat(x_plot_2, 3 ,axis=3)
fig,ax=plt.subplots(1,2,figsize=(10,5))

ax[0].imshow(x_rgb_1[0])
ax[1].imshow(x_rgb_2[0])

plt.suptitle(f'{(F.pairwise_distance(output_1[0],output_2[0])<1e-
04)}')

plt.tight_layout()
plt.show()
```



Hence, we have succesfully converted paper research work into real time implementation project