GMI_Attack

January 30, 2023

1 setup

[1]:
```
import os
import time
import numpy as np
from matplotlib import pyplot as plt
from google.colab import drive
import h5py
```

[2]:
```
import torch
import torch.nn as nn
from sklearn.model_selection import train_test_split
from torch.utils.data import Dataset, DataLoader

import torchvision.utils as tvls
```

[3]:
```
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

[4]:
```
# Balazs
%cd "/content/drive/MyDrive/University/2. UniStuttgart MSc./Studies/2022W/
 ↪Machine Perception and Learning/Project"
```

```
/content/drive/MyDrive/University/2. UniStuttgart MSc./Studies/2022W/Machine
Perception and Learning/Project
```

[5]:
```
# Mohamed / Ekaterina
# %cd "/content/drive/MyDrive/Project"
```

2 TODO List

- Dataset:
- preproc code
-

2.1 feature: img, target: pID

- Classifier:
- predicts pID
- splitting based on train/val/test (80-10-10)
  - VGG16 (mandatory)
  - Underfitting VGG16
  - Custom Classifier
  -

2.2 ResNet152

- GAN:
- code
- auxiliary knowledge creator (IP)
  - add powerful gaussian blur and masking (meta-function) to be used for central/chess masking
- target (gets G image and theroetical label => target loss)
  - attack with noise
    - VGG16
    - Custom Net
    - Underfit VGG16
    - ResNet152
- D (G image => D loss)
  - add code for D & G together with training utilities
  - training GAN from noise

3 Utils

[6]:
```
def create_dataset(rng):
    f = h5py.File('datasets/MPIIFaceGaze.h5', 'r')
    imgs = np.array([])
    targets = np.array([])

    # private dataset
    for i in rng:
        print('$ {:02}...'.format(i), end='')
        img = np.array(
            [x[()][...,::-1] for x in list(f['p{:02}'.format(i)]['image'].
↪values())]
        )
```

```
        t = np.array([i for x in range(len(img))])

        if not np.any(imgs):
            imgs = img
        else:
            imgs = np.append(imgs, img, axis=0)

        targets = np.append(targets, t, axis=0)
        print(len(imgs))

        plt.imshow(img[0])
        plt.title(t[0])
        plt.show()
    return imgs, targets
```

[7]:
```
from torchvision import transforms

# works both for resnet and vgg
transform = transforms.Compose([
    transforms.ToPILImage(),
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

transformGAN = transforms.Compose([
    transforms.ToPILImage(),
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])
```

[8]:
```
# Define a class for custom dataset
class DS(Dataset):
        def __init__(self, x, y, transforms=None):
                self.x = x
                self.y = torch.LongTensor(y)
                self.t = transforms

        def __getitem__(self, index):
                if self.t:
                        return self.t(self.x[index]), self.y[index]
                else:
                        return self.x[index], self.y[index]

        def __len__(self):
                return len(self.x)
```

4 Setup Dataset

[ ]:
```
%pwd
```

[ ]:
```
# %mkdir -p datasets
# %cd datasets
# !wget http://datasets.d2.mpi-inf.mpg.de/MPIIGaze/MPIIFaceGaze_normalized.zip
# !unzip MPIIFaceGaze_normalized.zip
# %mv MPIIFaceGaze_normalizad MPIIFaceGaze_normalized
# %cd ..
```

[ ]:
```
# !python src/preproc.py --dataset datasets/MPIIFaceGaze_normalized -o datasets/
```

---

5 Target Network(s) (VGG16s + custom, ResNet-152)

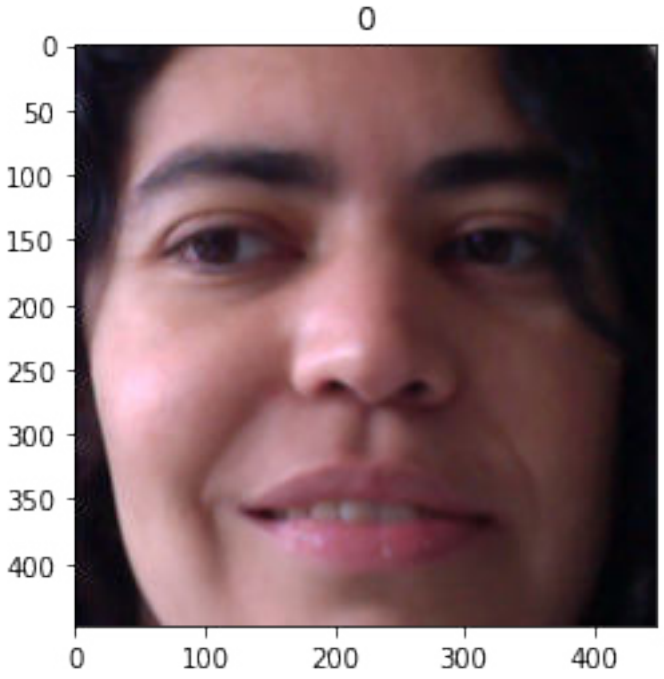
5.1 Data Preprocessing

[9]:
```
%pwd
```

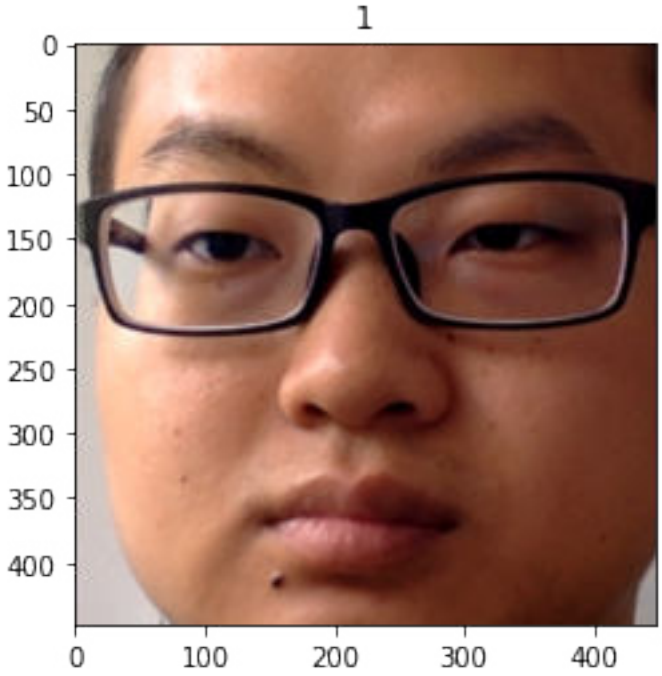
[9]:
```
'/content/drive/MyDrive/University/2. UniStuttgart MSc./Studies/2022W/Machine
Perception and Learning/Project'
```

[ ]:
```
target_x, target_y = create_dataset(range(10))
```

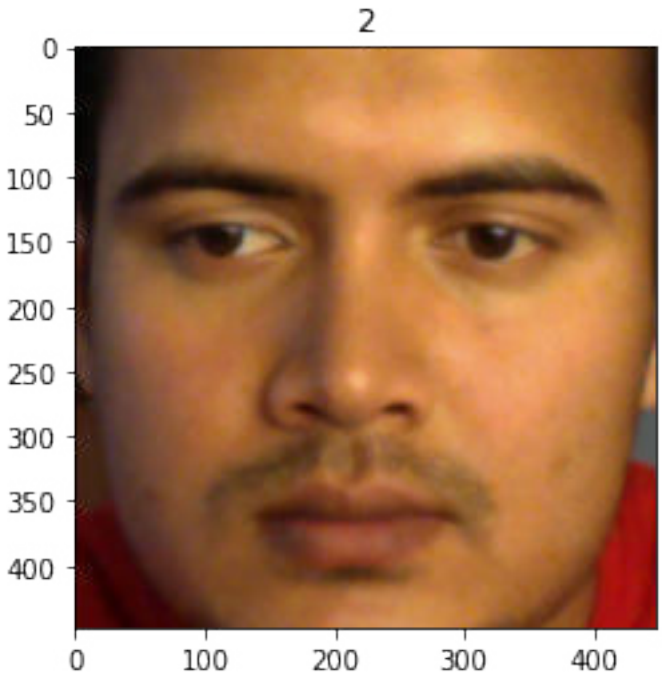
```
$ 00...3000
```



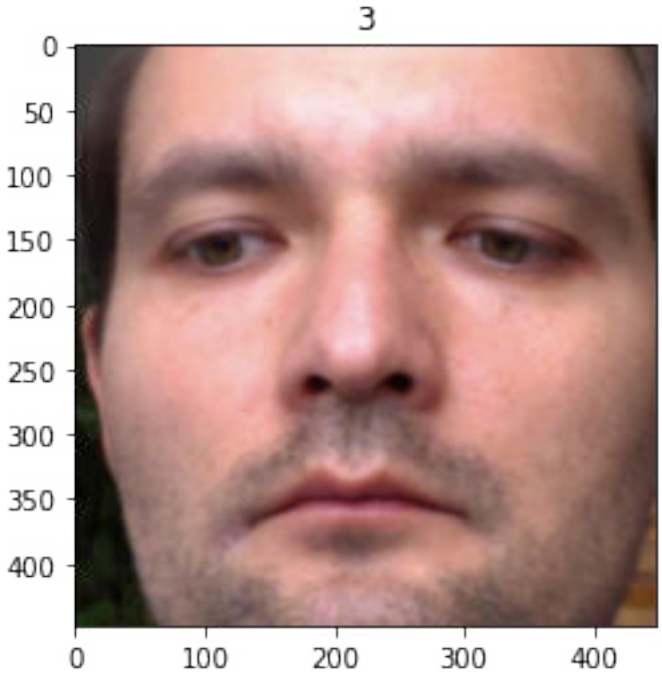
$ 01...6000



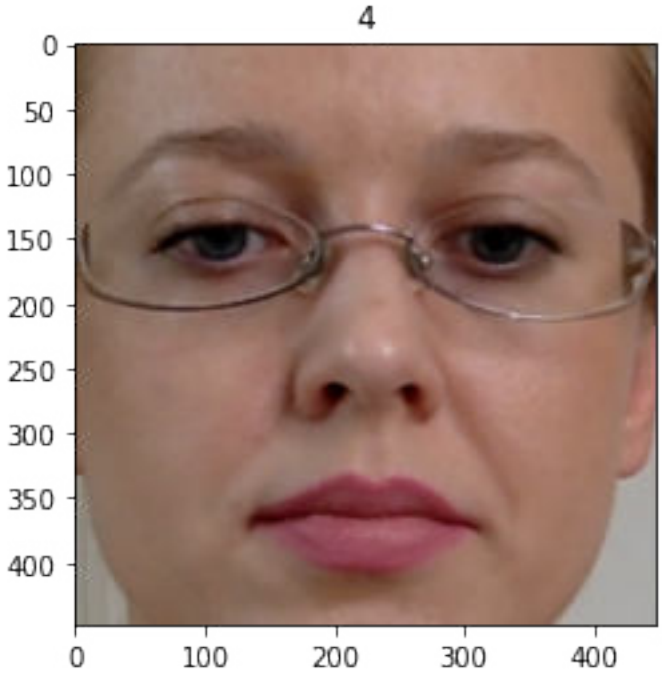
$ 02...9000


2
0
50
100
150
200
250
300
350
400
0
100
200
300
400

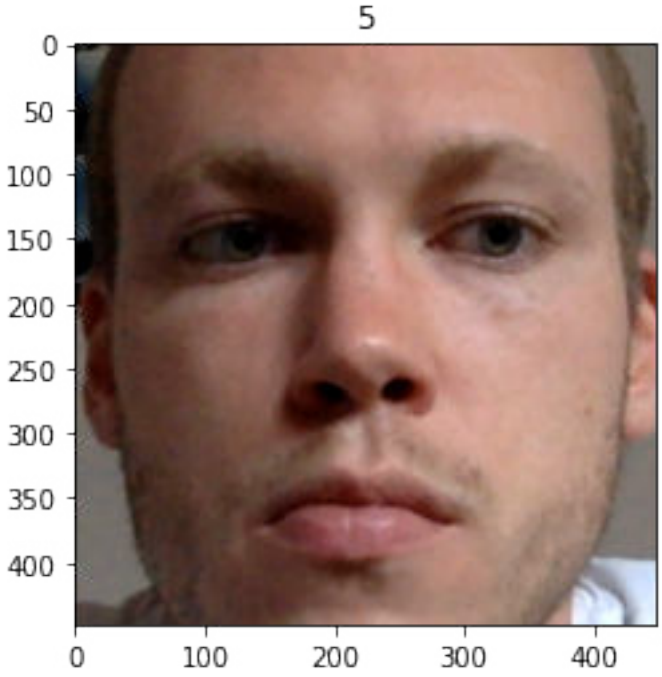
$ 03...12000


3
0
50
100
150
200
250
300
350
400
0
100
200
300
400

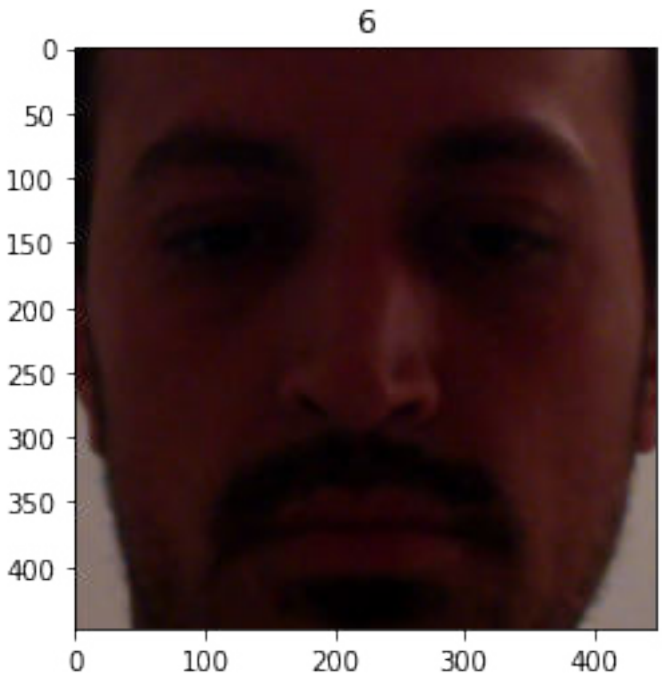
$ 04...15000



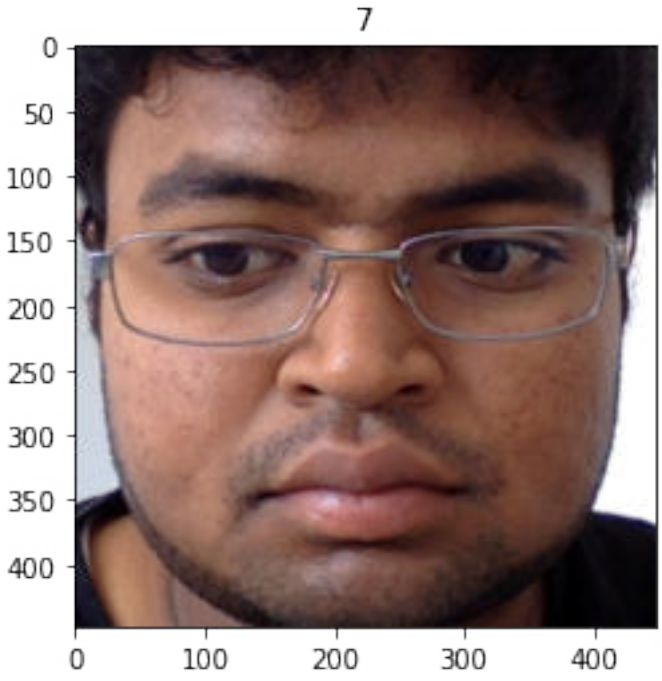
$ 05...18000



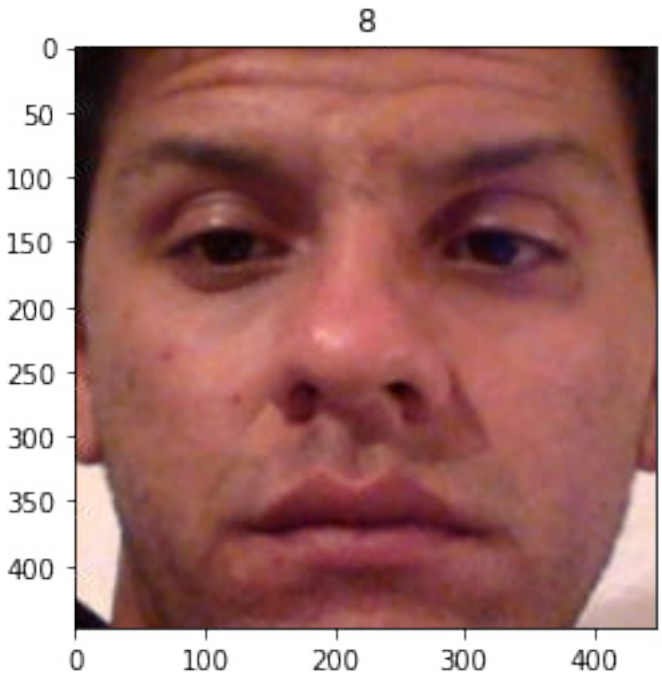
$ 06...21000



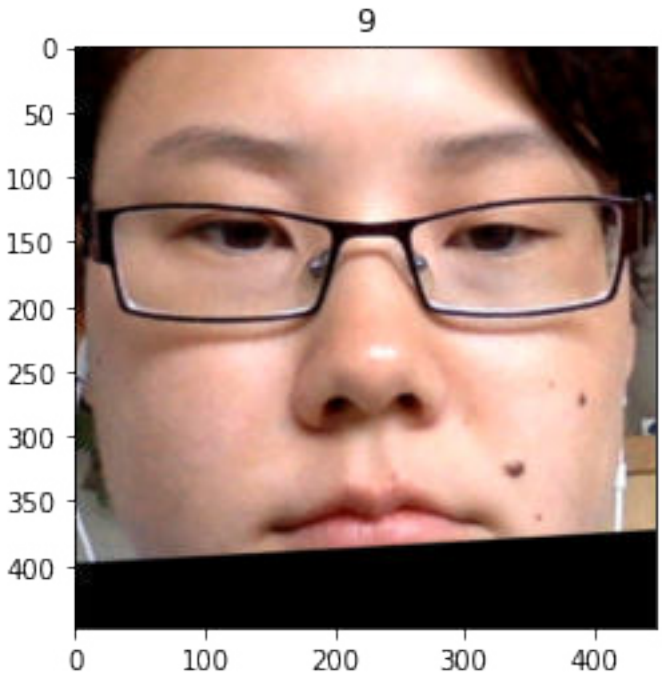
$ 07...24000



$ 08...27000

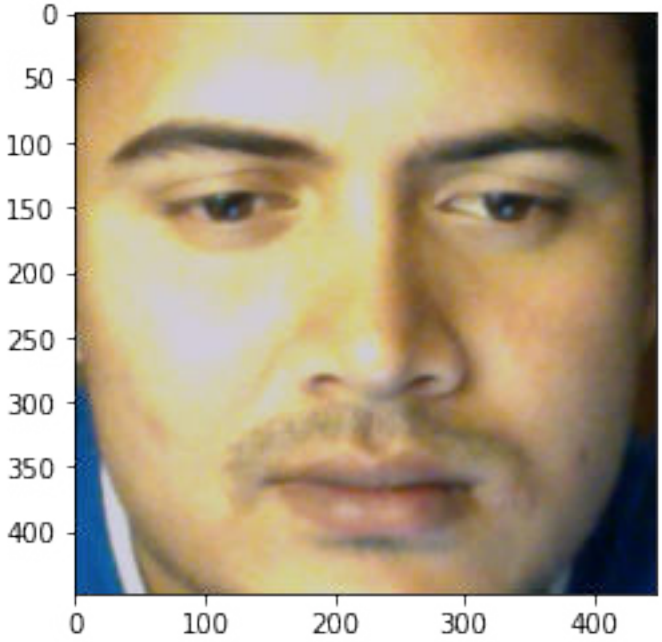


$ 09...30000



[ ]:
```
print(target_y[12345])
plt.imshow(target_x[12345])
plt.show()
```

```
2.0
```



[11]:
```
target_x.shape
```

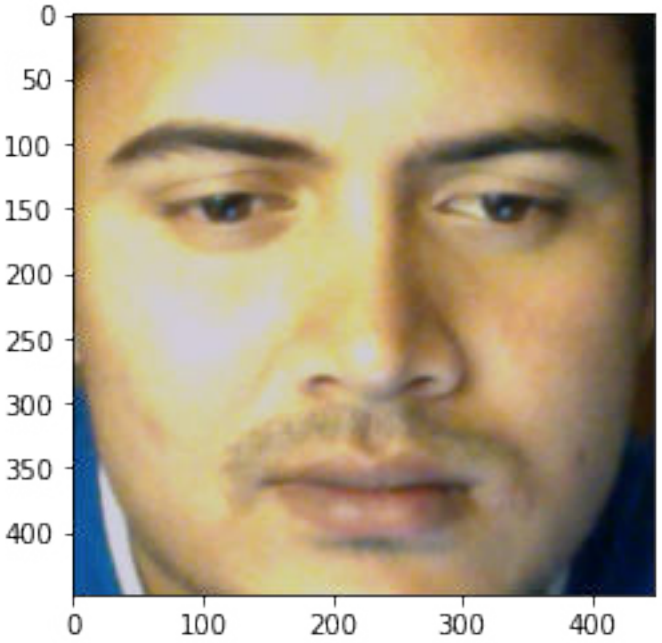
[11]:
```
(10000, 448, 448, 3)
```

[12]:
```
target_x, target_test_x, target_y, target_test_y = train_test_split(
    target_x, target_y, test_size=.1, random_state=42, shuffle=True
)
target_x, target_val_x, target_y, target_val_y = train_test_split(
    target_x, target_y, test_size=.11111, random_state=42, shuffle=True
)

print('Train:', target_x.shape)
print('Val:', target_val_x.shape)
print('Test:', target_test_x.shape)
```

```
Train: (8000, 448, 448, 3)
Val: (1000, 448, 448, 3)
Test: (1000, 448, 448, 3)
```

[ ]:
```
print(target_y[12345])
plt.imshow(target_x[12345])
plt.show()
```

```
2.0
```



5.2 ML

ML setup

[13]:
```
# Create train, validation, test Dataset
train = DS(target_x, target_y, transform)
val = DS(target_val_x, target_val_y, transform)
test = DS(target_test_x, target_test_y, transform)
```

[14]:
```
train_loader = DataLoader(train, batch_size=32, shuffle=True)
val_loader = DataLoader(val, batch_size=32, shuffle=True)
test_loader = DataLoader(test, batch_size=32, shuffle=True)
```

[15]:
```
%cd src/
```

```
/content/drive/MyDrive/University/2. UniStuttgart MSc./Studies/2022W/Machine
Perception and Learning/Project/src
```

[16]:
```
from target_network import VGG16, CustomTNet, ResNet152

import torch.nn as nn
import torch.optim as optim
```

[17]:
```
device = "cuda" if torch.cuda.is_available() else "cpu"
```

[18]:
```
device
```

[18]:
```
'cuda'
```

utility

[21]:
```
def train(net, output_file, underfit=False):
    train_loss = []
    val_loss = []

    net = net.to(device)

    # Early Stopping
    es_condition = 3
    es_it = 0
    es_loss = np.inf

    # ###
    print('!! Training start...')
    epochs = 3 if underfit else 20
    for epoch in range(epochs):
        net.train()
        tloss, tacc = [], []
        for iter, (x, y) in enumerate(train_loader):
            x = x.to(device)
```

```
            y = y.to(device)
            y_hat = net(x)
            loss = crit(y_hat, y)
            tloss.append(loss.item())

            opt.zero_grad()

            loss.backward()
            opt.step()

            argmax_Y = torch.max(y_hat.data, 1)[1].view(-1, 1)
            tacc.append((y.float().view(-1, 1) == argmax_Y.float()).sum().item()␣
↪/ len(y.float().view(-1, 1)) * 100)
        # ###
        tloss = np.mean(np.array(tloss))
        tacc = np.mean(np.array(tacc))
        train_loss.append(tloss)

        # #####
        net.eval()
        vloss, vacc = [], []
        for iter, (x, y) in enumerate(val_loader):
            x = x.to(device)
            y = y.to(device)
            y_hat = net(x)
            loss = crit(y_hat, y)
            vloss.append(loss.item())

            argmax_Y = torch.max(y_hat.data, 1)[1].view(-1, 1)
            vacc.append((y.float().view(-1, 1) == argmax_Y.float()).sum().item()␣
↪/ len(y.float().view(-1, 1)) * 100)
        # ###
        vloss = np.mean(np.array(vloss))
        vacc = np.mean(np.array(vacc))
        val_loss.append(vloss)

        if epoch % 2 == 0:
            print(f'Epoch {epoch}:')
            print(f' - loss: train: {tloss}, val: {vloss}')
            print(f' - ACC: train: {tacc}, val: {vacc}', end='\n')

        # ### Early Stopping ###
        if vloss < es_loss:
            print('* Early Stopping reset...')
            es_it = 0
            es_loss = vloss
        else:
```

```
            print('* Early Stopping increase...')
            es_it += 1
        if es_it == es_condition:
            print('*** Early Stopping!')
            break

    # save model
    torch.save(net.state_dict(), output_file)
    print('!! Training stop...')

    # ###
    import matplotlib.pyplot as plt
    x_axis = len(train_loss)
    plt.plot(np.arange(x_axis), train_loss, label='train losses')
    plt.plot(np.arange(x_axis), val_loss, label='val losses')
    plt.legend()
    plt.show()
```

[22]:
```
def test(net):
    net.eval()

    with torch.no_grad():
        taccs = []
        for iter, (x, y) in enumerate(test_loader):
            y_hat = net(x)

            # calculate accuracy
            argmax_Y = torch.max(y_hat, 1)[1].view(-1, 1)
            test_acc=((y.float().view(-1, 1)== argmax_Y.float()).sum().item()/
↪len(y.float().view(-1, 1)) * 100)
            taccs.append(test_acc)
        return sum(taccs) / len(taccs)
```

5.2.1 Target Network (VGG16)

train

[ ]:
```
# create network model
net = VGG16(n_classes=10)
# loss function
crit = nn.CrossEntropyLoss()
# optimizer
opt = optim.SGD(net.parameters(), lr=10e-3, momentum=0.9, weight_decay=10e-4)
```

[ ]:
```
train(
    net,
    "target-net.pt"
)
```

test

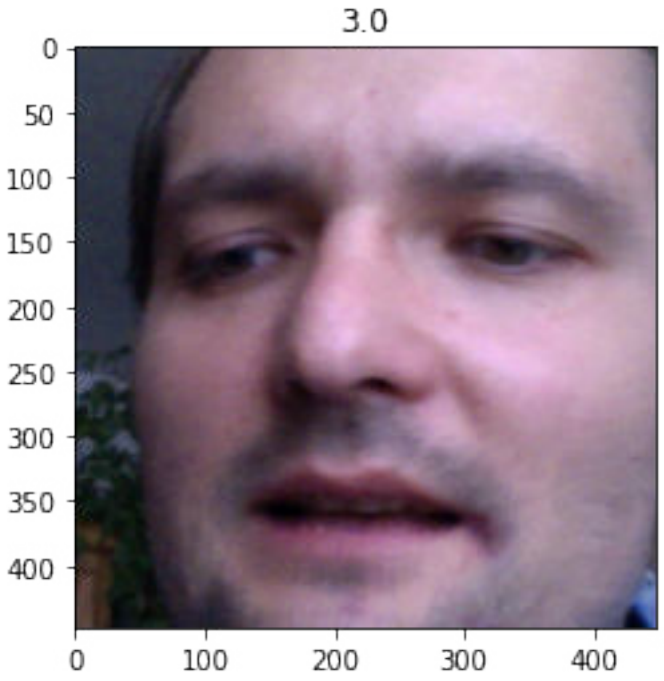
```
[ ]: # create network model
net = VGG16(n_classes=10)
net.load_state_dict(torch.load("target-net.pt"))
net = net.to('cpu')

print('Test accuracy:', test(net))
```

```
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be
removed in the future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are
deprecated since 0.13 and may be removed in the future. The current behavior is
equivalent to passing `weights=VGG16_BN_Weights.IMAGENET1K_V1`. You can also use
`weights=VGG16_BN_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)

Test accuracy: 99.96675531914893
```

test one sample

```
[ ]: random_idx = np.random.randint(3000)
x, y = target_x[random_idx], target_y[random_idx]
```

```
[ ]: plt.imshow(x)
plt.title(y)
plt.show()
```



[ ]:
```
# won't map to device since CPU is used for inference :-?
x_t = transform(x).unsqueeze(0)  # [batch, channels, H, W]
y_hat = torch.argmax(net(x_t)).detach().numpy()
print(y_hat)
```

```
3
```

5.2.2 Underfitting Target Network (VGG16)

train

[19]:
```
# create network model
underfit_net = VGG16(n_classes=10)
# loss function
crit = nn.CrossEntropyLoss()
# optimizer
opt = optim.SGD(underfit_net.parameters(), lr=10e-3, momentum=0.9,␣
↪weight_decay=10e-4)
```

```
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be
removed in the future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are
deprecated since 0.13 and may be removed in the future. The current behavior is
equivalent to passing `weights=VGG16_BN_Weights.IMAGENET1K_V1`. You can also use
`weights=VGG16_BN_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
```

[ ]:
```
train(
    underfit_net,
    "target-net_underfit.pt",
    underfit=True
)
```

test

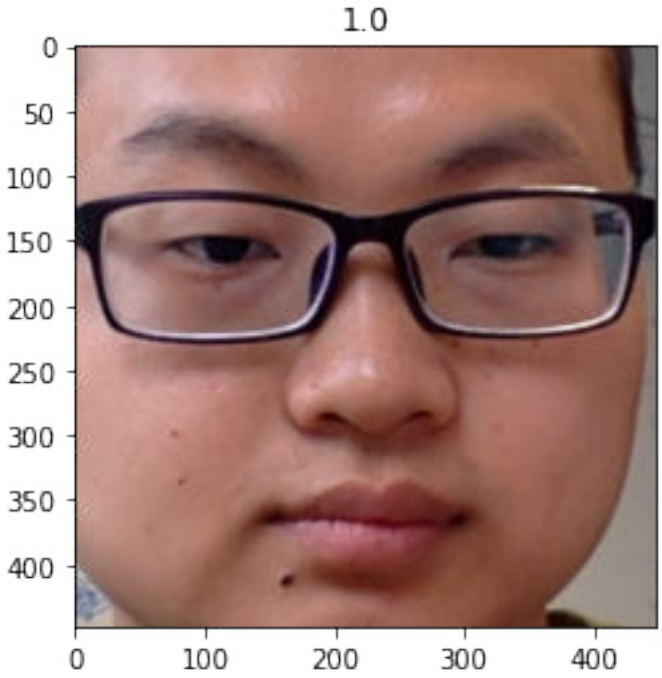
[25]:
```
# create network model
underfit_net = VGG16(n_classes=10)
underfit_net.load_state_dict(torch.load("target-net_underfit.pt"))
underfit_net = underfit_net.to('cpu')

print('Test accuracy:', test(underfit_net))
```

```
Test accuracy: 84.08203125
```

test one sample

[26]:
```
random_idx = np.random.randint(3000)
x, y = target_x[random_idx], target_y[random_idx]
```

[27]:
```
plt.imshow(x)
plt.title(y)
plt.show()
```



[28]:
```
# won't map to device since CPU is used for inference :-?
x_t = transform(x).unsqueeze(0)  # [batch, channels, H, W]
y_hat = torch.argmax(underfit_net(x_t)).detach().numpy()
print(y_hat)
```

9

5.2.3 Custom Network

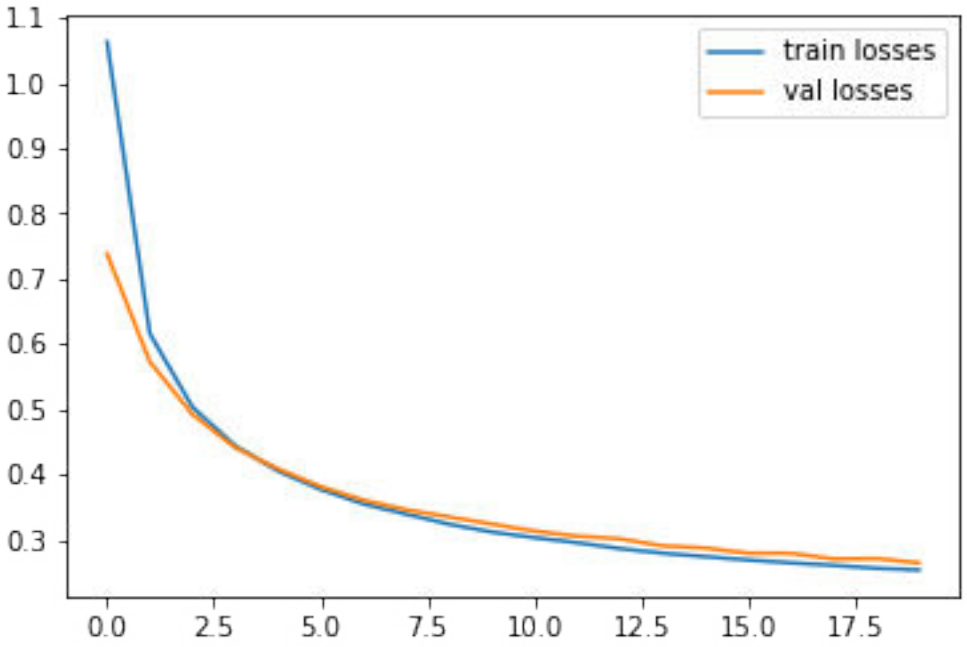
train

[ ]:
```
# create network model
cnet = CustomTNet(n_classes=10)
# loss function
crit = nn.CrossEntropyLoss()
# optimizer
opt = optim.SGD(cnet.parameters(), lr=10e-3, momentum=0.9, weight_decay=10e-4)
```

[ ]:
```
train(
    cnet,
    "custom_target-net.pt"
)
```

```
!! Training start...
Epoch 0:
 - loss: train: 1.0626860184872404, val: 0.7382570010550479
 - ACC: train: 71.5342420212766, val: 81.90539513677813
* Early Stopping reset...
* Early Stopping reset...
Epoch 2:
 - loss: train: 0.5036902589366791, val: 0.4925938667135036
 - ACC: train: 87.94464760638297, val: 88.15064589665653
* Early Stopping reset...
* Early Stopping reset...
Epoch 4:
 - loss: train: 0.4059229252978842, val: 0.40930509694079137
 - ACC: train: 90.10139627659575, val: 89.97910334346504
* Early Stopping reset...
* Early Stopping reset...
Epoch 6:
 - loss: train: 0.35556716170716796, val: 0.36140764742455583
 - ACC: train: 91.38131648936171, val: 91.34213525835865
* Early Stopping reset...
* Early Stopping reset...
Epoch 8:
 - loss: train: 0.324561347114913, val: 0.33579297807622466
 - ACC: train: 92.29554521276596, val: 91.73632218844985
* Early Stopping reset...
* Early Stopping reset...
Epoch 10:
 - loss: train: 0.30394071705163794, val: 0.31402236064697836
 - ACC: train: 92.6737034574468, val: 92.4059650455927
* Early Stopping reset...
* Early Stopping reset...
Epoch 12:
 - loss: train: 0.28718854399754645, val: 0.30224527069862855
 - ACC: train: 93.16821808510639, val: 92.66717325227964
* Early Stopping reset...
* Early Stopping reset...
Epoch 14:
 - loss: train: 0.2748827933472522, val: 0.288106603825346
 - ACC: train: 93.55053191489361, val: 92.95212765957447
* Early Stopping reset...
* Early Stopping reset...
Epoch 16:
 - loss: train: 0.2654368886922268, val: 0.2800272652443419
```

```
 - ACC: train: 93.66688829787235, val: 93.36056231003039
* Early Stopping reset...
* Early Stopping reset...
Epoch 18:
 - loss: train: 0.2570429951586622, val: 0.2718337352605576
 - ACC: train: 93.9453125, val: 93.93047112462007
* Early Stopping increase...
* Early Stopping reset...
!! Training stop...
```



test

[ ]:
```
# create network model
cnet = CustomTNet(n_classes=10)
cnet.load_state_dict(torch.load("custom_target-net.pt"))
cnet = cnet.to('cpu')

print('Test accuracy:', test(cnet))
```

```
Test accuracy: 94.59061550151975
```

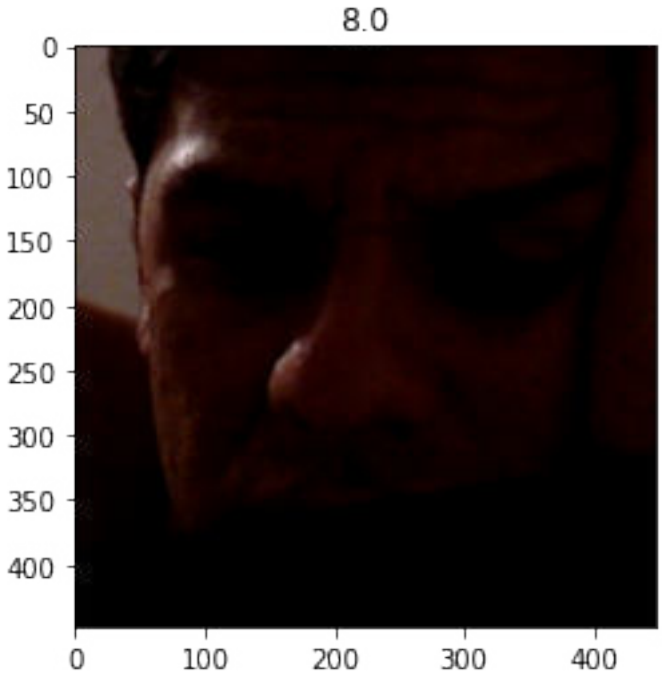
test one sample

[31]:
```
random_idx = np.random.randint(3000)
x, y = target_x[random_idx], target_y[random_idx]

x_t = transform(x).unsqueeze(0)  # [batch, channels, H, W]
y_hat = torch.argmax(cnet(x_t)).detach().numpy()
```

```
while y == y_hat:
    random_idx = np.random.randint(3000)
    x, y = target_x[random_idx], target_y[random_idx]

    x_t = transform(x).unsqueeze(0)  # [batch, channels, H, W]
    y_hat = torch.argmax(cnet(x_t)).detach().numpy()

print(random_idx)
```

```
524
```

[32]:
```
plt.imshow(target_x[random_idx])
plt.title(target_y[random_idx])
plt.show()
```



[33]:
```
# won't map to device since CPU is used for inference :-?
x_t = transform(target_x[random_idx]).unsqueeze(0)  # [batch, channels, H, W]
y_hat = torch.argmax(underfit_net(x_t)).detach().numpy()
print(y_hat)
```

```
6
```

5.2.4 EvalClassifier (ResNet152)

train

[ ]:
```
# create network model
ec_net = ResNet152(n_classes=10)
# loss function
crit = nn.CrossEntropyLoss()
# optimizer
opt = optim.SGD(ec_net.parameters(), lr=10e-3, momentum=0.9, weight_decay=10e-4)
```

[ ]:
```
train(
    ec_net,
    "eval_class-net.pt"
)
```

test

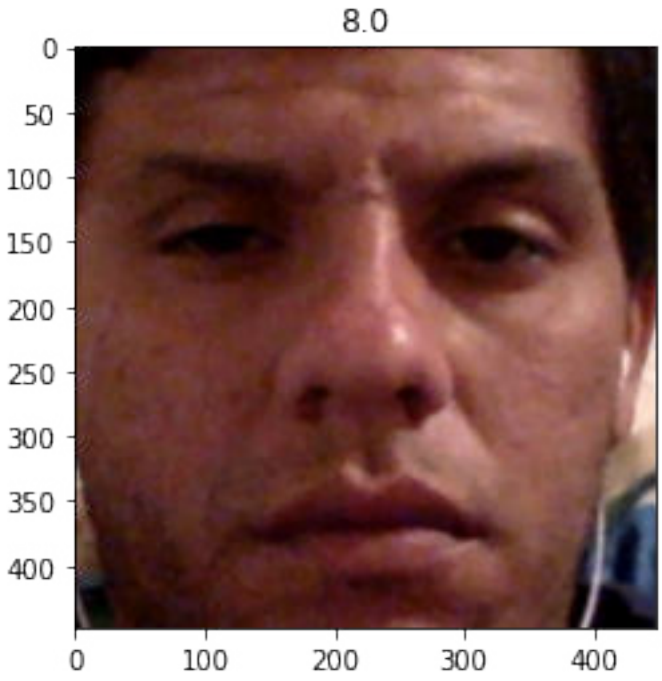
[ ]:
```
# create network model
ec_net = ResNet152(n_classes=10)
ec_net.load_state_dict(torch.load("eval_class-net.pt"))
ec_net = ec_net.to('cpu')

print('Test accuracy:', test(ec_net))
```

```
Test accuracy: 99.9002659574468
```

test one sample

[38]:
```
random_idx = np.random.randint(3000)
x, y = target_x[random_idx], target_y[random_idx]
```

[39]:
```
plt.imshow(x)
plt.title(y)
plt.show()
```



[40]:
```
# won't map to device since CPU is used for inference :-?
x_t = transform(x).unsqueeze(0)  # [batch, channels, H, W]
y_hat = torch.argmax(ec_net(x_t)).detach().numpy()
print(y_hat)
```

```
8
```

---

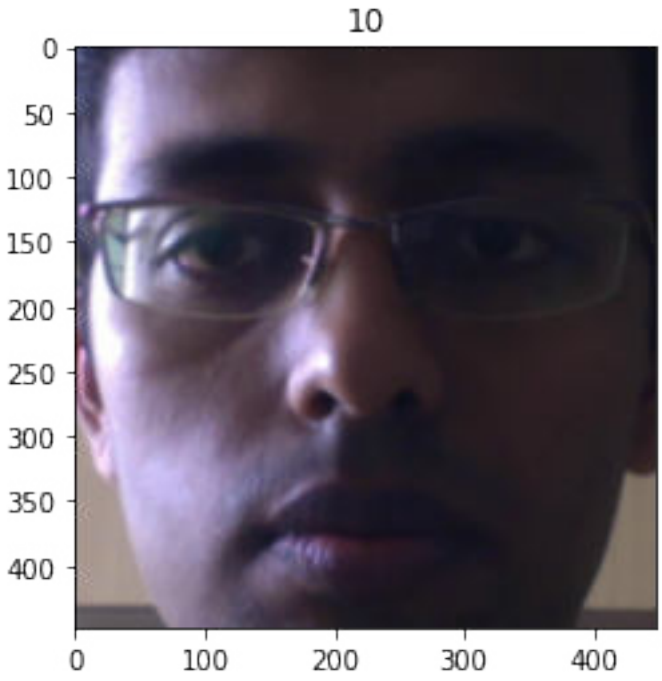
6 GAN

6.1 train

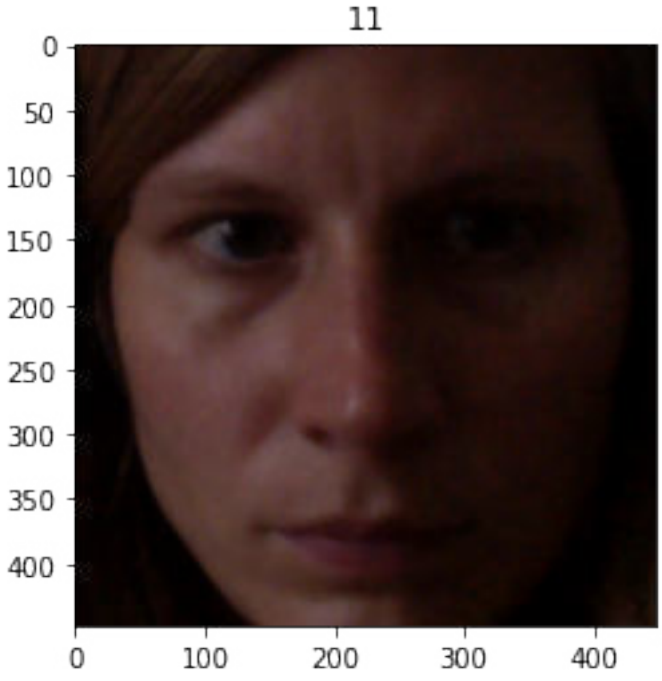
Check PWD in *Project* root!!

[ ]:
```
shadow_x, shadow_y = create_dataset(np.arange(10, 15))
```

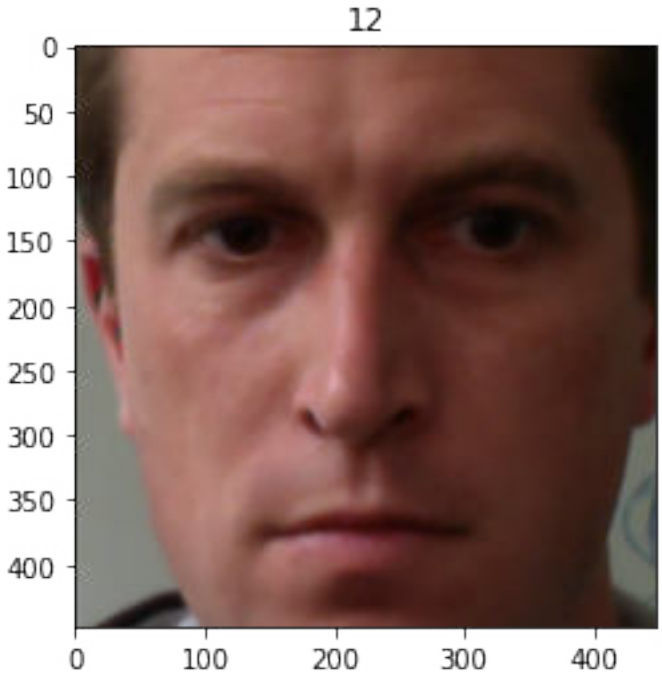
```
$ 10...3000
```


10
0
50
100
150
200
250
300
350
400
0
100
200
300
400

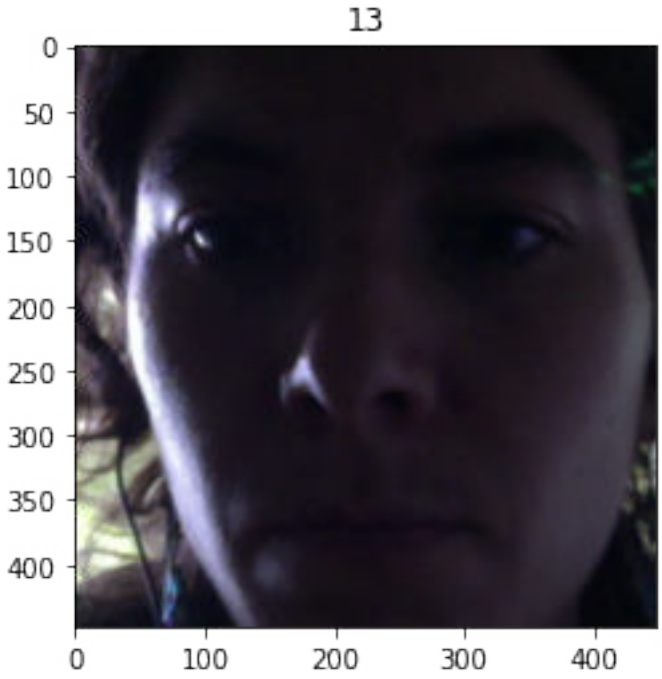
```
$ 11...6000
```


11
0
50
100
150
200
250
300
350
400
0
100
200
300
400

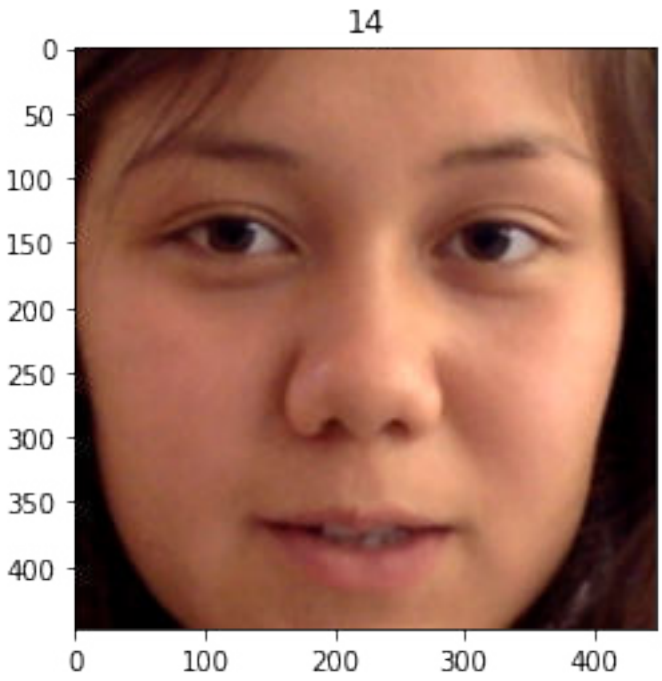
$ 12...9000



$ 13...12000



```
$ 14...15000
```



[ ]:
```
%pwd
```

[ ]:
```
%cd src/
```

[ ]:
```
from gan import Gen, Discr
from gan import Utils
```

[ ]:
```
lr = 1e-4
batch_size = 256
z_dim = 100
epochs = 200
n_critic = 5

# load data properly
gan_data = DS(shadow_x, shadow_y, transformGAN)
gan_loader = DataLoader(gan_data, batch_size=batch_size, shuffle=True)

dst_folder = 'gan_train224x224'

print("<<< Training [GAN] >>>")
```

```
G = Gen(z_dim)
DG = Discr(3)

G = torch.nn.DataParallel(G).cuda()
DG = torch.nn.DataParallel(DG).cuda()

g_optimizer = torch.optim.Adam(G.parameters(), lr=lr, betas=(0.5, 0.999))
dg_optimizer = torch.optim.Adam(DG.parameters(), lr=lr, betas=(0.5, 0.999))

step = 0

for epoch in range(epochs):
    start = time.time()
    for i, (imgs, _) in enumerate(gan_loader):
        step += 1
        imgs = imgs.cuda()
        bs = imgs.size(0)  # batch size

        Utils.freeze(G)
        Utils.unfreeze(DG)

        z = torch.randn(bs, z_dim).cuda()
        f_imgs = G(z)

        r_logit = DG(imgs)
        f_logit = DG(f_imgs)

        wd = r_logit.mean() - f_logit.mean()  # Wasserstein-1 Distance
        gp = Utils.gradient_penalty(imgs.data, f_imgs.data, DG=DG)
        dg_loss = - wd + gp * 10.0

        dg_optimizer.zero_grad()
        dg_loss.backward()
        dg_optimizer.step()

        # train G
        if step % n_critic == 0:
            Utils.freeze(DG)
            Utils.unfreeze(G)

            z = torch.randn(bs, z_dim).cuda()
            f_imgs = G(z)
            logit_dg = DG(f_imgs)
            # calculate g_loss
            g_loss = - logit_dg.mean()

            g_optimizer.zero_grad()
```

```
            g_loss.backward()
            g_optimizer.step()

    end = time.time()
    print('dT[{}]: {:.3f}'.format(epoch + 1, end - start))

    if (epoch + 1) % 10 == 0:
        z = torch.randn(32, z_dim).cuda()
        fake_image = G(z)
        tvls.save_image(fake_image.detach(), "{}/result_image_{}.png".
↪format(dst_folder, epoch + 1), nrow = 8, normalize=True)

    torch.save(G.state_dict(), "{}/G-net.tar".format(dst_folder))
    torch.save(DG.state_dict(), "{}/D-net.tar".format(dst_folder))
```

6.2 attack

Check PWD in *Project* root!!

6.2.1 setup

[9]:
```
%pwd
```

[9]:
```
'/content/drive/MyDrive/University/2. UniStuttgart MSc./Studies/2022W/Machine
Perception and Learning/Project'
```

[10]:
```
%cd src/
```

```
/content/drive/MyDrive/University/2. UniStuttgart MSc./Studies/2022W/Machine
Perception and Learning/Project/src
```

[11]:
```
from target_network import VGG16, CustomTNet, ResNet152

from gan import Gen, Discr
from gan import Utils
```

[12]:
```
gan_src_folder = 'gan_train224x224'
```

[13]:
```
identities = torch.arange(10)
```

[14]:
```
T = VGG16(n_classes=10).cuda()
T.load_state_dict(torch.load("target-net.pt"))

underfitT = VGG16(n_classes=10).cuda()
underfitT.load_state_dict(torch.load("target-net_underfit.pt"))

cT = CustomTNet(n_classes=10).cuda()
cT.load_state_dict(torch.load("custom_target-net.pt"))
```

```
E = ResNet152(n_classes=10).cuda()
E.load_state_dict(torch.load("eval_class-net.pt"))

G = Gen(100)
G = nn.DataParallel(G).cuda()
G.load_state_dict(torch.load('{}/G-net.tar'.format(gan_src_folder)))

D = Discr(3)
D = nn.DataParallel(D).cuda()
D.load_state_dict(torch.load('{}/D-net.tar'.format(gan_src_folder)))
```

```
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:208:
UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be
removed in the future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/torchvision/models/_utils.py:223:
UserWarning: Arguments other than a weight enum or `None` for 'weights' are
deprecated since 0.13 and may be removed in the future. The current behavior is
equivalent to passing `weights=VGG16_BN_Weights.IMAGENET1K_V1`. You can also use
`weights=VGG16_BN_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
```

[14]:
```
<All keys matched successfully>
```

[20]:
```
print('$$$$$\nT:', T)
print('$$$$$\nunderfitT:', underfitT)
print('$$$$$\ncT:', cT)
print('$$$$$\nE:', E)
print('$$$$$\nG:', G)
print('$$$$$\nD:', D)
```

```
$$$$$
T: VGG16(
  (feature): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
    (9): ReLU(inplace=True)
    (10): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (12): ReLU(inplace=True)
    (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (14): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (16): ReLU(inplace=True)
    (17): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (19): ReLU(inplace=True)
    (20): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (21): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (22): ReLU(inplace=True)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (24): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (26): ReLU(inplace=True)
    (27): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (28): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (29): ReLU(inplace=True)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (32): ReLU(inplace=True)
    (33): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (35): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (36): ReLU(inplace=True)
    (37): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (38): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (39): ReLU(inplace=True)
    (40): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (41): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (42): ReLU(inplace=True)
    (43): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
```

```
ceil_mode=False)
  )
  (bn): BatchNorm1d(25088, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (fc_layer): Linear(in_features=25088, out_features=10, bias=True)
)
$$$$$
underfitT: VGG16(
  (feature): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (9): ReLU(inplace=True)
    (10): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (12): ReLU(inplace=True)
    (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (14): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (16): ReLU(inplace=True)
    (17): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (19): ReLU(inplace=True)
    (20): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (21): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (22): ReLU(inplace=True)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (24): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (26): ReLU(inplace=True)
    (27): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
```

```
    (28): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (29): ReLU(inplace=True)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (32): ReLU(inplace=True)
    (33): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (35): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (36): ReLU(inplace=True)
    (37): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (38): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (39): ReLU(inplace=True)
    (40): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (41): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (42): ReLU(inplace=True)
    (43): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (bn): BatchNorm1d(25088, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (fc_layer): Linear(in_features=25088, out_features=10, bias=True)
)
$$$$$
cT: CustomTNet(
  (fa): Sequential(
    (0): Sequential(
      (0): MaxPool2d(kernel_size=28, stride=28, padding=0, dilation=1,
ceil_mode=False)
    )
  )
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (fc): Linear(in_features=192, out_features=10, bias=True)
)
$$$$$
E: ResNet152(
  (model): ResNet(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
```

```
ceil_mode=False)
    (layer1): Sequential(
      (0): Bottleneck(
        (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (downsample): Sequential(
          (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): Bottleneck(
        (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (2): Bottleneck(
        (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
    )
    (layer2): Sequential(
```

```
      (0): Bottleneck(
        (conv1): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (downsample): Sequential(
          (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): Bottleneck(
        (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (2): Bottleneck(
        (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (3): Bottleneck(
        (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (4): Bottleneck(
        (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (5): Bottleneck(
        (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (6): Bottleneck(
        (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
```

```
      (7): Bottleneck(
        (conv1): Conv2d(512, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(128, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn3): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
    )
    (layer3): Sequential(
      (0): Bottleneck(
        (conv1): Conv2d(512, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (downsample): Sequential(
          (0): Conv2d(512, 1024, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
```

```
      )
      (2): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (3): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (4): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (5): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
```

```
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (6): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (7): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (8): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (9): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (10): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (11): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (12): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (13): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (14): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
```

```
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (15): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (16): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (17): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
```

```
      )
      (18): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (19): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (20): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (21): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
```

```
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (22): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (23): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (24): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (25): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (26): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (27): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (28): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (29): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (30): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
```

```
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (31): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (32): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (33): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
```

```
      )
      (34): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (35): Bottleneck(
        (conv1): Conv2d(1024, 256, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(256, 1024, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
    )
    (layer4): Sequential(
      (0): Bottleneck(
        (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (downsample): Sequential(
```

```
          (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): Bottleneck(
        (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
      (2): Bottleneck(
        (conv1): Conv2d(2048, 512, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (conv3): Conv2d(512, 2048, kernel_size=(1, 1), stride=(1, 1),
bias=False)
        (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
      )
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
    (fc): Linear(in_features=2048, out_features=10, bias=True)
  )
)
$$$$$
G: DataParallel(
  (module): Gen(
    (l1): Sequential(
      (0): Linear(in_features=100, out_features=8192, bias=False)
      (1): BatchNorm1d(8192, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      (2): ReLU()
```

```
    )
    (l2_5): Sequential(
      (0): Sequential(
        (0): ConvTranspose2d(512, 256, kernel_size=(15, 15), stride=(2, 2),
padding=(2, 2), output_padding=(1, 1), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (2): ReLU()
      )
      (1): Sequential(
        (0): ConvTranspose2d(256, 128, kernel_size=(15, 15), stride=(2, 2),
padding=(2, 2), output_padding=(1, 1), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (2): ReLU()
      )
      (2): Sequential(
        (0): ConvTranspose2d(128, 64, kernel_size=(15, 15), stride=(2, 2),
padding=(2, 2), output_padding=(1, 1), bias=False)
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (2): ReLU()
      )
      (3): ConvTranspose2d(64, 16, kernel_size=(11, 11), stride=(2, 2),
padding=(2, 2), output_padding=(1, 1))
      (4): ConvTranspose2d(16, 3, kernel_size=(15, 15), stride=(1, 1))
      (5): Sigmoid()
    )
  )
)
$$$$$
D: DataParallel(
  (module): Discr(
    (ls): Sequential(
      (0): Conv2d(3, 16, kernel_size=(15, 15), stride=(1, 1))
      (1): Conv2d(16, 64, kernel_size=(11, 11), stride=(2, 2), padding=(2, 2))
      (2): LeakyReLU(negative_slope=0.2)
      (3): Sequential(
        (0): Conv2d(64, 128, kernel_size=(15, 15), stride=(2, 2), padding=(2,
2))
        (1): InstanceNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
        (2): LeakyReLU(negative_slope=0.2)
      )
      (4): Sequential(
        (0): Conv2d(128, 256, kernel_size=(15, 15), stride=(2, 2), padding=(2,
2))
        (1): InstanceNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=False)
        (2): LeakyReLU(negative_slope=0.2)
      )
      (5): Sequential(
        (0): Conv2d(256, 512, kernel_size=(15, 15), stride=(2, 2), padding=(2,
2))
        (1): InstanceNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=False)
        (2): LeakyReLU(negative_slope=0.2)
      )
      (6): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1))
    )
  )
)
```

[16]:
```python
def attack(target_net, eval_net=None, g_net=None, d_net=None, iden=None,␣
↪verbose=False):
    lr = 1e-2
    momentum = 0.9
    z_size = 100

    iden = iden.long().cuda()
    criterion = nn.CrossEntropyLoss().cuda()
    identity_number = iden.shape[0]  # batch size

    # set networks to inference mode
    target_net.eval()
    g_net.eval()
    d_net.eval()

    max_score = torch.zeros(identity_number)
    max_iden = torch.zeros(identity_number)
    z_hat = torch.zeros(identity_number, z_size)

    it = 1
    for random_seed in range(5):
        tf = time.time()

        torch.manual_seed(random_seed)
        torch.cuda.manual_seed(random_seed)
        np.random.seed(random_seed)

        z = torch.randn(identity_number, z_size).cuda().float()
        z.requires_grad = True
        v = torch.zeros(identity_number, z_size).cuda().float()

        for i in range(1_000):
```

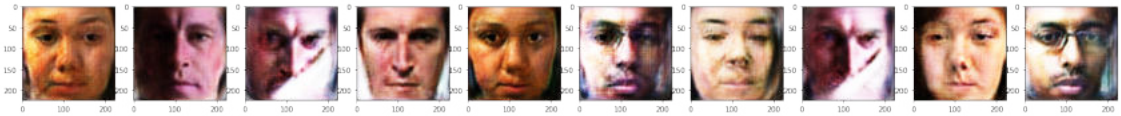
```
            fake = g_net(z)
            label = d_net(fake)
            out = target_net(fake)

            if z.grad is not None:
                z.grad.data.zero_()

            Prior_Loss = - label.mean()
            Iden_Loss = criterion(out, iden)
            Total_Loss = Prior_Loss + 100 * Iden_Loss

            Total_Loss.backward()

            # SGD
            v_prev = v.clone()
            gradient = z.grad.data
            v = momentum * v - lr * gradient
            z = z + ( - momentum * v_prev + (1 + momentum) * v)
            z = torch.clamp(z.detach(), -1, 1).float()
            z.requires_grad = True

            if (i + 1) % 300 == 0 or (verbose and i < 10):
                fake_img = g_net(z.detach())

                f, axarr = plt.subplots(1, 10, figsize=(27, 48))
                for _i in range(10):
                    axarr[_i].imshow(fake_img[_i].permute(1, 2, 0).cpu().
↪detach())
                plt.show()
                eval_prob = target_net(fake_img)
                eval_iden = torch.argmax(eval_prob, dim=1)
                print(eval_iden)
                acc = iden.eq(eval_iden.long()).sum().item() * 1.0 /␣
↪identity_number
                print("[{}.{}]\tPrior Loss: {:.2f}\tIden Loss: {:.2f}\tAttack␣
↪Acc: {:.2f}".format(it, i + 1, Prior_Loss.item(), Iden_Loss.item(), acc))

            # print(i, end=', ')

        fake = g_net(z)
        score = target_net(fake)
        eval_prob = target_net(fake)
        eval_iden = torch.argmax(eval_prob, dim=1)

        cnt = 0
        for i in range(identity_number):
            gt = iden[i].item()
```

```
            # if after z => G => T the prediction is more certain that ID is i␣
↪(in [0, 10])
            if score[i, i].item() > max_score[i].item():
                max_score[i] = score[i, i]
                max_iden[i] = eval_iden[i]
                # store the new optimal Z's
                z_hat[i, :] = z[i, :]
            if eval_iden[i].item() == gt:
                cnt += 1

        interval = time.time() - tf
        print("[{}]\tTime:{:.2f}\tAcc:{:.2f}\t".format(it, interval, cnt * 1.0 /␣
↪identity_number))
        print('#' * 100)

        # increment to the next seed
        it += 1

    correct = 0
    for i in range(identity_number):
        gt = iden[i].item()
        if max_iden[i].item() == gt:
            correct += 1
    acc = correct * 1.0 / identity_number
    print("Attack Accuracy: {:.2f}".format(acc))

    return z_hat
```

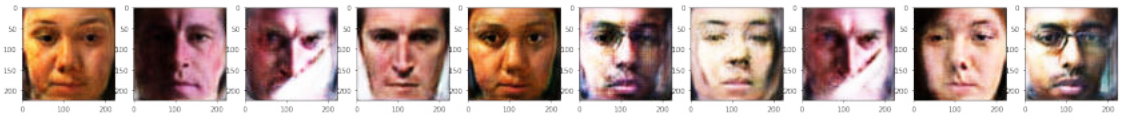
6.2.2 Target Networks (VGG16s)

VGG16 (fitted)

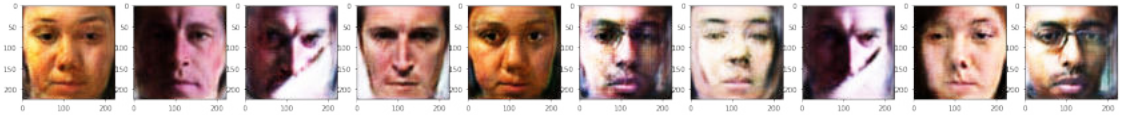
[ ]:
```
optim_z_s = attack(
    target_net=T,
    g_net=G,
    d_net=D,
    iden=identities
)
```



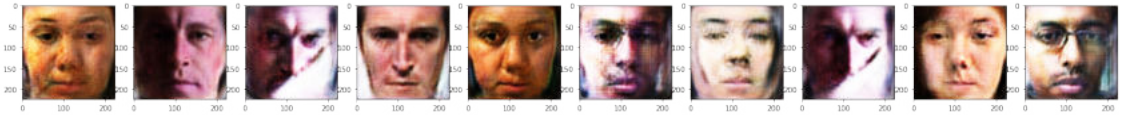
```
tensor([1, 1, 1, 3, 1, 5, 1, 1, 1, 1], device='cuda:0')
[1.300] Prior Loss: 189.95      Iden Loss: 3.63 Attack Acc: 0.30
```



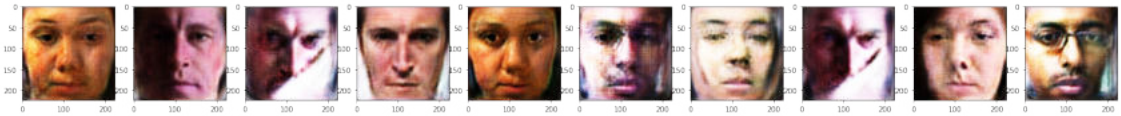
```
tensor([1, 1, 1, 3, 1, 5, 1, 1, 1, 1], device='cuda:0')
[1.600] Prior Loss: 179.74      Iden Loss: 3.25 Attack Acc: 0.30
```



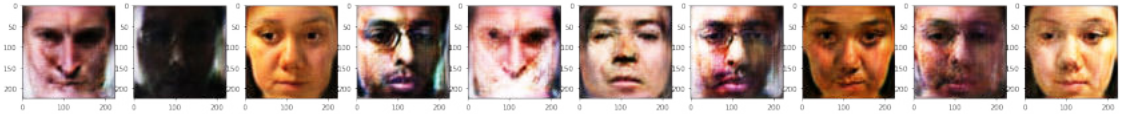
```
tensor([1, 1, 1, 3, 1, 5, 2, 1, 1, 1], device='cuda:0')
[1.900] Prior Loss: 182.66      Iden Loss: 2.63 Attack Acc: 0.30
```



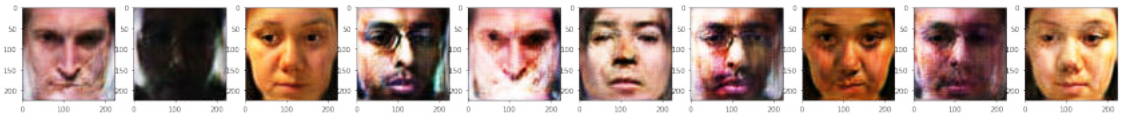
```
tensor([1, 1, 1, 3, 1, 5, 2, 1, 1, 1], device='cuda:0')
[1.1200]        Prior Loss: 182.18      Iden Loss: 2.94 Attack Acc: 0.30
```



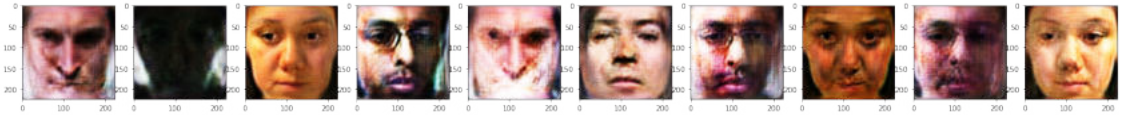
```
tensor([1, 1, 1, 3, 1, 5, 1, 1, 1, 1], device='cuda:0')
[1.1500]        Prior Loss: 181.76      Iden Loss: 3.39 Attack Acc: 0.30
[1]     Time:108.24     Acc:0.30
################################################################################
####################
```



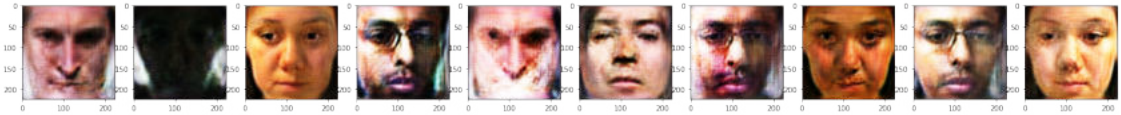
```
tensor([1, 1, 2, 3, 1, 5, 1, 1, 8, 1], device='cuda:0')
[2.300] Prior Loss: 205.32      Iden Loss: 1.97 Attack Acc: 0.50
```



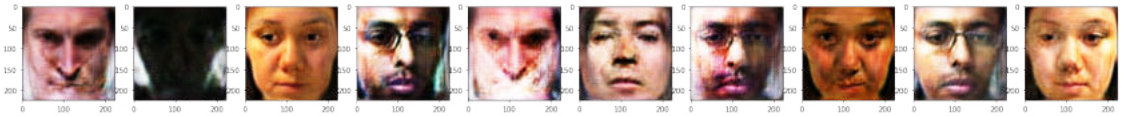
```
tensor([1, 1, 2, 3, 1, 5, 1, 1, 8, 1], device='cuda:0')
[2.600] Prior Loss: 205.92      Iden Loss: 2.04 Attack Acc: 0.50
```



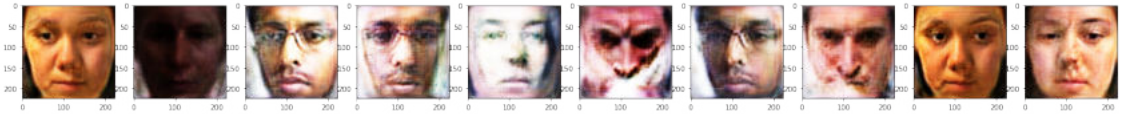
```
tensor([1, 1, 2, 3, 1, 5, 1, 1, 8, 1], device='cuda:0')
[2.900] Prior Loss: 193.79      Iden Loss: 1.99 Attack Acc: 0.50
```



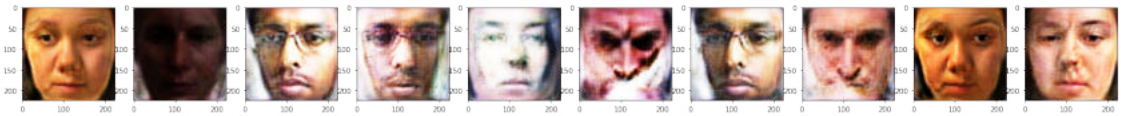
```
tensor([1, 1, 2, 3, 1, 5, 1, 1, 1, 1], device='cuda:0')
[2.1200]        Prior Loss: 203.41      Iden Loss: 2.03 Attack Acc: 0.40
```



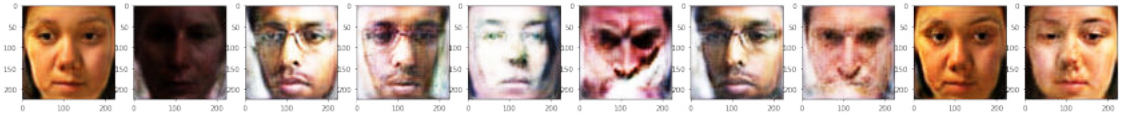
```
tensor([1, 1, 2, 3, 1, 5, 1, 1, 1, 1], device='cuda:0')
[2.1500]        Prior Loss: 199.90      Iden Loss: 1.88 Attack Acc: 0.40
[2]     Time:108.81     Acc:0.40
################################################################################
####################
```



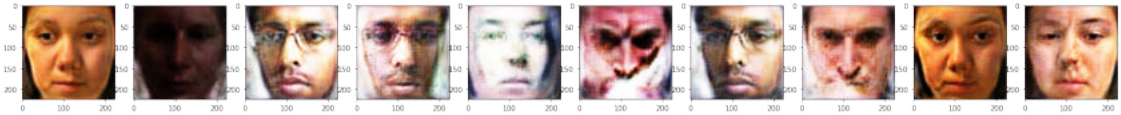
```
tensor([1, 1, 2, 3, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.300] Prior Loss: 176.75      Iden Loss: 2.73 Attack Acc: 0.30
```



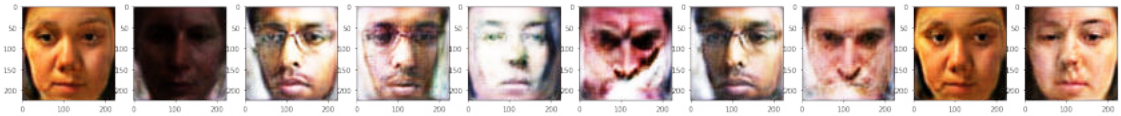
```
tensor([1, 1, 2, 3, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.600] Prior Loss: 175.33      Iden Loss: 2.47 Attack Acc: 0.30
```



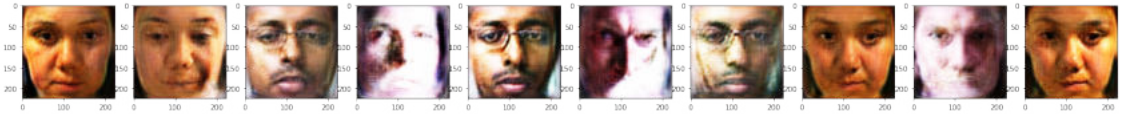
```
tensor([1, 1, 2, 3, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.900] Prior Loss: 176.12      Iden Loss: 2.27 Attack Acc: 0.30
```



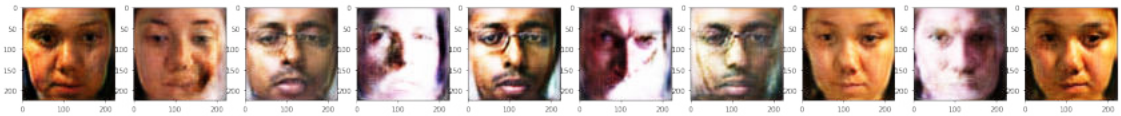
```
tensor([1, 1, 2, 3, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.1200]        Prior Loss: 172.99      Iden Loss: 2.31 Attack Acc: 0.30
```



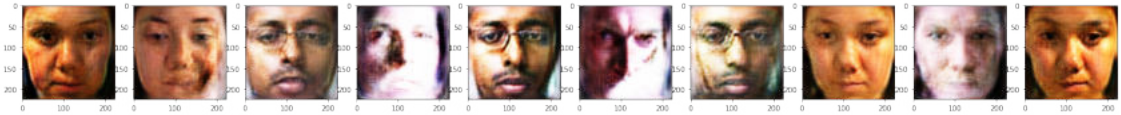
```
tensor([1, 1, 2, 3, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.1500]        Prior Loss: 177.57      Iden Loss: 2.40 Attack Acc: 0.30
[3]     Time:108.36     Acc:0.30
################################################################################
####################
```



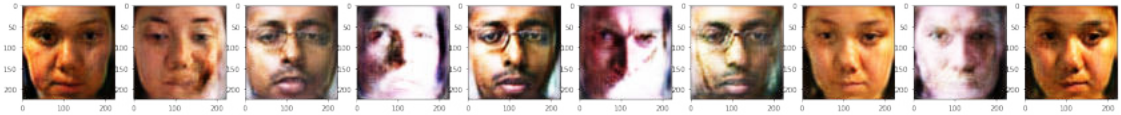
```
tensor([1, 1, 2, 3, 4, 1, 1, 1, 1, 1], device='cuda:0')
[4.300] Prior Loss: 201.18      Iden Loss: 2.25 Attack Acc: 0.40
```



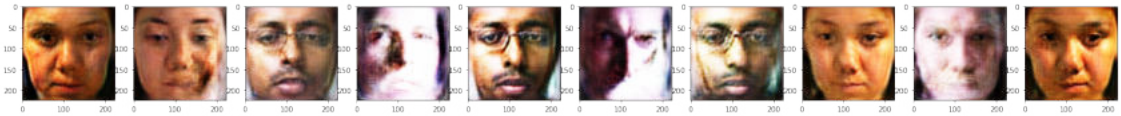
```
tensor([1, 1, 2, 3, 4, 1, 1, 1, 1, 1], device='cuda:0')
[4.600] Prior Loss: 193.89      Iden Loss: 2.02 Attack Acc: 0.40
```



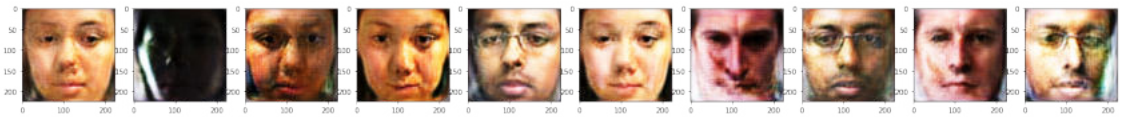
```
tensor([1, 1, 2, 3, 4, 1, 1, 1, 1, 1], device='cuda:0')
[4.900] Prior Loss: 191.63      Iden Loss: 1.85 Attack Acc: 0.40
```



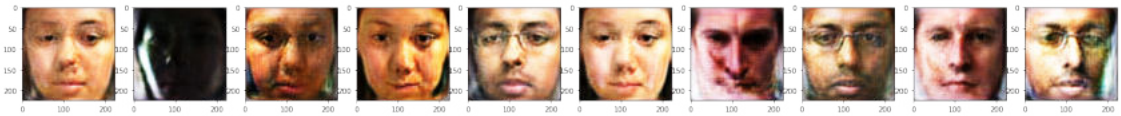
```
tensor([1, 1, 2, 3, 4, 1, 1, 1, 1, 1], device='cuda:0')
[4.1200]        Prior Loss: 191.04      Iden Loss: 2.13 Attack Acc: 0.40
```



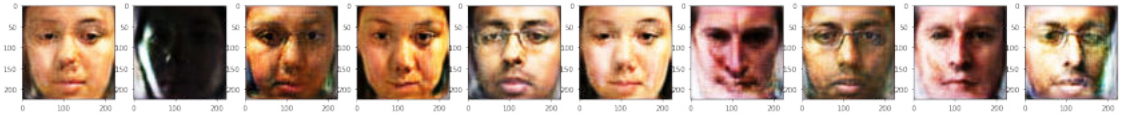
```
tensor([1, 1, 2, 3, 4, 1, 1, 1, 1, 1], device='cuda:0')
[4.1500]        Prior Loss: 190.49      Iden Loss: 1.96 Attack Acc: 0.40
[4]     Time:108.71     Acc:0.40
################################################################################
####################
```



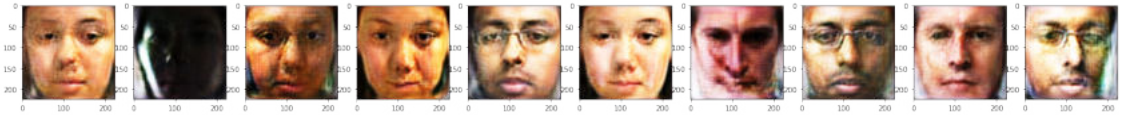
```
tensor([1, 1, 1, 3, 1, 5, 1, 1, 1, 1], device='cuda:0')
[5.300] Prior Loss: 202.62      Iden Loss: 2.19 Attack Acc: 0.30
```



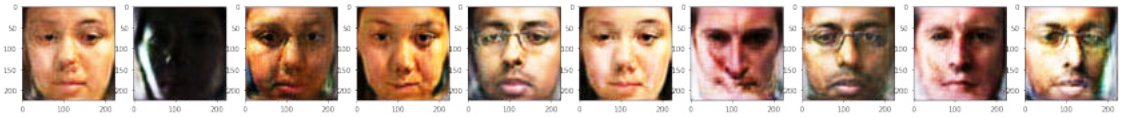
```
tensor([1, 1, 1, 3, 1, 5, 1, 1, 1, 1], device='cuda:0')
[5.600] Prior Loss: 209.03      Iden Loss: 2.50 Attack Acc: 0.30
```



```
tensor([1, 1, 1, 3, 1, 5, 1, 1, 1, 1], device='cuda:0')
[5.900] Prior Loss: 202.07      Iden Loss: 1.98 Attack Acc: 0.30
```



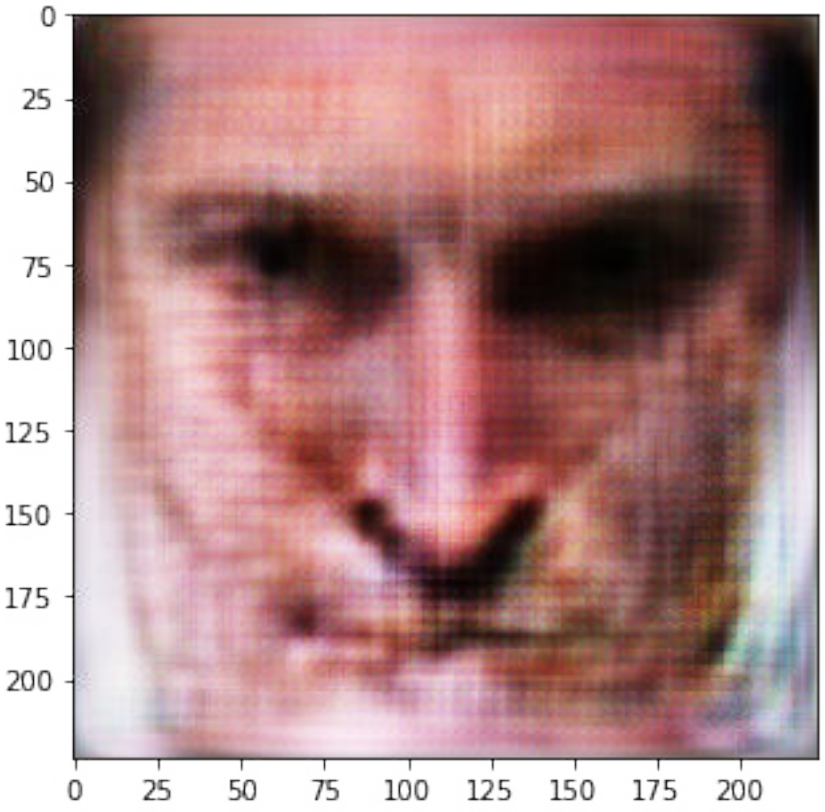
```
tensor([1, 1, 1, 3, 1, 5, 1, 1, 1, 1], device='cuda:0')
[5.1200]        Prior Loss: 200.09      Iden Loss: 2.01 Attack Acc: 0.30
```



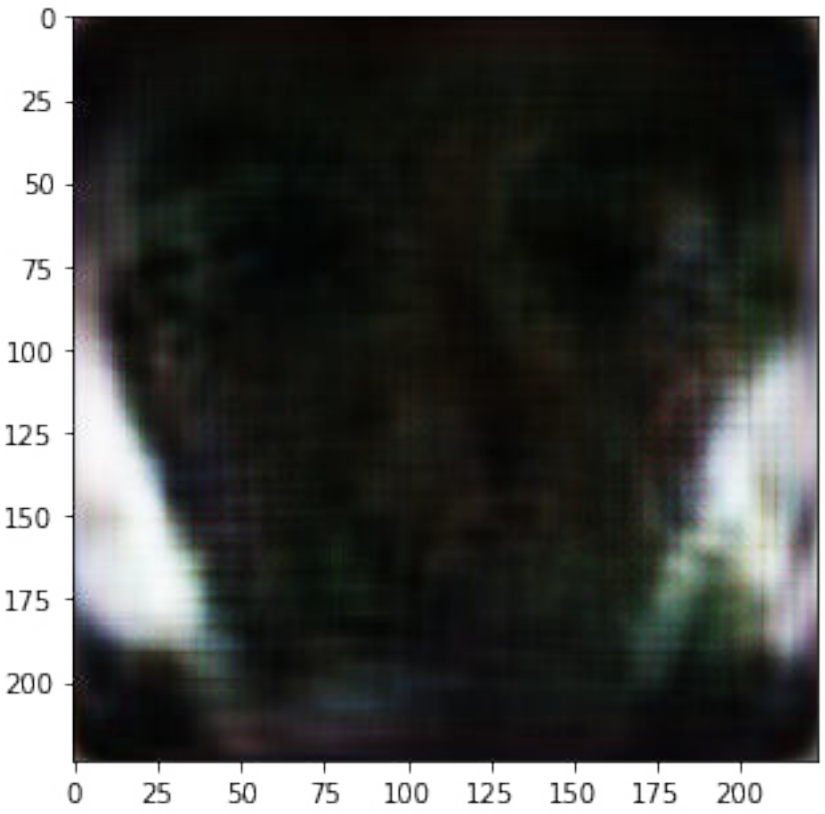
```
tensor([1, 1, 1, 3, 1, 5, 1, 1, 1, 1], device='cuda:0')
[5.1500]        Prior Loss: 205.35      Iden Loss: 1.94 Attack Acc: 0.30
[5]     Time:108.70     Acc:0.30
################################################################################
####################
Attack Accuracy: 0.50
```

[ ]:
```python
for i in range(10):
    z = optim_z_s[i]
    fake = G(z.unsqueeze(0))
    plt.imshow(fake.squeeze(0).permute(1, 2, 0).cpu().detach())
    plt.show()
    pred = T(fake)
    pred_id = torch.argmax(pred, dim=1)
```

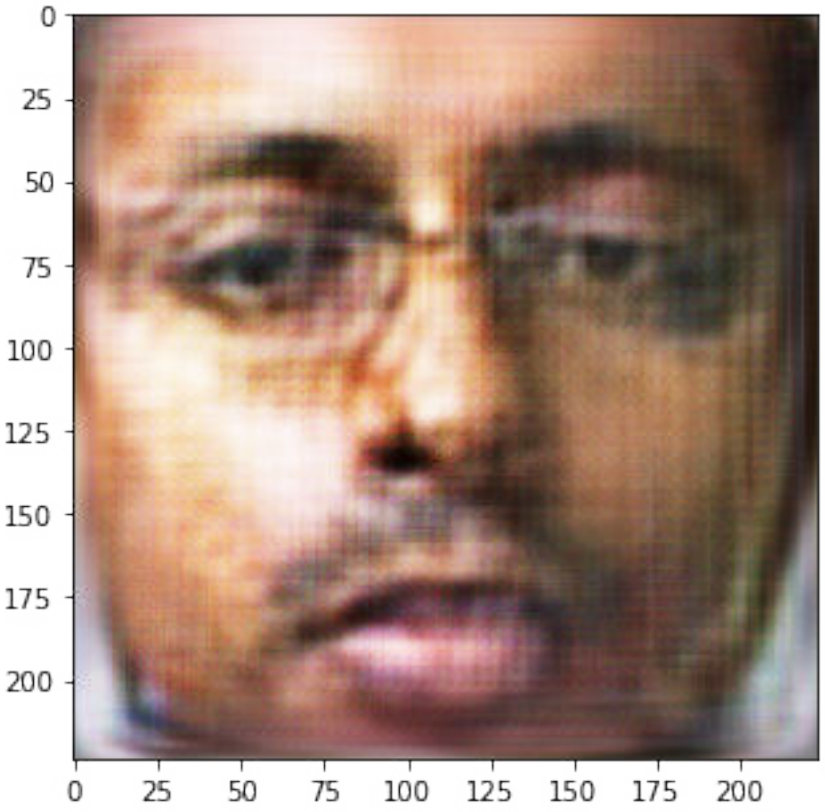
```
    print('Expected ID: {}, resulting ID: {}'.format(i, pred_id))
```



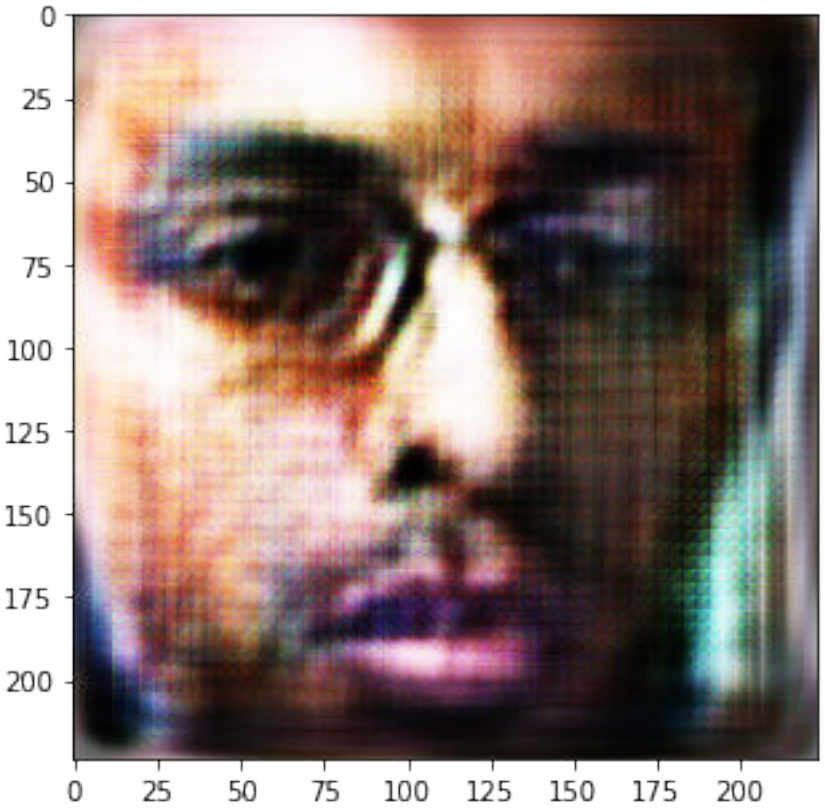

```
Expected ID: 0, resulting ID: tensor([1], device='cuda:0')
```



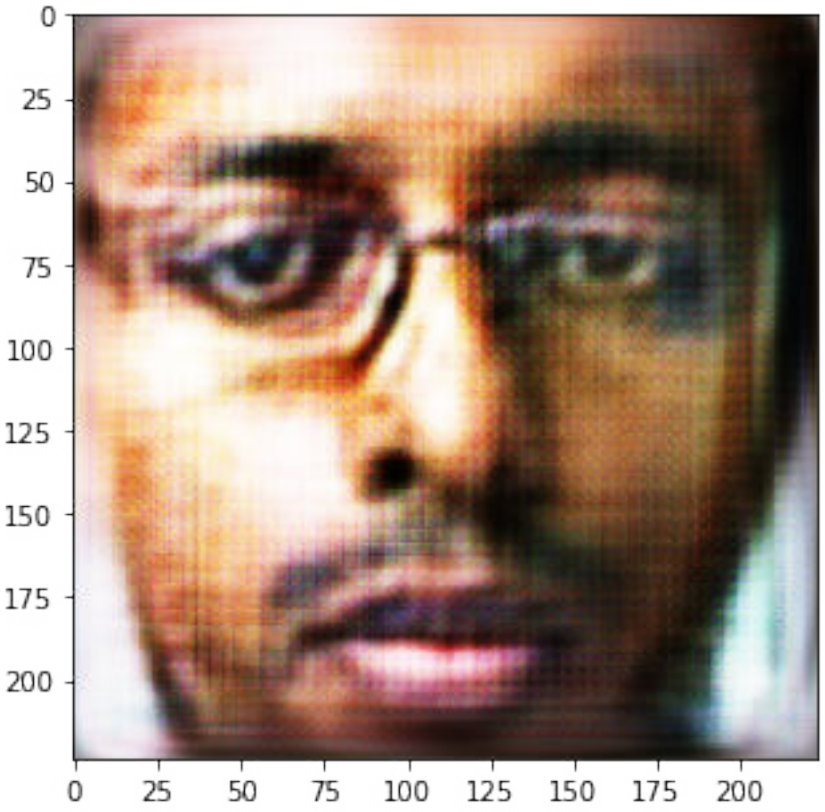

```
Expected ID: 1, resulting ID: tensor([1], device='cuda:0')
```



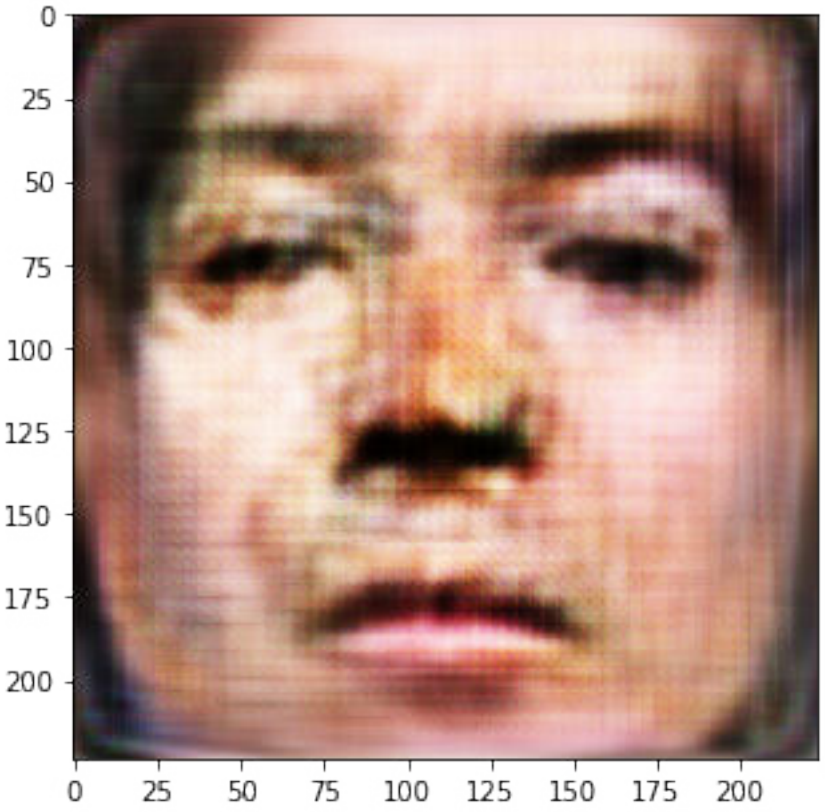

```
Expected ID: 2, resulting ID: tensor([2], device='cuda:0')
```



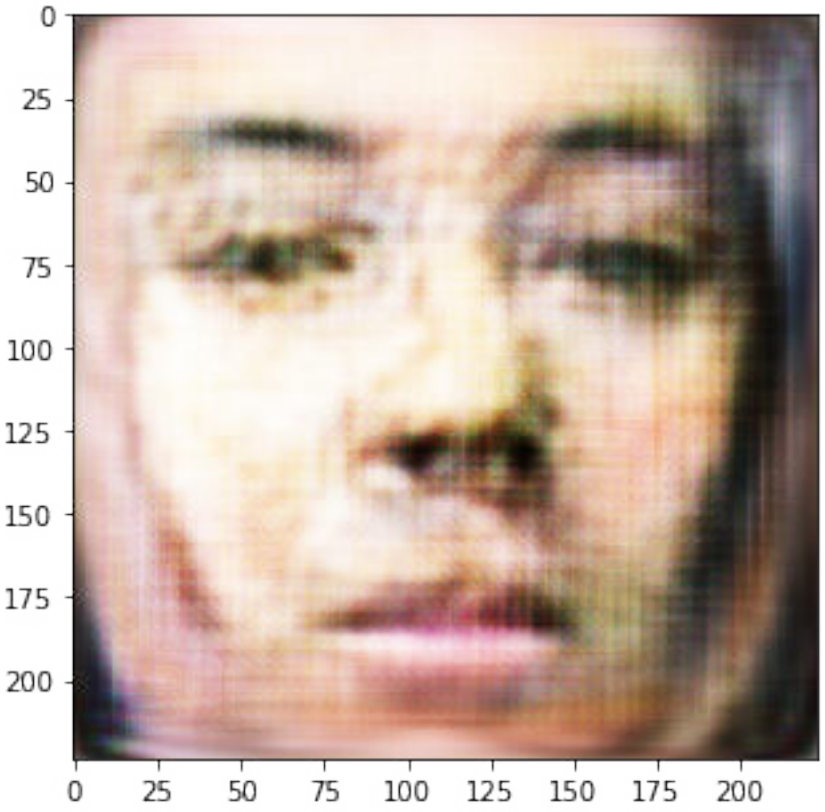

```
Expected ID: 3, resulting ID: tensor([3], device='cuda:0')
```



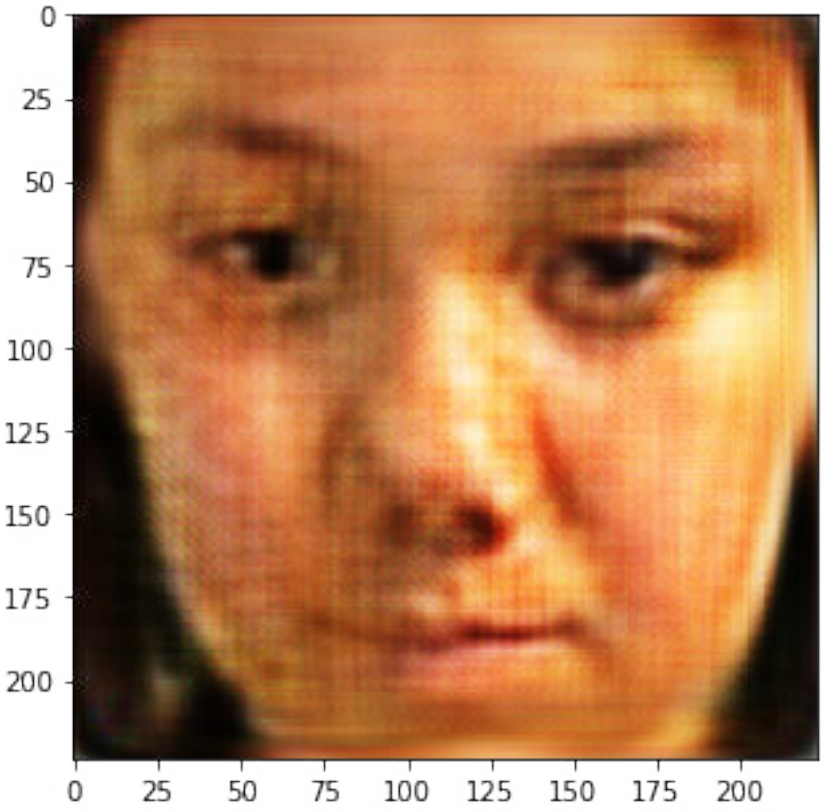

```
Expected ID: 4, resulting ID: tensor([4], device='cuda:0')
```



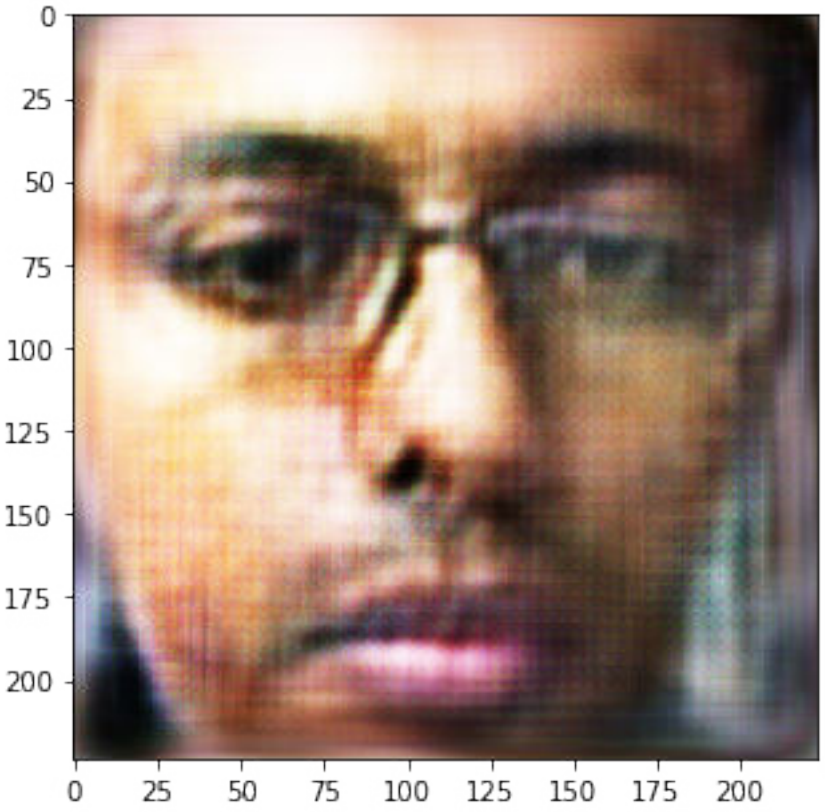

```
Expected ID: 5, resulting ID: tensor([5], device='cuda:0')
```



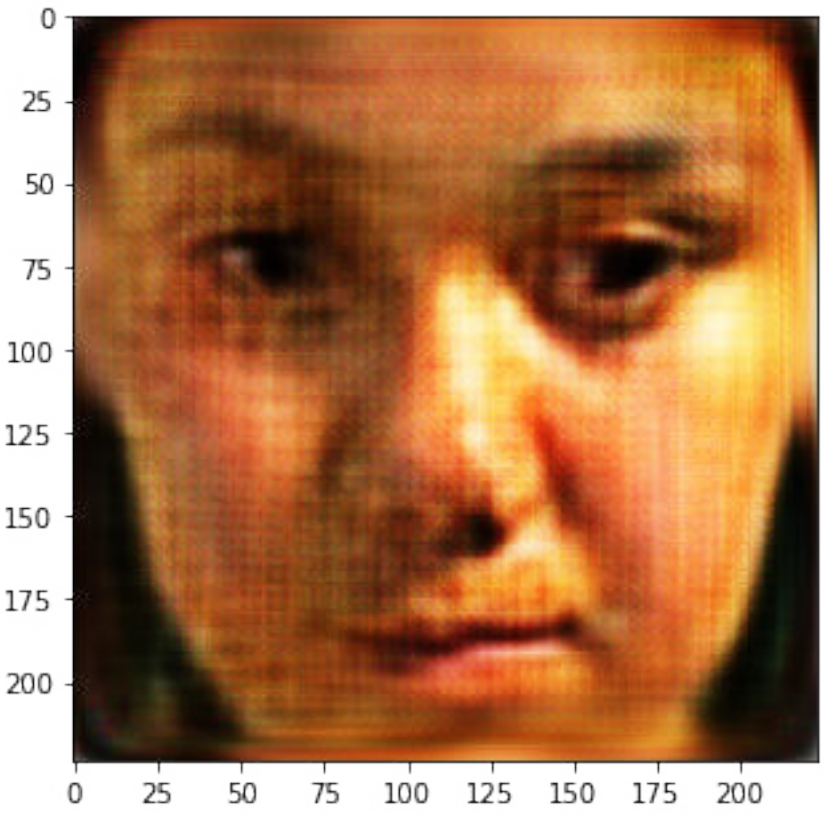

```
Expected ID: 6, resulting ID: tensor([1], device='cuda:0')
```




```
Expected ID: 7, resulting ID: tensor([1], device='cuda:0')
```



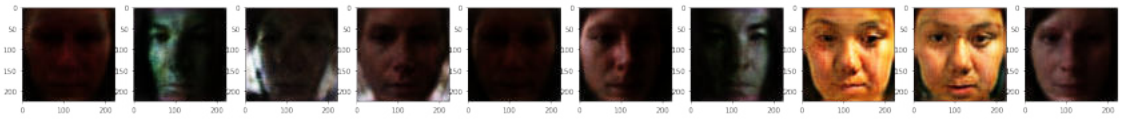

```
Expected ID: 8, resulting ID: tensor([1], device='cuda:0')
```



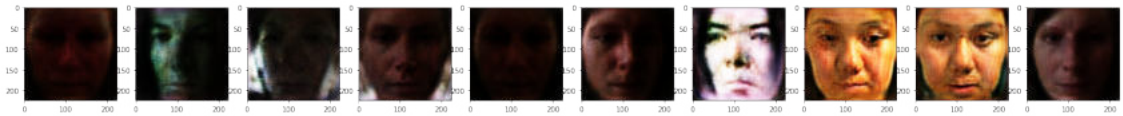

```
Expected ID: 9, resulting ID: tensor([1], device='cuda:0')
```

VGG16 (underfitted)

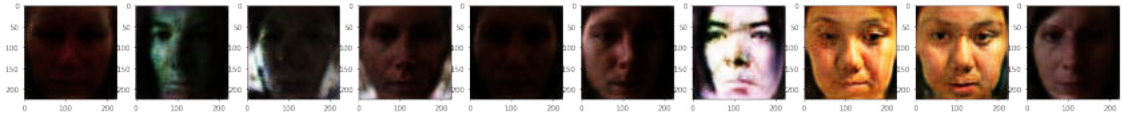
[20]:
```
optim_z_s = attack(
    target_net=underfitT,
    g_net=G,
    d_net=D,
    iden=identities
)
```



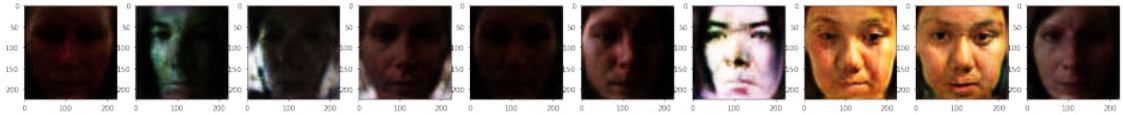
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 6, 1], device='cuda:0')
[1.300] Prior Loss: 166.99      Iden Loss: 8.46 Attack Acc: 0.10
```



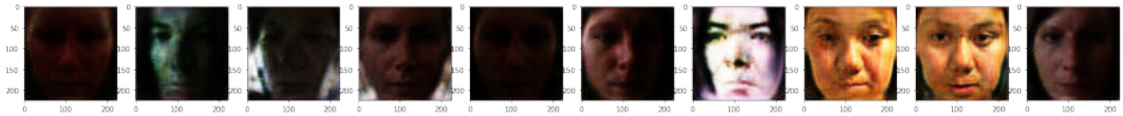
```
tensor([1, 1, 1, 1, 1, 1, 6, 1, 6, 1], device='cuda:0')
[1.600] Prior Loss: 164.48      Iden Loss: 8.05 Attack Acc: 0.20
```



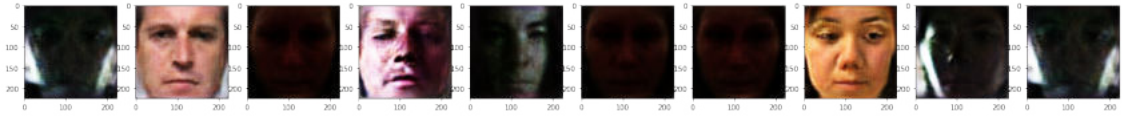
```
tensor([1, 1, 1, 1, 1, 1, 6, 1, 6, 1], device='cuda:0')
[1.900] Prior Loss: 163.12      Iden Loss: 8.04 Attack Acc: 0.20
```



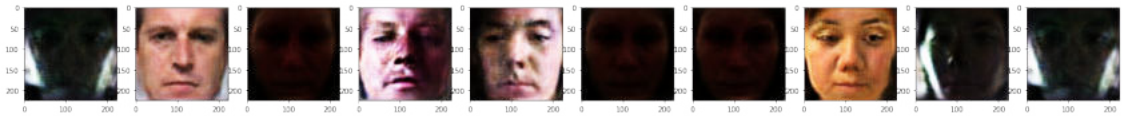
```
tensor([1, 1, 1, 1, 1, 1, 6, 1, 6, 1], device='cuda:0')
[1.1200]        Prior Loss: 162.70      Iden Loss: 8.05 Attack Acc: 0.20
```



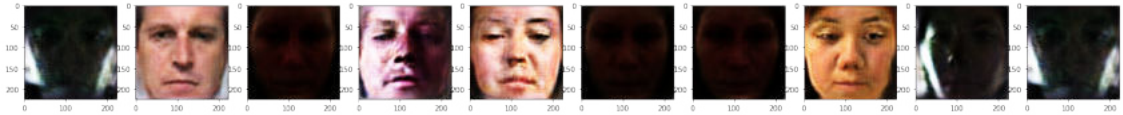
```
tensor([1, 1, 1, 1, 1, 1, 6, 1, 6, 1], device='cuda:0')
[1.1500]        Prior Loss: 162.28      Iden Loss: 8.05 Attack Acc: 0.20
[1]     Time:888.11     Acc:0.20
################################################################################
####################
```



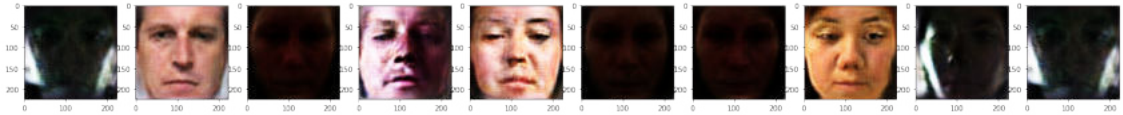
```
tensor([1, 1, 1, 3, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.300] Prior Loss: 145.22      Iden Loss: 8.32 Attack Acc: 0.20
```



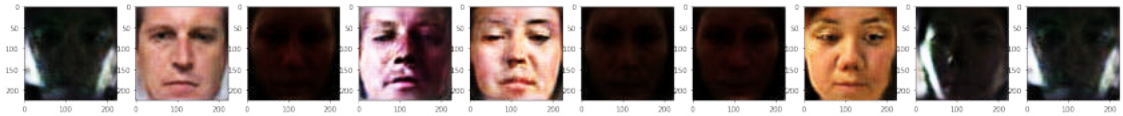
```
tensor([1, 1, 1, 3, 6, 1, 1, 1, 1, 1], device='cuda:0')
[2.600] Prior Loss: 145.44      Iden Loss: 7.78 Attack Acc: 0.20
```



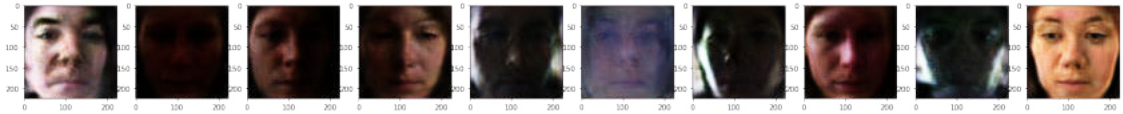
```
tensor([1, 1, 1, 3, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.900] Prior Loss: 148.75      Iden Loss: 7.23 Attack Acc: 0.20
```



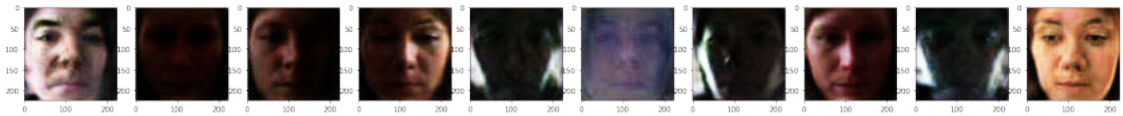
```
tensor([1, 1, 1, 3, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.1200]        Prior Loss: 148.51      Iden Loss: 7.22 Attack Acc: 0.20
```



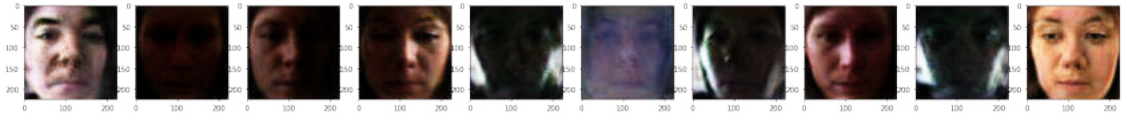
```
tensor([1, 1, 1, 3, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.1500]        Prior Loss: 148.55      Iden Loss: 7.22 Attack Acc: 0.20
[2]     Time:887.92     Acc:0.20
################################################################################
####################
```



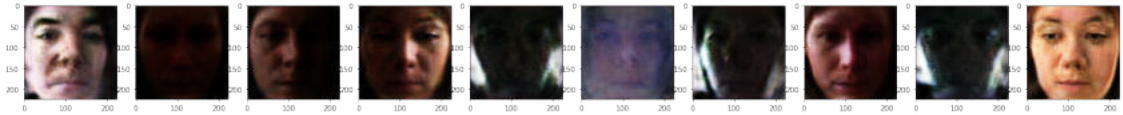
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.300] Prior Loss: 148.03      Iden Loss: 8.40 Attack Acc: 0.10
```



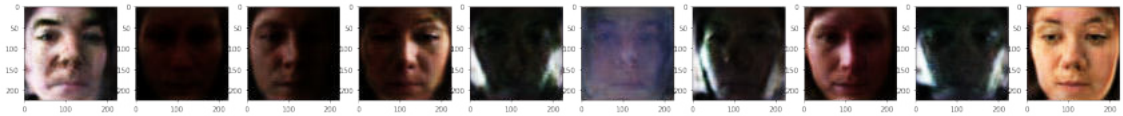
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 3, 1], device='cuda:0')
[3.600] Prior Loss: 141.08      Iden Loss: 8.30 Attack Acc: 0.10
```



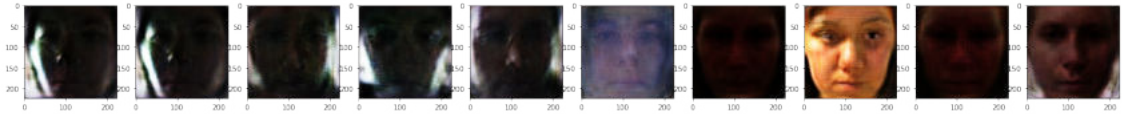
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 3, 1], device='cuda:0')
[3.900] Prior Loss: 140.13      Iden Loss: 8.27 Attack Acc: 0.10
```



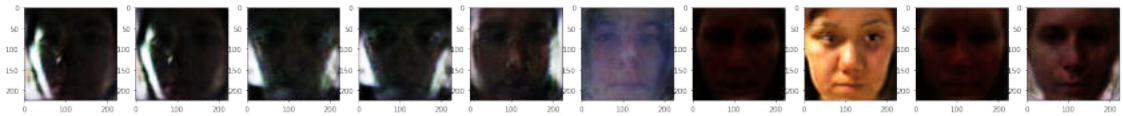
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 3, 1], device='cuda:0')
[3.1200]        Prior Loss: 139.94      Iden Loss: 8.26 Attack Acc: 0.10
```



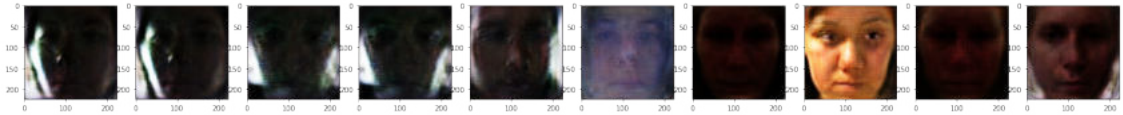
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 3, 1], device='cuda:0')
[3.1500]        Prior Loss: 139.78      Iden Loss: 8.26 Attack Acc: 0.10
[3]     Time:887.90     Acc:0.10
################################################################################
####################
```



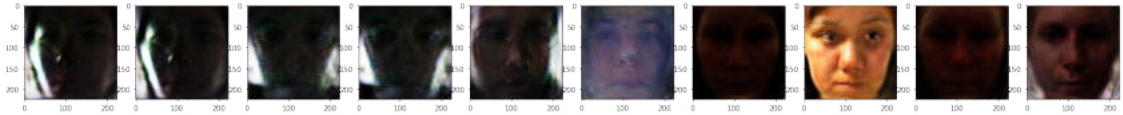
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[4.300] Prior Loss: 127.98      Iden Loss: 8.98 Attack Acc: 0.10
```



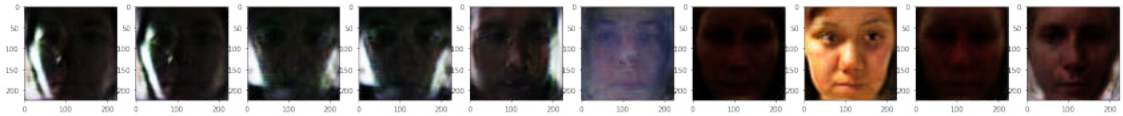
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[4.600] Prior Loss: 124.48      Iden Loss: 8.83 Attack Acc: 0.10
```



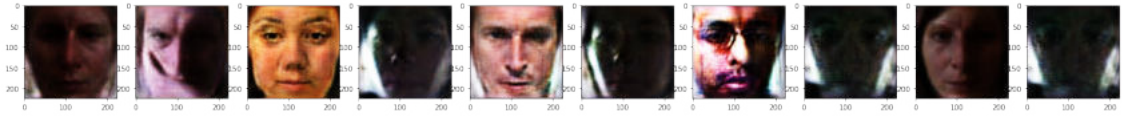
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[4.900] Prior Loss: 124.29      Iden Loss: 8.82 Attack Acc: 0.10
```



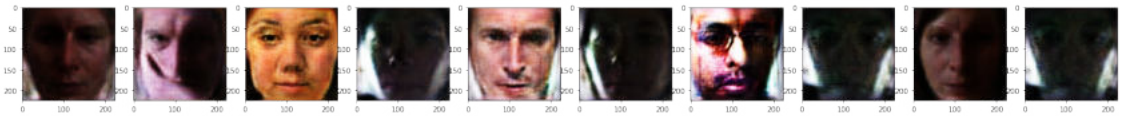
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[4.1200]        Prior Loss: 124.05      Iden Loss: 8.82 Attack Acc: 0.10
```



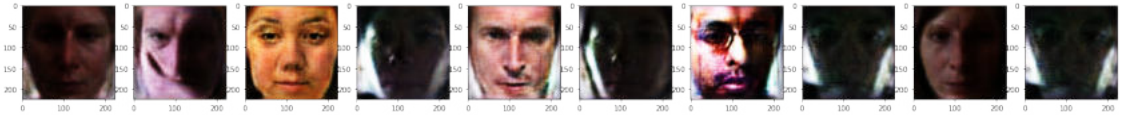
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[4.1500]        Prior Loss: 123.89      Iden Loss: 8.82 Attack Acc: 0.10
[4]     Time:888.07     Acc:0.10
################################################################################
####################
```



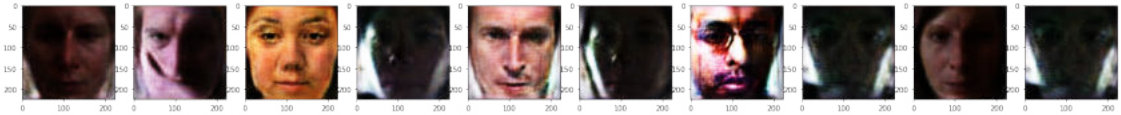
```
tensor([1, 1, 1, 3, 6, 1, 6, 1, 1, 1], device='cuda:0')
[5.300] Prior Loss: 121.47      Iden Loss: 7.14 Attack Acc: 0.30
```



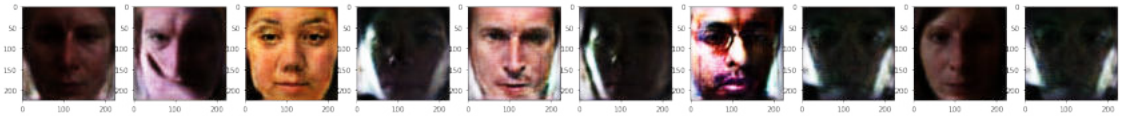
```
tensor([1, 1, 1, 3, 6, 1, 6, 1, 1, 1], device='cuda:0')
[5.600] Prior Loss: 121.01      Iden Loss: 7.13 Attack Acc: 0.30
```



```
tensor([1, 1, 1, 3, 6, 1, 6, 1, 1, 1], device='cuda:0')
[5.900] Prior Loss: 120.82      Iden Loss: 7.13 Attack Acc: 0.30
```



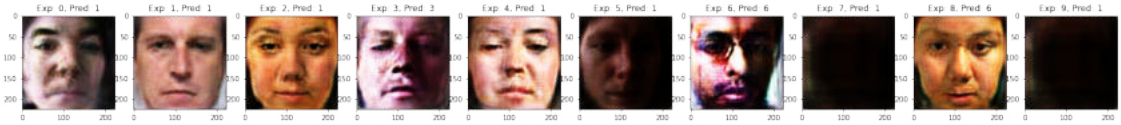
```
tensor([1, 1, 1, 3, 6, 1, 6, 1, 1, 1], device='cuda:0')
[5.1200]        Prior Loss: 120.80      Iden Loss: 7.13 Attack Acc: 0.30
```



```
tensor([1, 1, 1, 3, 6, 1, 6, 1, 1, 1], device='cuda:0')
[5.1500]        Prior Loss: 120.77      Iden Loss: 7.13 Attack Acc: 0.30
[5]     Time:890.64     Acc:0.30
################################################################################
####################
Attack Accuracy: 0.30
```

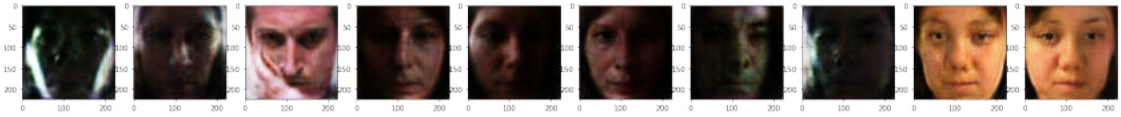
[22]:
```
f, axarr = plt.subplots(1, 10, figsize=(27, 48))
for _i in range(10):
    z = optim_z_s[_i]
    fake = G(z.unsqueeze(0))
    axarr[_i].imshow(fake.squeeze(0).permute(1, 2, 0).cpu().detach())
    pred = underfitT(fake)
    pred_id = torch.argmax(pred, dim=1)
```

```
    axarr[_i].title.set_text('Exp: {}, Pred: {}'.format(_i, pred_id.cpu()[0]))
plt.show()
```

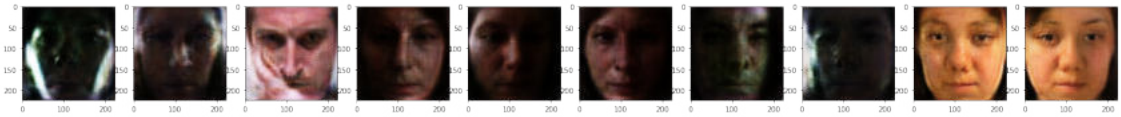


6.2.3 Custom Network

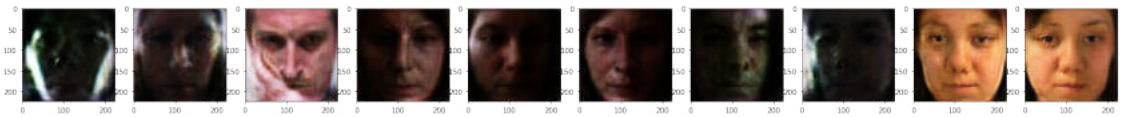
[17]:
```
optim_z_s = attack(
    target_net=cT,
    g_net=G,
    d_net=D,
    iden=identities
)
```



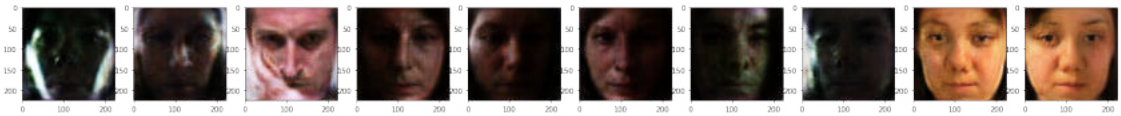
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 2, 9], device='cuda:0')
[1.300] Prior Loss: 156.97      Iden Loss: 0.91 Attack Acc: 0.90
```



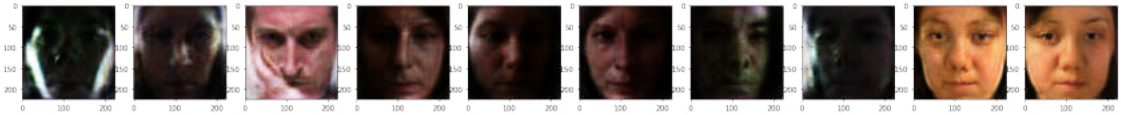
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 2, 9], device='cuda:0')
[1.600] Prior Loss: 154.50      Iden Loss: 0.89 Attack Acc: 0.90
```



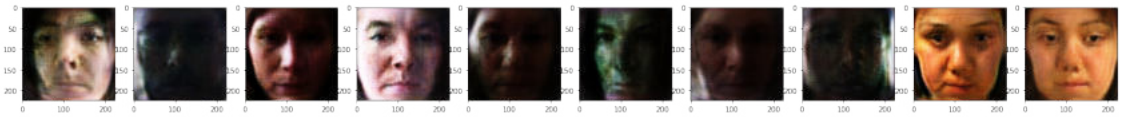
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 2, 9], device='cuda:0')
[1.900] Prior Loss: 154.53      Iden Loss: 0.89 Attack Acc: 0.90
```



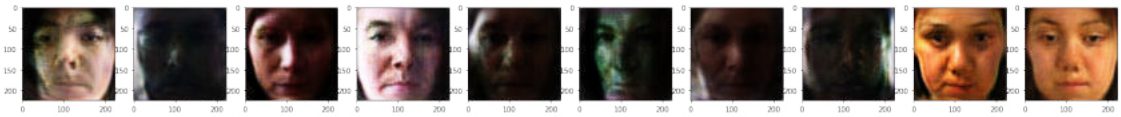
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 2, 9], device='cuda:0')
[1.1200]        Prior Loss: 154.85      Iden Loss: 0.88 Attack Acc: 0.90
```



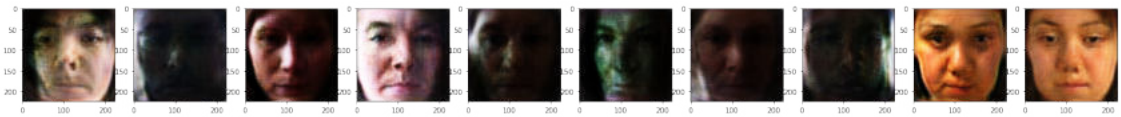
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 2, 9], device='cuda:0')
[1.1500]        Prior Loss: 154.96      Iden Loss: 0.88 Attack Acc: 0.90
[1]     Time:645.97     Acc:0.90
################################################################################
####################
```



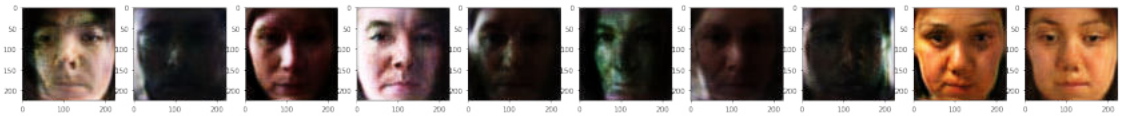
```
tensor([0, 5, 2, 5, 4, 5, 6, 7, 5, 9], device='cuda:0')
[2.300] Prior Loss: 159.23      Iden Loss: 0.96 Attack Acc: 0.70
```



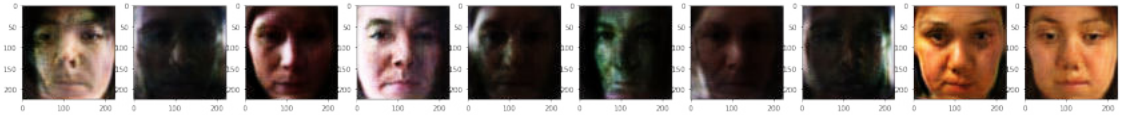
```
tensor([0, 5, 2, 5, 4, 5, 6, 7, 5, 9], device='cuda:0')
[2.600] Prior Loss: 156.06      Iden Loss: 0.95 Attack Acc: 0.70
```



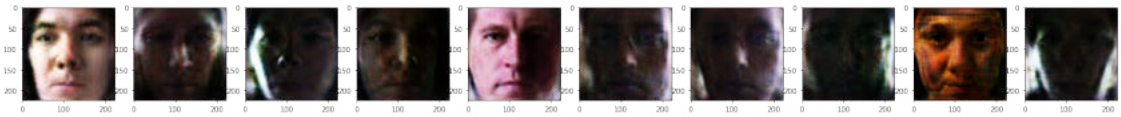
```
tensor([0, 5, 2, 5, 4, 5, 6, 7, 5, 9], device='cuda:0')
[2.900] Prior Loss: 155.41      Iden Loss: 0.95 Attack Acc: 0.70
```



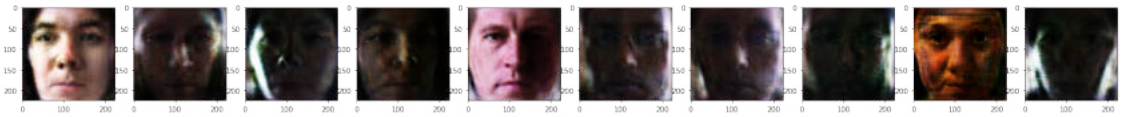
```
tensor([0, 5, 2, 5, 4, 5, 6, 7, 5, 9], device='cuda:0')
[2.1200]        Prior Loss: 155.26      Iden Loss: 0.94 Attack Acc: 0.70
```



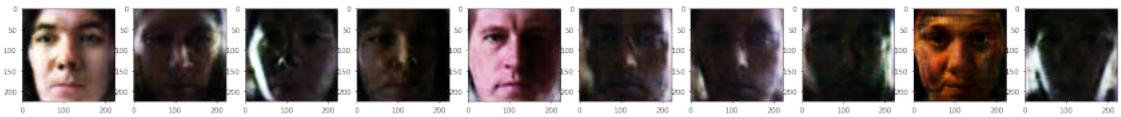
```
tensor([0, 1, 2, 5, 4, 5, 6, 7, 5, 9], device='cuda:0')
[2.1500]        Prior Loss: 152.83      Iden Loss: 0.90 Attack Acc: 0.80
[2]     Time:649.91     Acc:0.80
################################################################################
####################
```



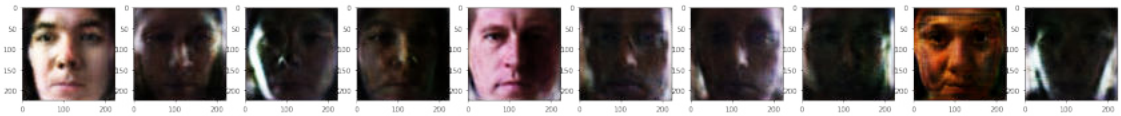
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], device='cuda:0')
[3.300] Prior Loss: 133.73      Iden Loss: 0.67 Attack Acc: 1.00
```



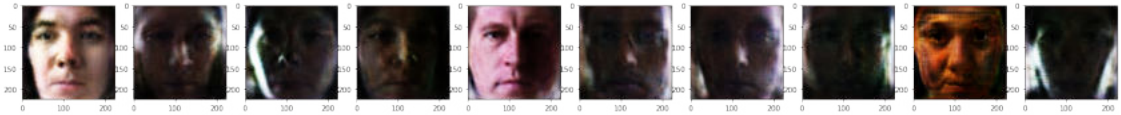
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], device='cuda:0')
[3.600] Prior Loss: 132.61      Iden Loss: 0.67 Attack Acc: 1.00
```



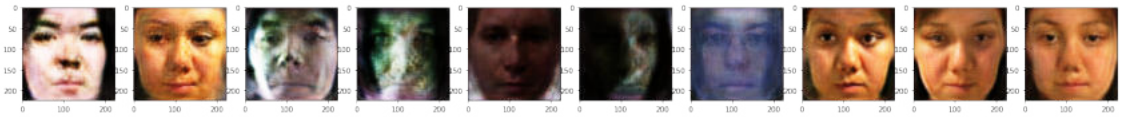
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], device='cuda:0')
[3.900] Prior Loss: 132.00      Iden Loss: 0.66 Attack Acc: 1.00
```



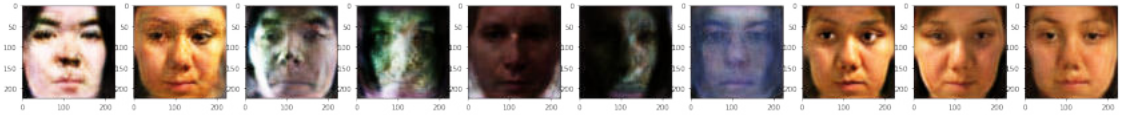
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], device='cuda:0')
[3.1200]        Prior Loss: 131.75      Iden Loss: 0.66 Attack Acc: 1.00
```



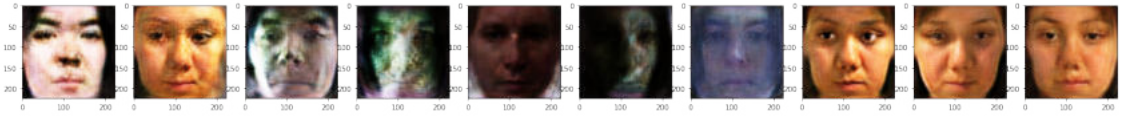
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9], device='cuda:0')
[3.1500]        Prior Loss: 131.77      Iden Loss: 0.66 Attack Acc: 1.00
[3]     Time:649.46     Acc:1.00
################################################################################
####################
```



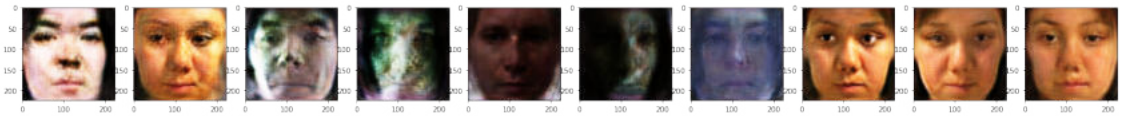
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 5, 9], device='cuda:0')
[4.300] Prior Loss: 157.28      Iden Loss: 0.78 Attack Acc: 0.90
```



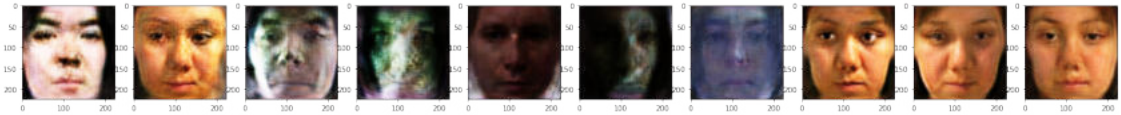
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 5, 9], device='cuda:0')
[4.600] Prior Loss: 155.74      Iden Loss: 0.76 Attack Acc: 0.90
```



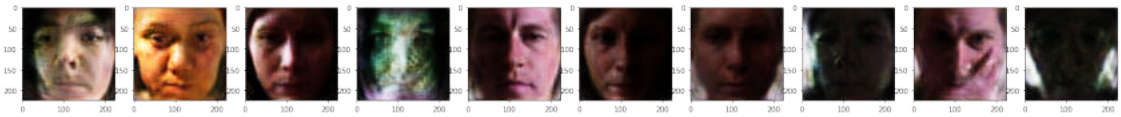
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 5, 9], device='cuda:0')
[4.900] Prior Loss: 153.28      Iden Loss: 0.76 Attack Acc: 0.90
```



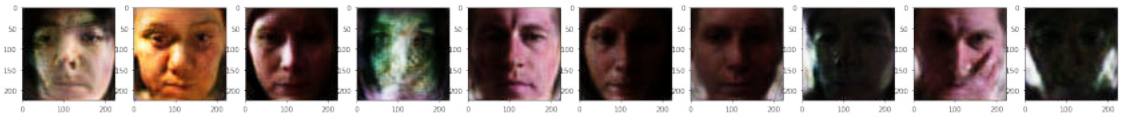
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 5, 9], device='cuda:0')
[4.1200]        Prior Loss: 153.13      Iden Loss: 0.75 Attack Acc: 0.90
```



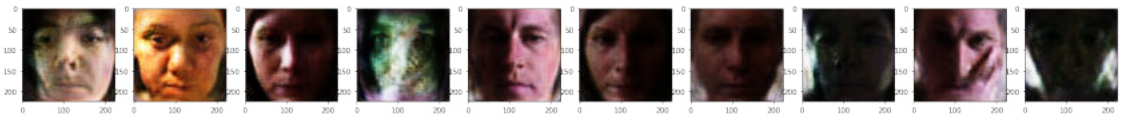
```
tensor([0, 1, 2, 3, 4, 5, 6, 7, 5, 9], device='cuda:0')
[4.1500]        Prior Loss: 153.02      Iden Loss: 0.75 Attack Acc: 0.90
[4]     Time:650.13     Acc:0.90
################################################################################
####################
```



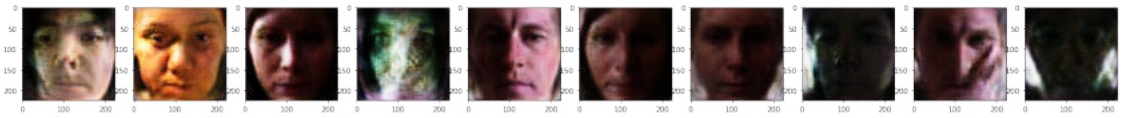
```
tensor([0, 1, 2, 3, 4, 5, 6, 2, 5, 9], device='cuda:0')
[5.300] Prior Loss: 134.23      Iden Loss: 0.83 Attack Acc: 0.80
```



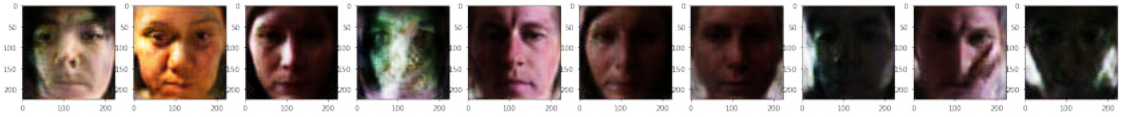
```
tensor([0, 1, 2, 3, 4, 5, 6, 2, 5, 9], device='cuda:0')
[5.600] Prior Loss: 130.56      Iden Loss: 0.82 Attack Acc: 0.80
```



```
tensor([0, 1, 2, 3, 4, 5, 6, 2, 5, 9], device='cuda:0')
[5.900] Prior Loss: 129.68      Iden Loss: 0.76 Attack Acc: 0.80
```

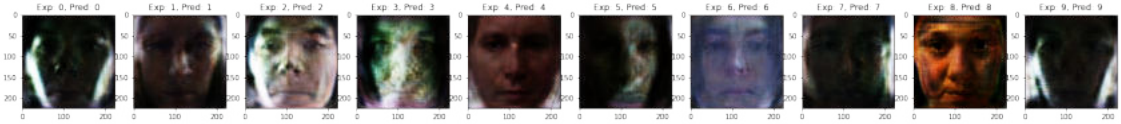


```
tensor([0, 1, 2, 3, 4, 5, 6, 2, 5, 9], device='cuda:0')
[5.1200]        Prior Loss: 126.25      Iden Loss: 0.77 Attack Acc: 0.80
```



```
tensor([0, 1, 2, 3, 4, 5, 6, 2, 5, 9], device='cuda:0')
[5.1500]        Prior Loss: 125.83      Iden Loss: 0.77 Attack Acc: 0.80
[5]     Time:649.67     Acc:0.80
################################################################################
####################
Attack Accuracy: 1.00
```

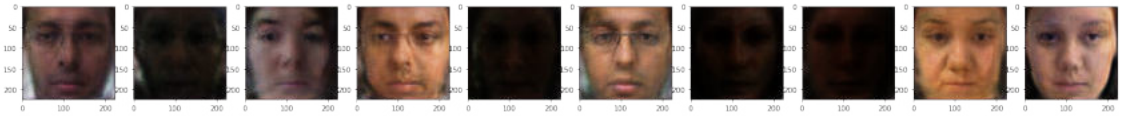
[28]:
```
f, axarr = plt.subplots(1, 10, figsize=(27, 48))
for _i in range(10):
    z = optim_z_s[_i]
    fake = G(z.unsqueeze(0))
    axarr[_i].imshow(fake.squeeze(0).permute(1, 2, 0).cpu().detach())
    pred = cT(fake)
    pred_id = torch.argmax(pred, dim=1)
    axarr[_i].title.set_text('Exp: {}, Pred: {}'.format(_i, pred_id.cpu()[0]))
plt.show()
```



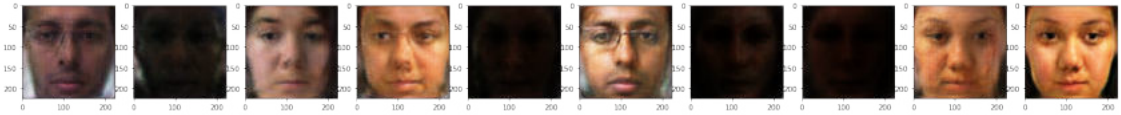
6.2.4 Eval Classifier (ResNet152)

[17]:
```
optim_z_s = attack(
    target_net=E,
    g_net=G,
    d_net=D,
```

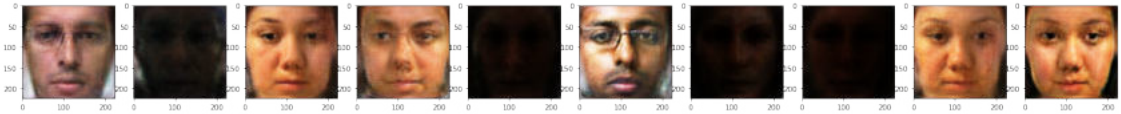
```
    iden=identities,
    verbose=True
)
```



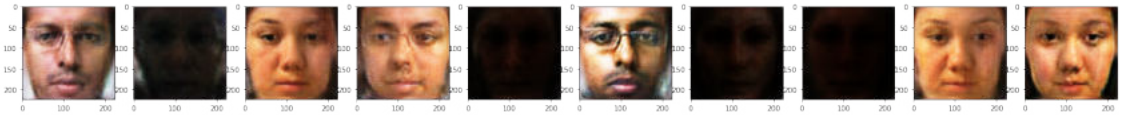
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[1.1]   Prior Loss: 241.57      Iden Loss: 9.83 Attack Acc: 0.10
```



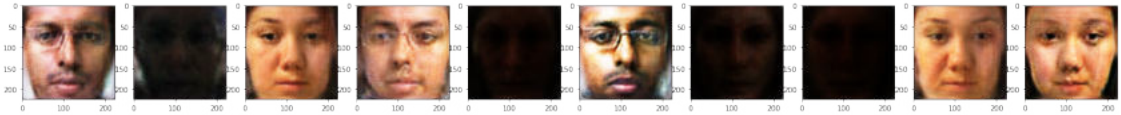
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[1.2]   Prior Loss: 240.95      Iden Loss: 11.03        Attack Acc: 0.10
```



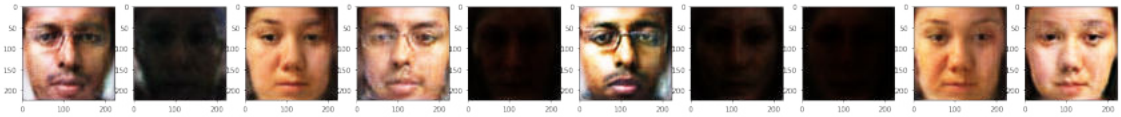
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[1.3]   Prior Loss: 234.06      Iden Loss: 10.07        Attack Acc: 0.10
```



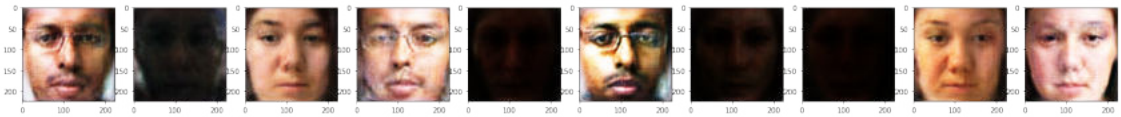
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[1.4]   Prior Loss: 237.32      Iden Loss: 9.35 Attack Acc: 0.10
```



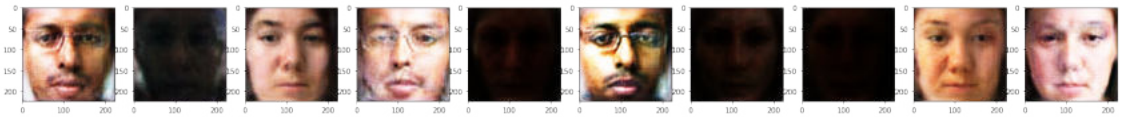
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[1.5]   Prior Loss: 242.03      Iden Loss: 8.98 Attack Acc: 0.10
```



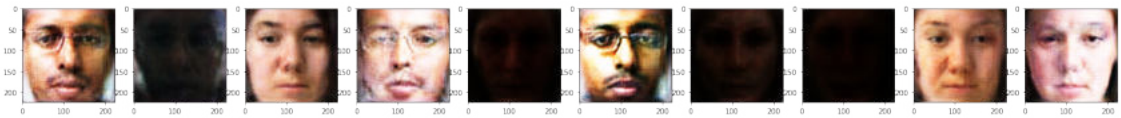
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[1.6]   Prior Loss: 244.42      Iden Loss: 8.65 Attack Acc: 0.10
```



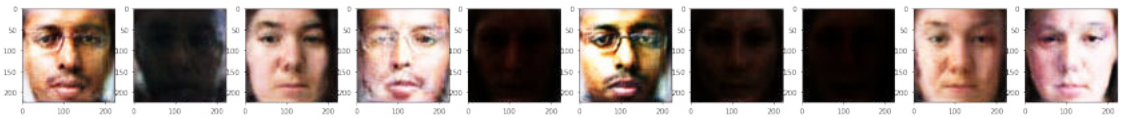
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 6], device='cuda:0')
[1.7]   Prior Loss: 235.46      Iden Loss: 8.37 Attack Acc: 0.10
```



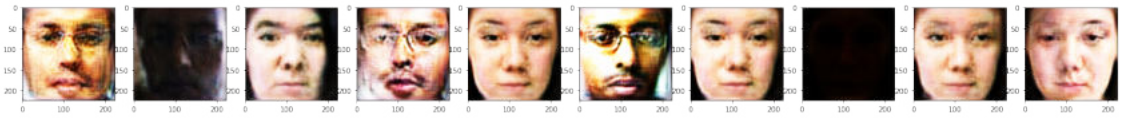
```
tensor([1, 1, 1, 6, 1, 1, 1, 1, 1, 6], device='cuda:0')
[1.8]   Prior Loss: 235.39      Iden Loss: 8.10 Attack Acc: 0.10
```



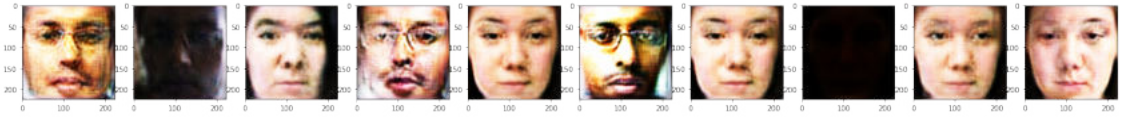
```
tensor([1, 1, 1, 6, 1, 1, 1, 1, 1, 6], device='cuda:0')
[1.9]   Prior Loss: 235.14      Iden Loss: 7.94 Attack Acc: 0.10
```



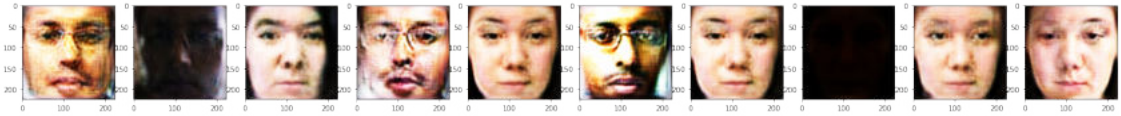
```
tensor([1, 1, 1, 6, 1, 1, 1, 1, 1, 6], device='cuda:0')
[1.10]  Prior Loss: 234.22      Iden Loss: 7.81 Attack Acc: 0.10
```



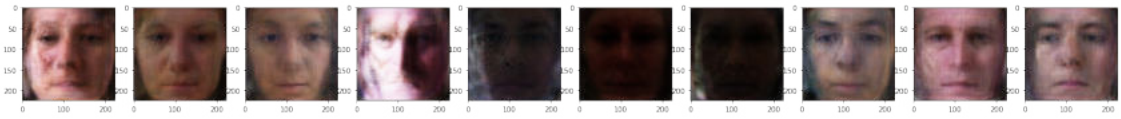
```
tensor([6, 1, 6, 6, 6, 6, 6, 1, 6, 6], device='cuda:0')
[1.300] Prior Loss: 143.43      Iden Loss: 5.06 Attack Acc: 0.20
```



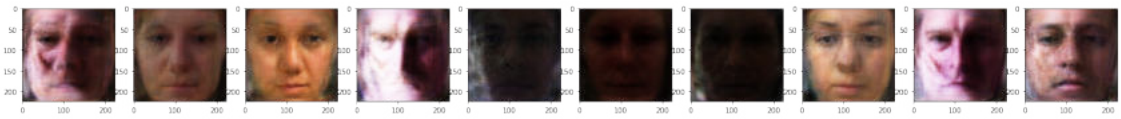
```
tensor([6, 1, 6, 6, 6, 6, 6, 1, 6, 6], device='cuda:0')
[1.600] Prior Loss: 142.09      Iden Loss: 5.04 Attack Acc: 0.20
```



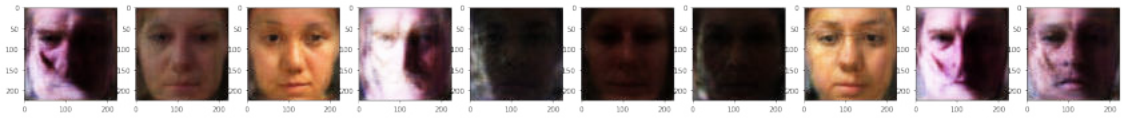
```
tensor([6, 1, 6, 6, 6, 6, 6, 1, 6, 6], device='cuda:0')
[1.900] Prior Loss: 142.08      Iden Loss: 5.04 Attack Acc: 0.20
[1]     Time:678.42     Acc:0.20
################################################################################
####################
```



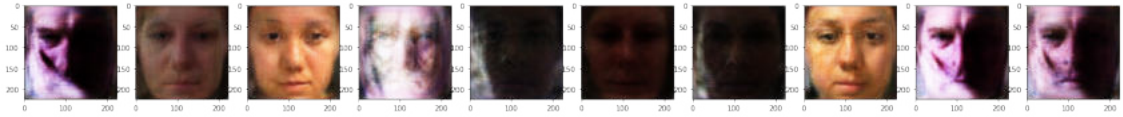
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.1]   Prior Loss: 236.13      Iden Loss: 10.38        Attack Acc: 0.10
```



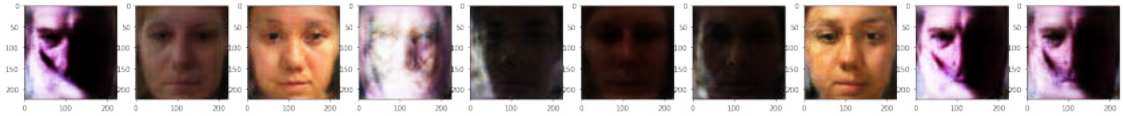
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.2]   Prior Loss: 238.75      Iden Loss: 10.05        Attack Acc: 0.10
```



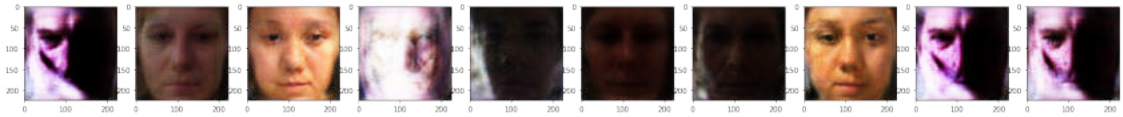
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.3]   Prior Loss: 230.77      Iden Loss: 9.61 Attack Acc: 0.10
```



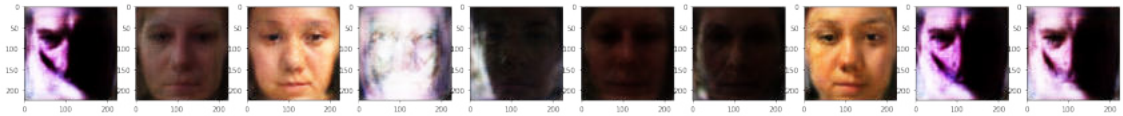
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.4]   Prior Loss: 225.42      Iden Loss: 9.28 Attack Acc: 0.10
```



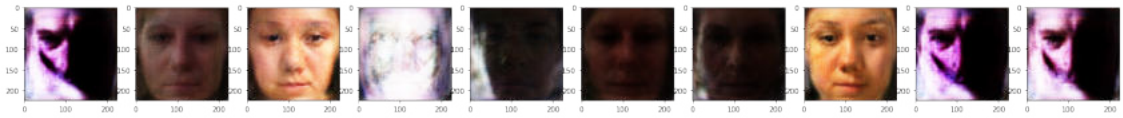
```
tensor([1, 1, 1, 6, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.5]   Prior Loss: 227.50      Iden Loss: 8.83 Attack Acc: 0.10
```



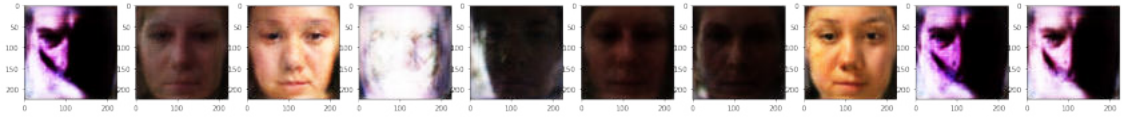
```
tensor([1, 1, 1, 6, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.6]   Prior Loss: 223.23      Iden Loss: 8.49 Attack Acc: 0.10
```



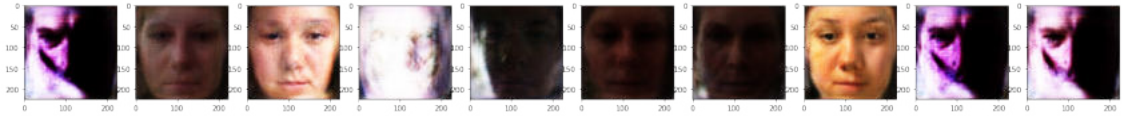
```
tensor([1, 1, 1, 6, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.7]   Prior Loss: 214.22      Iden Loss: 8.32 Attack Acc: 0.10
```



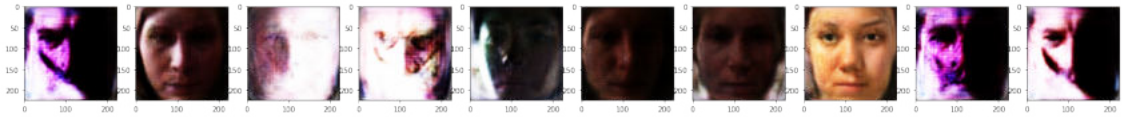
```
tensor([1, 1, 1, 6, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.8]   Prior Loss: 218.18      Iden Loss: 8.12 Attack Acc: 0.10
```



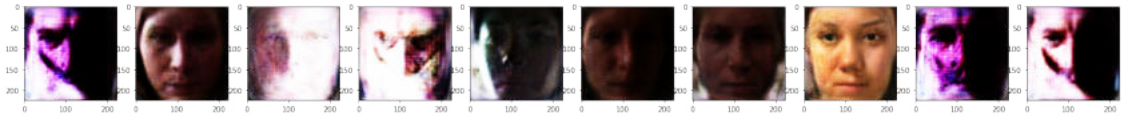
```
tensor([1, 1, 1, 6, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.9]   Prior Loss: 216.31      Iden Loss: 7.99 Attack Acc: 0.10
```



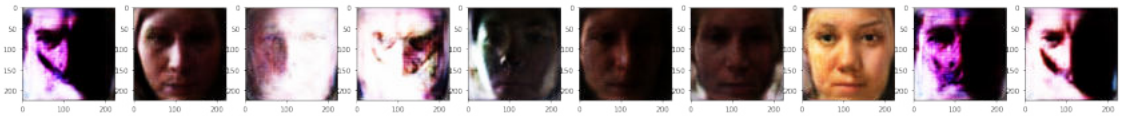
```
tensor([1, 1, 1, 6, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.10]  Prior Loss: 210.38      Iden Loss: 7.90 Attack Acc: 0.10
```



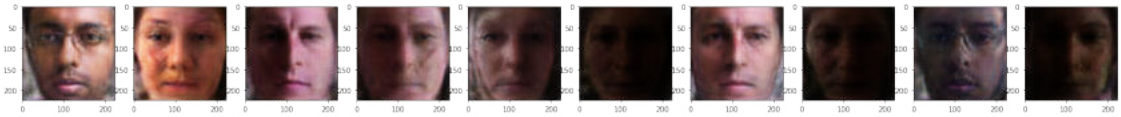
```
tensor([1, 1, 2, 6, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.300] Prior Loss: 149.76      Iden Loss: 6.73 Attack Acc: 0.20
```



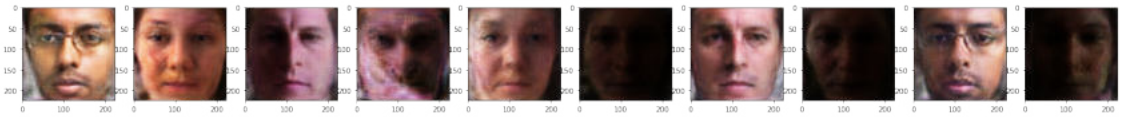
```
tensor([1, 1, 2, 6, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.600] Prior Loss: 150.92      Iden Loss: 6.72 Attack Acc: 0.20
```



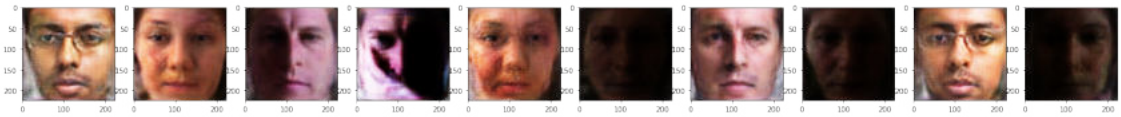
```
tensor([1, 1, 2, 6, 1, 1, 1, 1, 1, 1], device='cuda:0')
[2.900] Prior Loss: 148.86      Iden Loss: 6.67 Attack Acc: 0.20
[2]     Time:676.76     Acc:0.20
################################################################################
####################
```



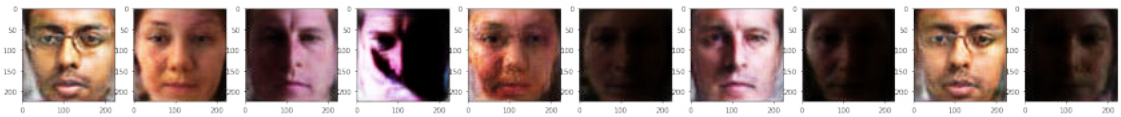
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.1]   Prior Loss: 244.46      Iden Loss: 11.08        Attack Acc: 0.10
```



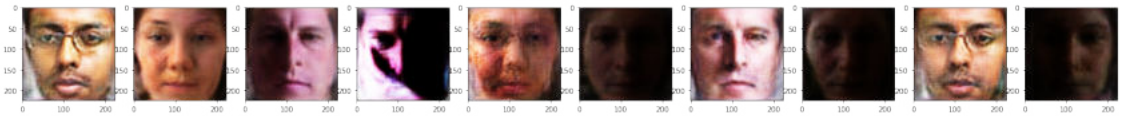
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.2]   Prior Loss: 240.03      Iden Loss: 11.17        Attack Acc: 0.10
```



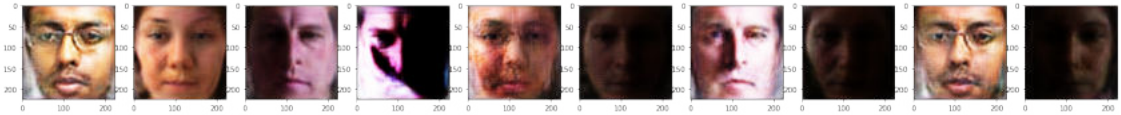
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.3]   Prior Loss: 236.97      Iden Loss: 10.11        Attack Acc: 0.10
```



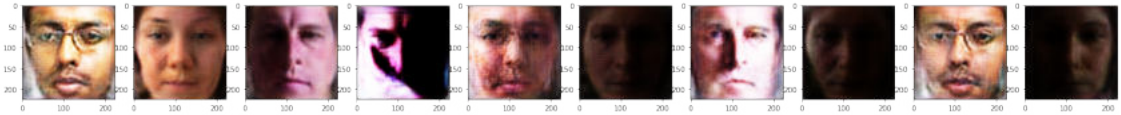
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.4]   Prior Loss: 235.07      Iden Loss: 9.31 Attack Acc: 0.10
```



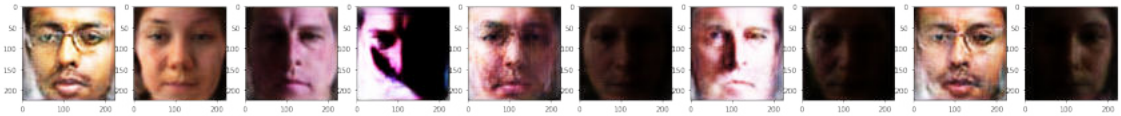
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.5]   Prior Loss: 232.90      Iden Loss: 9.04 Attack Acc: 0.10
```



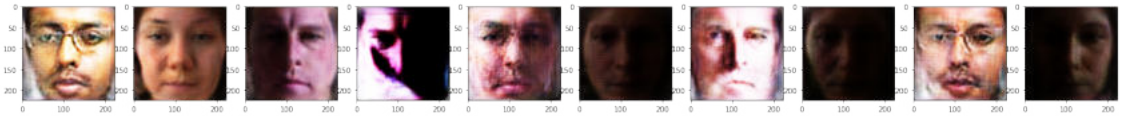
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.6]   Prior Loss: 233.94      Iden Loss: 8.82 Attack Acc: 0.10
```



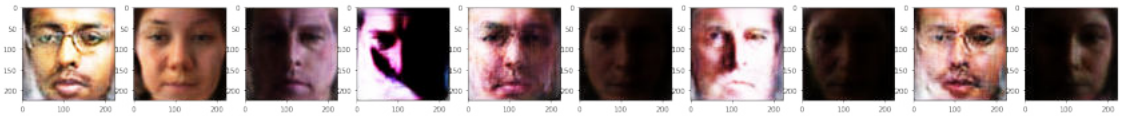
```
tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.7]   Prior Loss: 227.43      Iden Loss: 8.60 Attack Acc: 0.10
```



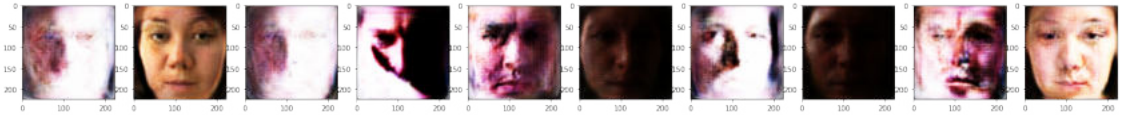
```
tensor([6, 1, 1, 1, 1, 1, 1, 1, 1, 1], device='cuda:0')
[3.8]   Prior Loss: 223.89      Iden Loss: 8.43 Attack Acc: 0.10
```



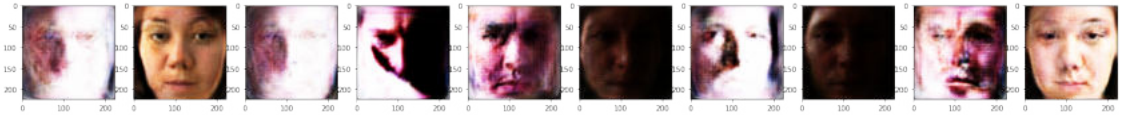
```
tensor([6, 1, 1, 1, 1, 1, 1, 1, 6, 1], device='cuda:0')
[3.9]   Prior Loss: 219.91      Iden Loss: 8.29 Attack Acc: 0.10
```



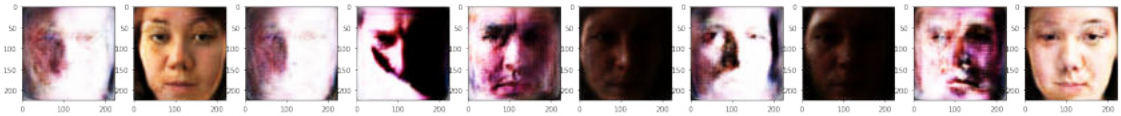
```
tensor([6, 1, 1, 1, 1, 1, 1, 1, 6, 1], device='cuda:0')
[3.10]  Prior Loss: 216.61      Iden Loss: 8.21 Attack Acc: 0.10
```



```
tensor([1, 1, 2, 1, 1, 1, 6, 1, 6, 6], device='cuda:0')
[3.300] Prior Loss: 139.67      Iden Loss: 5.99 Attack Acc: 0.30
```



```
tensor([1, 1, 2, 1, 1, 1, 6, 1, 6, 6], device='cuda:0')
[3.600] Prior Loss: 136.84      Iden Loss: 5.96 Attack Acc: 0.30
```



```
tensor([1, 1, 2, 1, 1, 1, 6, 1, 6, 6], device='cuda:0')
[3.900] Prior Loss: 137.95      Iden Loss: 5.92 Attack Acc: 0.30
[3]     Time:676.80     Acc:0.30
################################################################################
####################
Attack Accuracy: 0.40
```

[19]:
```
f, axarr = plt.subplots(1, 10, figsize=(27, 48))
for _i in range(10):
    z = optim_z_s[_i]
    fake = G(z.unsqueeze(0))
    axarr[_i].imshow(fake.squeeze(0).permute(1, 2, 0).cpu().detach())
    pred = E(fake)
    pred_id = torch.argmax(pred, dim=1)
```

```
    axarr[_i].title.set_text('Exp: {}, Pred: {}'.format(_i, pred_id.cpu()[0]))
plt.show()
```