Deep Learning School

Курс "Глубокое обучение". Первый семестр

Домашнее задание. CycleGAN Pytorch

Выполнил: Дмитрий Шерешевский, ID 36196483 (Dmitry Shereshevskiy)

Введение

Архитектуры подробно изложены в https://arxiv.org/pdf/1703.10593.pdf (1) и https://arxiv.org/pdf/1603.08155.pdf (2). Реализованная здесь архитектура полностью соответствует описаниям в статьях.
Для дискриминатора использована упомянутая в статьях архтектура Markovian discriminator (PatchGAN). Его идею можно посмотреть, в частности, здесь.
Было изучено несколько реализаций CycleGAN, в том числе и на соответствие принципам, изложенным в исходниках (1) и (2). Некоторые идеи для реализации отдельных частей архитектур, которые показались удачными и соответствующими исходным статьям, взяты отсюда https://github.com/aitorzip/PyTorch-CycleGAN, дабы не изобретать велосипед. Код адаптирован, значительно переработан и дополнен.

In [ ]:
```
from google.colab import drive
drive.mount('/content/gdrive/')
```

```
Drive already mounted at /content/gdrive/; to attempt to forcibly remount, call drive.mo
unt("/content/gdrive/", force_remount=True).
```

In [ ]:
```
import os
os.chdir("gdrive/MyDrive/Colab Notebooks/DLSchool_mipt_1sem/cycleGAN/")
```

Для начала решим вопрос с датасетом.

Для отладки в этой работы использовались датасеты из репозитория **UC Berkeley**:
https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets/

Имена датасетов в репозитории:

apple2orange, summer2winter_yosemite, horse2zebra, monet2photo, cezanne2photo, ukiyoe2photo, vangogh2photo, maps, cityscapes, facades, iphone2dslr_flower, ae_photos

download dataset

In [ ]:
```
dataset_name = "horse2zebra"
```

Для скачивания датасета необходимо раскоментить следующую ячейку при первом прохождении ноутбука.
В настоящий момент ячейка закоментина, чтобы не скачивать при отладке одни и те же файлы.

In [ ]:
```
# URL = f"https://people.eecs.berkeley.edu/~taesung_park/CycleGAN/datasets/{dataset_nam
# if not os.path.exists('datasets'):
#     os.mkdir('datasets')
# ZIP_FILE = f"./datasets/{dataset_name}.zip"
# TARGET_DIR = f"./datasets/{dataset_name}"
# ! mkdir -p ./datasets
# ! wget -N $URL -O $ZIP_FILE
# ! unzip $ZIP_FILE -d ./datasets/
# ! rm $ZIP_FILE
```

Теперь сформируам наш **уникальный** датасет.

Попросту скачаем вручную картинки черных и белых кошек и разложим по соответствующим папкам.

In [ ]:
```
dataset_name = "whitecat2blackcat"
```

In [ ]:
```
import glob
import random
import os

from torch.utils.data import Dataset, DataLoader, RandomSampler
from PIL import Image
import torchvision.transforms as transforms
import torch
import sys
import numpy as np
from torchvision.utils import save_image

import matplotlib.pyplot as plt
```

```
import itertools
import json
```

класс для Dataset

In [ ]:
```
class ImageDataset(Dataset):
    def __init__(self, root, transforms_=None, unaligned=False, mode='train'):
        self.transform = transforms.Compose(transforms_)
        self.unaligned = unaligned
        self.mode = mode

        self.files_A = sorted(glob.glob(os.path.join(root, f'{mode}A', '*.*')))
        self.files_B = sorted(glob.glob(os.path.join(root, f'{mode}B', '*.*')))

    def to3channs(self, image):
        num_channels = len(image.split())
        if num_channels == 1:
            image = Image.merge("RGB", [image] * 3)
        return image

    def __getitem__(self, index):
        image = self.to3channs(Image.open(self.files_A[index % len(self.files_A)]).conv
        item_A = self.transform(image)

        if self.mode == "train":
            if self.unaligned:
                image = self.to3channs(Image.open(self.files_B[random.randint(0, len(se
                item_B = self.transform(image)
            else:
                image = self.to3channs(Image.open(self.files_B[index % len(self.files_B
                item_B = self.transform(image)

        elif self.mode == "test":
            image = self.to3channs(Image.open(self.files_B[index % len(self.files_B)]).
            item_B = self.transform(image)

        else:
            raise ValueError("The parameter 'mode' should only be 'train' or 'test'.")

        return {'A': item_A, 'B': item_B}

    def __len__(self):
        return max(len(self.files_A), len(self.files_B))
```

посмотрим на некоторые изображения тестового датасета

In [ ]:
```
size = 256
test_transforms = [transforms.Resize(int(size),  Image.BICUBIC),
                   transforms.CenterCrop(size),
                   transforms.ToTensor(),
                   transforms.Normalize((0.5,0.5,0.5), (0.5,0.5,0.5))]
test_dataset = ImageDataset(f"datasets/{dataset_name}", test_transforms, mode="test")
```

In [ ]:
```
def plot_dataset(dataset, inds=tuple(range(6))):
    plt.figure(figsize=(18, 6))
    for i in inds:
        plt.subplot(2, len(inds), i+1)
        plt.axis("off")
```

```
        plt.imshow(0.5 * (test_dataset[i]["A"].numpy().transpose((1, 2, 0)) + 1))
        plt.title("real_A")

        plt.subplot(2, len(inds), i+7)
        plt.axis("off")
        plt.imshow(0.5 * (test_dataset[i]["B"].numpy().transpose((1, 2, 0)) + 1))
        plt.title("real_B")
    plt.show()
```

In [ ]:
```
plot_dataset(test_dataset)
```

Модели

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

class ResidualBlock(nn.Module):
    def __init__(self, in_features):
        super().__init__()

        conv_block = [nn.ReflectionPad2d(1),
                      nn.Conv2d(in_features, in_features, 3),
                      nn.InstanceNorm2d(in_features),
                      nn.ReLU(inplace=True),
                      nn.ReflectionPad2d(1),
                      nn.Conv2d(in_features, in_features, 3),
                      nn.InstanceNorm2d(in_features)]

        self.conv_block = nn.Sequential(*conv_block)

    def forward(self, x):
        return x + self.conv_block(x)

class Generator(nn.Module):
    def __init__(self, input_nc, output_nc, n_residual_blocks=9):
        super().__init__()

        # Initial convolution block
        model = [nn.ReflectionPad2d(3),
                 nn.Conv2d(input_nc, 64, 7),
                 nn.InstanceNorm2d(64),
                 nn.ReLU(inplace=True)]

        # Downsampling
        in_features = 64
```

```
        out_features = in_features*2
        for _ in range(2):
            model += [nn.Conv2d(in_features, out_features, 3, stride=2, padding=1),
                      nn.InstanceNorm2d(out_features),
                      nn.ReLU(inplace=True)]

            in_features = out_features
            out_features = in_features*2

        # Residual blocks
        for _ in range(n_residual_blocks):
            model += [ResidualBlock(in_features)]

        # Upsampling
        out_features = in_features // 2
        for _ in range(2):
            model += [nn.ConvTranspose2d(in_features, out_features, 3, stride=2, paddin
                      nn.InstanceNorm2d(out_features),
                      nn.ReLU(inplace=True)]
            in_features = out_features
            out_features = in_features // 2

        # Output layer
        model += [nn.ReflectionPad2d(3),
                  nn.Conv2d(64, output_nc, 7),
                  nn.Tanh()]

        self.model = nn.Sequential(*model)

    def forward(self, x):
        return self.model(x)


# for discriminator was used Markovian discriminator (PatchGAN) with the average poolin
class PatchlBlock(nn.Module):
    def __init__(self, in_features, out_features):
        super().__init__()

        conv_block = [nn.Conv2d(in_features, out_features, 4, stride=2, padding=1),
                      nn.InstanceNorm2d(out_features),
                      nn.LeakyReLU(0.2, inplace=True)]

        self.conv_block = nn.Sequential(*conv_block)

    def forward(self, x):
        return self.conv_block(x)

class Discriminator(nn.Module):
    def __init__(self, input_nc):
        super().__init__()
        model = []
        model += [PatchlBlock(input_nc, 64),
                  PatchlBlock(64, 128),
                  PatchlBlock(128, 256),
                  PatchlBlock(256, 512)]
        # to one channel for the next average pooling
        model += [nn.Conv2d(512, 1, 4, padding=1)]

        self.model = nn.Sequential(*model)

    def forward(self, x):
```

```
        x =  self.model(x)
        # Average pooling and flatten
        return F.avg_pool2d(x, x.shape[2:]).view(x.shape[0], -1)
```

In [ ]:
```
class ImageBuffer:
    def __init__(self, max_size=50):
        if max_size < 0:
            print('Warning: Empty buffer or trying to create a black hole. Be careful.'
        self.max_size = max_size
        self.data = []

    def update(self, data):
        to_return = []
        for element in data.data:
            element = torch.unsqueeze(element, 0)
            if len(self.data) < self.max_size:
                self.data.append(element)
                to_return.append(element)
            else:
                if random.uniform(0,1) > 0.5:
                    i = random.randint(0, self.max_size-1)
                    to_return.append(self.data[i].clone())
                    self.data[i] = element
                else:
                    to_return.append(element)
        return torch.cat(to_return)

class LambdaLR:
    def __init__(self, n_epochs, offset=0, decay_start_epoch=100):
        if (n_epochs - decay_start_epoch) < 0:
            print("Warning: Decay must start before the training session ends!")
        self.n_epochs = n_epochs
        self.offset = offset
        self.decay_start_epoch = decay_start_epoch

    def step(self, epoch):
        return 1.0 - max(0, epoch + self.offset - self.decay_start_epoch) / (self.n_epo

def weights_init_normal(m):
    classname = m.__class__.__name__
    if classname.find('Conv') != -1:
        torch.nn.init.normal_(m.weight.data, 0.0, 0.02)
    elif classname.find('BatchNorm2d') != -1:
        torch.nn.init.normal_(m.weight.data, 1.0, 0.02)
        torch.nn.init.constant(m.bias.data, 0.0)
```

функция для отрисовки тестовых картинок при обучении

In [ ]:
```
def plot_images(images: dict):
    img_len = len(images)
    fig, axes = plt.subplots(ncols=img_len, figsize=(9, img_len))
    for i, img_name in enumerate(images):
        ax = axes[i]
        ax.axis("off")
        img = 0.5 * (images[img_name].cpu().detach().numpy().transpose((1, 2, 0)) + 1)
        ax.imshow(img)
        ax.set_title(img_name)
    plt.show()
```

Функции **train** и **test** - обучение и тестирование во время обучения

In [ ]:

```
def train(netG_A2B, netG_B2A, netD_A, netD_B, dataloader, epoch, plot_images_=False, pl
    mode = "TRAIN"
    epoch_log = {}
    lambda_ = 10
    for i, batch in enumerate(dataloader):
        # Set model input
        real_A = batch['A'].to(device)
        real_B = batch['B'].to(device)

        ###### Generators A2B and B2A ######
        optimizer_G.zero_grad()

        # Identity loss
        # G_A2B(B) should equal B if real B is fed
        same_B = netG_A2B(real_B)
        loss_identity_B = criterion_identity(same_B, real_B) * 0.5*lambda_
        # G_B2A(A) should equal A if real A is fed
        same_A = netG_B2A(real_A)
        loss_identity_A = criterion_identity(same_A, real_A) * 0.5*lambda_

        # GAN loss
        fake_B = netG_A2B(real_A)
        pred_fake = netD_B(fake_B)
        loss_GAN_A2B = criterion_GAN(pred_fake, target_real)

        fake_A = netG_B2A(real_B)
        pred_fake = netD_A(fake_A)
        loss_GAN_B2A = criterion_GAN(pred_fake, target_real)

        # Cycle loss
        recovered_A = netG_B2A(fake_B)
        loss_cycle_ABA = criterion_cycle(recovered_A, real_A) * lambda_

        recovered_B = netG_A2B(fake_A)
        loss_cycle_BAB = criterion_cycle(recovered_B, real_B) * lambda_

        # Total loss
        loss_G =  (loss_identity_A + loss_identity_B) + (loss_GAN_A2B + loss_GAN_B2A) +
        loss_G.backward()

        optimizer_G.step()
        ###################################

        ###### Discriminator A ######
        optimizer_D_A.zero_grad()

        # Real loss
        pred_real = netD_A(real_A)
        loss_D_real = criterion_GAN(pred_real, target_real)

        # Fake loss
        fake_A = fake_A_buffer.update(fake_A)
        pred_fake = netD_A(fake_A.detach())
        loss_D_fake = criterion_GAN(pred_fake, target_fake)

        # Total loss
```

```
        loss_D_A = (loss_D_real + loss_D_fake) / 2
        loss_D_A.backward()

        optimizer_D_A.step()
        ###################################

        ###### Discriminator B ######
        optimizer_D_B.zero_grad()

        # Real loss
        pred_real = netD_B(real_B)
        loss_D_real = criterion_GAN(pred_real, target_real)

        # Fake loss
        fake_B = fake_B_buffer.update(fake_B)
        pred_fake = netD_B(fake_B.detach())
        loss_D_fake = criterion_GAN(pred_fake, target_fake)

        # Total loss
        loss_D_B = (loss_D_real + loss_D_fake) / 2
        loss_D_B.backward()

        optimizer_D_B.step()
        ###################################

        # epoch_log
        epoch_log["loss_G"] = epoch_log.get("loss_G", 0) + loss_G.item()
        epoch_log["loss_G_identity"] = epoch_log.get("loss_G_identity", 0) + (loss_iden
        epoch_log["loss_G_GAN"] = epoch_log.get("loss_G_GAN", 0) + (loss_GAN_A2B + loss
        epoch_log["loss_G_cycle"] = epoch_log.get("loss_G_cycle", 0) + (loss_cycle_ABA
        epoch_log["loss_D"] = epoch_log.get("loss_D", 0) + (loss_D_A + loss_D_B).item()

        if i == 0 and epoch == 0:
            print("Init images")
            images={'real_A': real_A[0], 'fake_B': fake_B[0], 'real_B': real_B[0], 'fak
            plot_images(images)

    # Update learning rates
    lr_scheduler_G.step()
    lr_scheduler_D_A.step()
    lr_scheduler_D_B.step()

    # logs update
    for loss_item in log:
        epoch_log[loss_item] /= len(dataloader)
        log[loss_item].append(epoch_log[loss_item])

    # log print
    loss_line = ", ".join([f"{loss_item}: {epoch_log[loss_item]:.4f}" for loss_item in
    log_line = f"{mode} - {loss_line}"
    print(log_line)
    if plot_images_:
        if (epoch + 1) % plot_img_step == 0:
            images={'real_A': real_A[0], 'fake_B': fake_B[0], 'real_B': real_B[0], 'fak
            plot_images(images)

    # Save models checkpoints and log
    if not os.path.exists(f'output/{output_name}'):
        os.makedirs(f'output/{output_name}')
```

```
    torch.save(netG_A2B.state_dict(), f'output/{output_name}/netG_A2B.pth')
    torch.save(netG_B2A.state_dict(), f'output/{output_name}/netG_B2A.pth')
    torch.save(netD_A.state_dict(), f'output/{output_name}/netD_A.pth')
    torch.save(netD_B.state_dict(), f'output/{output_name}/netD_B.pth')

    with open(f"output/{output_name}/train_log.json", "w") as file:
        json.dump(log, file)
```

In [ ]:

```
def test(netG_A2B, netG_B2A, netD_A, netD_B, dataloader, epoch, plot_images_=False, plo
    mode = "TEST"
    epoch_log = {}
    lambda_ = 10
    for i, batch in enumerate(dataloader):
        # Set model input
        real_A = batch['A'].to(device)
        real_B = batch['B'].to(device)

        ###### Generators A2B and B2A ######
        with torch.no_grad():
            # Identity loss
            # G_A2B(B) should equal B if real B is fed
            same_B = netG_A2B(real_B)
            loss_identity_B = criterion_identity(same_B, real_B) * 0.5 * lambda_
            # G_B2A(A) should equal A if real A is fed
            same_A = netG_B2A(real_A)
            loss_identity_A = criterion_identity(same_A, real_A) * 0.5 * lambda_

            # GAN loss
            fake_B = netG_A2B(real_A)
            pred_fake = netD_B(fake_B)
            loss_GAN_A2B = criterion_GAN(pred_fake, target_real)

            fake_A = netG_B2A(real_B)
            pred_fake = netD_A(fake_A)
            loss_GAN_B2A = criterion_GAN(pred_fake, target_real)

            # Cycle loss
            recovered_A = netG_B2A(fake_B)
            loss_cycle_ABA = criterion_cycle(recovered_A, real_A) * lambda_

            recovered_B = netG_A2B(fake_A)
            loss_cycle_BAB = criterion_cycle(recovered_B, real_B) * lambda_

            # Total loss
            loss_G =  (loss_identity_A + loss_identity_B) + (loss_GAN_A2B + loss_GAN_B2

            ###################################

            ###### Discriminator A ######

            # Real loss
            pred_real = netD_A(real_A)
            loss_D_real = criterion_GAN(pred_real, target_real)

            # Fake loss
            pred_fake = netD_A(fake_A.detach())
            loss_D_fake = criterion_GAN(pred_fake, target_fake)

            # Total loss
            loss_D_A = (loss_D_real + loss_D_fake) / 2
```

```
            ###################################

            ###### Discriminator B ######
            # Real loss
            pred_real = netD_B(real_B)
            loss_D_real = criterion_GAN(pred_real, target_real)

            # Fake loss
            pred_fake = netD_B(fake_B.detach())
            loss_D_fake = criterion_GAN(pred_fake, target_fake)

            # Total loss
            loss_D_B = (loss_D_real + loss_D_fake) / 2
            ###################################

            # epoch_log
            epoch_log["loss_G"] = epoch_log.get("loss_G", 0) + loss_G.item()
            epoch_log["loss_G_identity"] = epoch_log.get("loss_G_identity", 0) + (loss_
            epoch_log["loss_G_GAN"] = epoch_log.get("loss_G_GAN", 0) + (loss_GAN_A2B +
            epoch_log["loss_G_cycle"] = epoch_log.get("loss_G_cycle", 0) + (loss_cycle_
            epoch_log["loss_D"] = epoch_log.get("loss_D", 0) + (loss_D_A + loss_D_B).it

    # logs update
    for loss_item in log:
        epoch_log[loss_item] /= len(dataloader)
        log[loss_item].append(epoch_log[loss_item])

    # log print and plot of images
    loss_line = ", ".join([f"{loss_item}: {epoch_log[loss_item]:.4f}" for loss_item in
    log_line = f"{mode} -  {loss_line}"
    print(log_line)
    if plot_images_:
        if (epoch + 1) % plot_img_step == 0:
            images={'real_A': real_A[0], 'fake_B': fake_B[0], 'real_B': real_B[0], 'fak
            plot_images(images)

    # Save log and the best G models
    if not os.path.exists(f'output/{output_name}/best'):
        os.makedirs(f'output/{output_name}/best')

    with open(f"output/{output_name}/test_log.json", "w") as file:
        json.dump(log, file)

    if log["loss_G"][-1] <= min(log["loss_G"]):
        torch.save(netG_A2B.state_dict(), f'output/{output_name}/best/netG_A2B.pth')
        torch.save(netG_B2A.state_dict(), f'output/{output_name}/best/netG_B2A.pth')
```

инициализация для обучения - параметры, классы моделей и таргеты

In [ ]:

```
# params init
n_epochs = 200  # number of epochs of training
batchSize = 3  # size of the batches
dataroot = f'datasets/{dataset_name}/'  # root directory of the dataset
lr = 0.0002  # initial learning rate
decay_epoch = 100  # epoch to start linearly decaying the learning rate to 0
size = 256  # size of the data crop (squared assumed)
input_nc = 3  # number of channels of input data
output_nc = 3  # number of channels of output data
```

```
n_cpu = 8  # number of cpu threads to use during batch generation

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


###### Definition of variables ######
# Networks
netG_A2B = Generator(input_nc, output_nc).to(device)
netG_B2A = Generator(output_nc, input_nc).to(device)
netD_A = Discriminator(input_nc).to(device)
netD_B = Discriminator(output_nc).to(device)

netG_A2B.apply(weights_init_normal)
netG_B2A.apply(weights_init_normal)
netD_A.apply(weights_init_normal)
netD_B.apply(weights_init_normal)

# Lossess
criterion_GAN = torch.nn.MSELoss()
criterion_cycle = torch.nn.L1Loss()
criterion_identity = torch.nn.L1Loss()

# Optimizers & LR schedulers
optimizer_G = torch.optim.Adam(itertools.chain(netG_A2B.parameters(), netG_B2A.paramete
                               lr=lr, betas=(0.5, 0.999))
optimizer_D_A = torch.optim.Adam(netD_A.parameters(), lr=lr, betas=(0.5, 0.999))
optimizer_D_B = torch.optim.Adam(netD_B.parameters(), lr=lr, betas=(0.5, 0.999))

lr_scheduler_G = torch.optim.lr_scheduler.LambdaLR(optimizer_G, lr_lambda=LambdaLR(n_ep
lr_scheduler_D_A = torch.optim.lr_scheduler.LambdaLR(optimizer_D_A, lr_lambda=LambdaLR(
lr_scheduler_D_B = torch.optim.lr_scheduler.LambdaLR(optimizer_D_B, lr_lambda=LambdaLR(

# Inputs
target_real = torch.ones(batchSize, 1, requires_grad=False).to(device)
target_fake = torch.zeros(batchSize, 1, requires_grad=False).to(device)

fake_A_buffer = ImageBuffer()
fake_B_buffer = ImageBuffer()
```

инициализация даталоадеров

In [ ]:

```
# Dataset loaders
num_samples =300  # датасет ограниченный

train_transforms = [transforms.Resize(int(size*1.12),  Image.BICUBIC),
                    transforms.RandomCrop(size),
                    transforms.RandomHorizontalFlip(),
                    transforms.ToTensor(),
                    transforms.Normalize((0.5,0.5,0.5), (0.5,0.5,0.5))]
train_dataset = ImageDataset(dataroot, transforms_=train_transforms, unaligned=True)
train_sampler = RandomSampler(train_dataset, replacement=True, num_samples=num_samples)
train_dataloader = DataLoader(train_dataset, sampler=train_sampler,
                        batch_size=batchSize, num_workers=n_cpu, drop_last=True)

test_transforms = [transforms.Resize(int(size),  Image.BICUBIC),
                   transforms.CenterCrop(size),
                   transforms.ToTensor(),
                   transforms.Normalize((0.5,0.5,0.5), (0.5,0.5,0.5))]
test_dataloader = DataLoader(ImageDataset(dataroot, transforms_=test_transforms, unalig
                        batch_size=batchSize, shuffle=True, num_workers=n_cpu, drop_las
```

непосредственно обучение

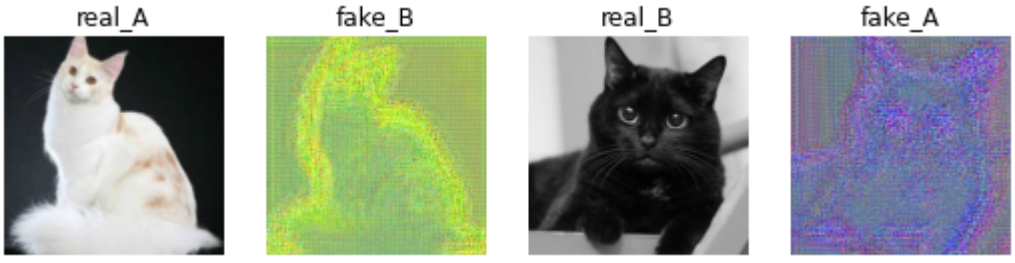
выводим на каждой эпохе информацию о лоссах и текущие тестовые картинки

In [ ]:
```
output_name = f"{dataset_name}_s{size}_ns{num_samples}_bs{batchSize}"
```

In [ ]:
```
def train_loop(from_checkpoint=False):
    start_epoch = 0
    # инициализация loss-логов
    losses = ['loss_G', 'loss_G_identity', 'loss_G_GAN', 'loss_G_cycle', 'loss_D']
    train_log = {loss: [] for loss in losses}
    test_log = {loss: [] for loss in losses}

    if from_checkpoint:
        with open(f"output/{output_name}/train_log.json", "r") as file:
            train_log = json.load(file)
        with open(f"output/{output_name}/test_log.json", "r") as file:
            test_log = json.load(file)

        start_epoch = len(train_log["loss_G"])

        # load state dicts
        netG_A2B.load_state_dict(torch.load(f'output/{output_name}/netG_A2B.pth', map_l
        netG_B2A.load_state_dict(torch.load(f'output/{output_name}/netG_B2A.pth', map_l
        netD_A.load_state_dict(torch.load(f'output/{output_name}/netD_A.pth', map_locat
        netD_B.load_state_dict(torch.load(f'output/{output_name}/netD_B.pth', map_locat

    for epoch in range(start_epoch, n_epochs):
        print(f"Epoch - {epoch+1}")
        netG_A2B.train()
        netG_B2A.train()
        netD_A.train()
        netD_B.train()
        train(netG_A2B, netG_B2A, netD_A, netD_B, train_dataloader, epoch, log=train_lo
        netG_A2B.eval()
        netG_B2A.eval()
        netD_A.eval()
        netD_B.eval()
        test(netG_A2B, netG_B2A, netD_A, netD_B, test_dataloader, epoch, plot_images_=T
```

In [ ]:
```
%%time
train_loop(from_checkpoint=False)
```

```
Epoch - 1
Init images
```




```
TRAIN - loss_G: 10.3229, loss_G_identity: 2.9785, loss_G_GAN: 1.0034, loss_G_cycle: 6.34
10, loss_D: 0.3903
TEST -  loss_G: 9.0135, loss_G_identity: 2.6900, loss_G_GAN: 1.2175, loss_G_cycle: 5.106
0, loss_D: 0.4249
Epoch - 2
TRAIN - loss_G: 8.9427, loss_G_identity: 2.5735, loss_G_GAN: 0.9886, loss_G_cycle: 5.380
```

```
5, loss_D: 0.3528
TEST -  loss_G: 9.5156, loss_G_identity: 2.3712, loss_G_GAN: 1.3115, loss_G_cycle: 5.833
0, loss_D: 0.2832
Epoch - 3
TRAIN - loss_G: 8.2415, loss_G_identity: 2.3587, loss_G_GAN: 1.0734, loss_G_cycle: 4.809
3, loss_D: 0.3149
TEST -  loss_G: 7.4273, loss_G_identity: 2.2874, loss_G_GAN: 0.9087, loss_G_cycle: 4.231
3, loss_D: 0.2312
Epoch - 4
TRAIN - loss_G: 8.0029, loss_G_identity: 2.2467, loss_G_GAN: 1.1731, loss_G_cycle: 4.583
0, loss_D: 0.2672
TEST -  loss_G: 8.7791, loss_G_identity: 2.7565, loss_G_GAN: 0.7601, loss_G_cycle: 5.262
5, loss_D: 0.3274
Epoch - 5
TRAIN - loss_G: 7.5171, loss_G_identity: 2.0579, loss_G_GAN: 1.1860, loss_G_cycle: 4.273
3, loss_D: 0.2453
TEST -  loss_G: 7.5669, loss_G_identity: 2.2285, loss_G_GAN: 1.0619, loss_G_cycle: 4.276
6, loss_D: 0.3233
Epoch - 6
TRAIN - loss_G: 7.1072, loss_G_identity: 1.8862, loss_G_GAN: 1.2451, loss_G_cycle: 3.975
9, loss_D: 0.2336
TEST -  loss_G: 6.7067, loss_G_identity: 1.7192, loss_G_GAN: 1.2158, loss_G_cycle: 3.771
7, loss_D: 0.2073
Epoch - 7
TRAIN - loss_G: 6.6709, loss_G_identity: 1.7064, loss_G_GAN: 1.2679, loss_G_cycle: 3.696
5, loss_D: 0.2134
TEST -  loss_G: 6.6453, loss_G_identity: 2.0812, loss_G_GAN: 0.7426, loss_G_cycle: 3.821
5, loss_D: 0.3402
Epoch - 8
TRAIN - loss_G: 7.0167, loss_G_identity: 1.7796, loss_G_GAN: 1.3179, loss_G_cycle: 3.919
2, loss_D: 0.2099
TEST -  loss_G: 7.6452, loss_G_identity: 1.8389, loss_G_GAN: 1.1437, loss_G_cycle: 4.662
6, loss_D: 0.3093
Epoch - 9
TRAIN - loss_G: 6.3901, loss_G_identity: 1.6056, loss_G_GAN: 1.3202, loss_G_cycle: 3.464
3, loss_D: 0.2016
TEST -  loss_G: 6.6066, loss_G_identity: 1.6491, loss_G_GAN: 1.4700, loss_G_cycle: 3.487
5, loss_D: 0.2130
Epoch - 10
TRAIN - loss_G: 6.4070, loss_G_identity: 1.6148, loss_G_GAN: 1.3215, loss_G_cycle: 3.470
7, loss_D: 0.2295
TEST -  loss_G: 6.7931, loss_G_identity: 1.6810, loss_G_GAN: 1.2834, loss_G_cycle: 3.828
7, loss_D: 0.2613
```


```
Epoch - 11
TRAIN - loss_G: 6.0331, loss_G_identity: 1.5099, loss_G_GAN: 1.3015, loss_G_cycle: 3.221
7, loss_D: 0.2034
TEST -  loss_G: 6.6311, loss_G_identity: 1.7784, loss_G_GAN: 1.2803, loss_G_cycle: 3.572
4, loss_D: 0.2401
Epoch - 12
TRAIN - loss_G: 6.2712, loss_G_identity: 1.6196, loss_G_GAN: 1.3091, loss_G_cycle: 3.342
6, loss_D: 0.2077
TEST -  loss_G: 7.1468, loss_G_identity: 1.7666, loss_G_GAN: 1.7476, loss_G_cycle: 3.632
6, loss_D: 0.2259
Epoch - 13
TRAIN - loss_G: 6.2076, loss_G_identity: 1.5615, loss_G_GAN: 1.3227, loss_G_cycle: 3.323
```

```
4, loss_D: 0.2010
TEST -  loss_G: 6.1886, loss_G_identity: 1.7767, loss_G_GAN: 1.2148, loss_G_cycle: 3.197
1, loss_D: 0.2537
Epoch - 14
TRAIN - loss_G: 6.3501, loss_G_identity: 1.5946, loss_G_GAN: 1.3295, loss_G_cycle: 3.426
0, loss_D: 0.2077
TEST -  loss_G: 6.3754, loss_G_identity: 1.6250, loss_G_GAN: 1.2293, loss_G_cycle: 3.521
1, loss_D: 0.2961
Epoch - 15
TRAIN - loss_G: 5.9468, loss_G_identity: 1.4727, loss_G_GAN: 1.2862, loss_G_cycle: 3.188
0, loss_D: 0.1831
TEST -  loss_G: 6.0855, loss_G_identity: 1.6078, loss_G_GAN: 0.8279, loss_G_cycle: 3.649
8, loss_D: 0.4034
Epoch - 16
TRAIN - loss_G: 5.8992, loss_G_identity: 1.4481, loss_G_GAN: 1.3143, loss_G_cycle: 3.136
8, loss_D: 0.1885
TEST -  loss_G: 6.1494, loss_G_identity: 1.5243, loss_G_GAN: 1.2153, loss_G_cycle: 3.409
7, loss_D: 0.2845
Epoch - 17
TRAIN - loss_G: 5.8068, loss_G_identity: 1.4265, loss_G_GAN: 1.3363, loss_G_cycle: 3.044
0, loss_D: 0.1973
TEST -  loss_G: 5.8005, loss_G_identity: 1.6657, loss_G_GAN: 0.9980, loss_G_cycle: 3.136
8, loss_D: 0.3079
Epoch - 18
TRAIN - loss_G: 6.1828, loss_G_identity: 1.4930, loss_G_GAN: 1.3331, loss_G_cycle: 3.356
7, loss_D: 0.1861
TEST -  loss_G: 5.6759, loss_G_identity: 1.6693, loss_G_GAN: 0.7397, loss_G_cycle: 3.266
9, loss_D: 0.3248
Epoch - 19
TRAIN - loss_G: 5.8181, loss_G_identity: 1.4297, loss_G_GAN: 1.3100, loss_G_cycle: 3.078
4, loss_D: 0.1929
TEST -  loss_G: 6.9118, loss_G_identity: 1.5941, loss_G_GAN: 2.0083, loss_G_cycle: 3.309
4, loss_D: 0.3580
Epoch - 20
TRAIN - loss_G: 5.9869, loss_G_identity: 1.4522, loss_G_GAN: 1.3442, loss_G_cycle: 3.190
5, loss_D: 0.1848
TEST -  loss_G: 6.3245, loss_G_identity: 1.5420, loss_G_GAN: 1.4691, loss_G_cycle: 3.313
3, loss_D: 0.2730
```


```
Epoch - 21
TRAIN - loss_G: 5.8717, loss_G_identity: 1.4666, loss_G_GAN: 1.2726, loss_G_cycle: 3.132
5, loss_D: 0.1966
TEST -  loss_G: 6.7168, loss_G_identity: 1.6729, loss_G_GAN: 1.3775, loss_G_cycle: 3.666
4, loss_D: 0.2994
Epoch - 22
TRAIN - loss_G: 5.7593, loss_G_identity: 1.4401, loss_G_GAN: 1.2588, loss_G_cycle: 3.060
5, loss_D: 0.1977
TEST -  loss_G: 6.5922, loss_G_identity: 1.4871, loss_G_GAN: 1.4200, loss_G_cycle: 3.685
1, loss_D: 0.2365
Epoch - 23
TRAIN - loss_G: 5.6245, loss_G_identity: 1.4023, loss_G_GAN: 1.2914, loss_G_cycle: 2.930
8, loss_D: 0.1938
TEST -  loss_G: 6.2890, loss_G_identity: 1.5627, loss_G_GAN: 1.3957, loss_G_cycle: 3.330
6, loss_D: 0.4032
Epoch - 24
TRAIN - loss_G: 5.6890, loss_G_identity: 1.4030, loss_G_GAN: 1.2864, loss_G_cycle: 2.999
```

```
6, loss_D: 0.2090
TEST -  loss_G: 5.5903, loss_G_identity: 1.5811, loss_G_GAN: 0.7478, loss_G_cycle: 3.261
4, loss_D: 0.3554
Epoch - 25
TRAIN - loss_G: 5.5923, loss_G_identity: 1.4072, loss_G_GAN: 1.2338, loss_G_cycle: 2.951
3, loss_D: 0.2026
TEST -  loss_G: 5.4131, loss_G_identity: 1.5219, loss_G_GAN: 0.8438, loss_G_cycle: 3.047
4, loss_D: 0.3255
Epoch - 26
TRAIN - loss_G: 5.5160, loss_G_identity: 1.3522, loss_G_GAN: 1.1828, loss_G_cycle: 2.981
0, loss_D: 0.2185
TEST -  loss_G: 5.1926, loss_G_identity: 1.4328, loss_G_GAN: 0.6844, loss_G_cycle: 3.075
3, loss_D: 0.3762
Epoch - 27
TRAIN - loss_G: 5.5334, loss_G_identity: 1.3695, loss_G_GAN: 1.1865, loss_G_cycle: 2.977
5, loss_D: 0.2089
TEST -  loss_G: 6.5547, loss_G_identity: 1.5473, loss_G_GAN: 1.8454, loss_G_cycle: 3.162
0, loss_D: 0.2771
Epoch - 28
TRAIN - loss_G: 5.4255, loss_G_identity: 1.2807, loss_G_GAN: 1.2291, loss_G_cycle: 2.915
7, loss_D: 0.2422
TEST -  loss_G: 5.2973, loss_G_identity: 1.4764, loss_G_GAN: 0.8944, loss_G_cycle: 2.926
5, loss_D: 0.4045
Epoch - 29
TRAIN - loss_G: 5.4671, loss_G_identity: 1.3627, loss_G_GAN: 1.1829, loss_G_cycle: 2.921
6, loss_D: 0.2034
TEST -  loss_G: 5.7447, loss_G_identity: 1.4187, loss_G_GAN: 1.0481, loss_G_cycle: 3.278
0, loss_D: 0.3766
Epoch - 30
TRAIN - loss_G: 5.5092, loss_G_identity: 1.3308, loss_G_GAN: 1.1952, loss_G_cycle: 2.983
3, loss_D: 0.2320
TEST -  loss_G: 6.1308, loss_G_identity: 1.5553, loss_G_GAN: 1.3016, loss_G_cycle: 3.273
8, loss_D: 0.3262
```


```
Epoch - 31
TRAIN - loss_G: 5.6054, loss_G_identity: 1.3582, loss_G_GAN: 1.2107, loss_G_cycle: 3.036
5, loss_D: 0.2069
TEST -  loss_G: 6.1096, loss_G_identity: 1.5142, loss_G_GAN: 1.0893, loss_G_cycle: 3.506
0, loss_D: 0.3765
Epoch - 32
TRAIN - loss_G: 5.7141, loss_G_identity: 1.3850, loss_G_GAN: 1.2250, loss_G_cycle: 3.104
0, loss_D: 0.2485
TEST -  loss_G: 6.0113, loss_G_identity: 1.4126, loss_G_GAN: 1.1377, loss_G_cycle: 3.461
0, loss_D: 0.3191
Epoch - 33
TRAIN - loss_G: 5.4780, loss_G_identity: 1.3170, loss_G_GAN: 1.2460, loss_G_cycle: 2.915
0, loss_D: 0.1891
TEST -  loss_G: 5.4614, loss_G_identity: 1.5705, loss_G_GAN: 0.8745, loss_G_cycle: 3.016
5, loss_D: 0.3989
Epoch - 34
TRAIN - loss_G: 5.2874, loss_G_identity: 1.3065, loss_G_GAN: 1.1565, loss_G_cycle: 2.824
4, loss_D: 0.2271
TEST -  loss_G: 6.2174, loss_G_identity: 1.4919, loss_G_GAN: 1.4826, loss_G_cycle: 3.242
9, loss_D: 0.4309
Epoch - 35
TRAIN - loss_G: 5.3694, loss_G_identity: 1.3186, loss_G_GAN: 1.1776, loss_G_cycle: 2.873
```

```
2, loss_D: 0.2206
TEST -  loss_G: 7.0635, loss_G_identity: 1.5553, loss_G_GAN: 2.0168, loss_G_cycle: 3.491
4, loss_D: 0.3846
Epoch - 36
TRAIN - loss_G: 5.2770, loss_G_identity: 1.2643, loss_G_GAN: 1.2086, loss_G_cycle: 2.804
1, loss_D: 0.2106
TEST -  loss_G: 5.6801, loss_G_identity: 1.4958, loss_G_GAN: 0.7546, loss_G_cycle: 3.429
6, loss_D: 0.3612
Epoch - 37
TRAIN - loss_G: 5.1745, loss_G_identity: 1.2470, loss_G_GAN: 1.1498, loss_G_cycle: 2.777
7, loss_D: 0.2287
TEST -  loss_G: 5.3218, loss_G_identity: 1.4940, loss_G_GAN: 0.9563, loss_G_cycle: 2.871
4, loss_D: 0.4927
Epoch - 38
TRAIN - loss_G: 5.1460, loss_G_identity: 1.2128, loss_G_GAN: 1.1710, loss_G_cycle: 2.762
3, loss_D: 0.2448
TEST -  loss_G: 5.9133, loss_G_identity: 1.5992, loss_G_GAN: 1.0969, loss_G_cycle: 3.217
2, loss_D: 0.3883
Epoch - 39
TRAIN - loss_G: 5.2811, loss_G_identity: 1.2345, loss_G_GAN: 1.2531, loss_G_cycle: 2.793
5, loss_D: 0.1993
TEST -  loss_G: 5.2883, loss_G_identity: 1.4358, loss_G_GAN: 0.6756, loss_G_cycle: 3.176
9, loss_D: 0.3612
Epoch - 40
TRAIN - loss_G: 5.3279, loss_G_identity: 1.2352, loss_G_GAN: 1.2958, loss_G_cycle: 2.796
8, loss_D: 0.1685
TEST -  loss_G: 5.7167, loss_G_identity: 1.4002, loss_G_GAN: 1.1770, loss_G_cycle: 3.139
5, loss_D: 0.4819
```


```
Epoch - 41
TRAIN - loss_G: 5.0952, loss_G_identity: 1.2194, loss_G_GAN: 1.1333, loss_G_cycle: 2.742
5, loss_D: 0.2211
TEST -  loss_G: 5.5218, loss_G_identity: 1.4995, loss_G_GAN: 1.0150, loss_G_cycle: 3.007
3, loss_D: 0.2814
Epoch - 42
TRAIN - loss_G: 4.9334, loss_G_identity: 1.1432, loss_G_GAN: 1.1538, loss_G_cycle: 2.636
4, loss_D: 0.2343
TEST -  loss_G: 5.1059, loss_G_identity: 1.3281, loss_G_GAN: 1.0790, loss_G_cycle: 2.698
8, loss_D: 0.4308
Epoch - 43
TRAIN - loss_G: 5.0218, loss_G_identity: 1.1862, loss_G_GAN: 1.1648, loss_G_cycle: 2.670
7, loss_D: 0.2175
TEST -  loss_G: 6.9827, loss_G_identity: 1.3903, loss_G_GAN: 1.7617, loss_G_cycle: 3.830
6, loss_D: 0.2387
Epoch - 44
TRAIN - loss_G: 4.9829, loss_G_identity: 1.1533, loss_G_GAN: 1.1708, loss_G_cycle: 2.658
7, loss_D: 0.2014
TEST -  loss_G: 5.2837, loss_G_identity: 1.3847, loss_G_GAN: 0.9160, loss_G_cycle: 2.982
9, loss_D: 0.2952
Epoch - 45
TRAIN - loss_G: 4.9530, loss_G_identity: 1.1449, loss_G_GAN: 1.1870, loss_G_cycle: 2.621
1, loss_D: 0.2350
TEST -  loss_G: 5.3477, loss_G_identity: 1.3159, loss_G_GAN: 1.1151, loss_G_cycle: 2.916
7, loss_D: 0.4164
Epoch - 46
TRAIN - loss_G: 4.7288, loss_G_identity: 1.0908, loss_G_GAN: 1.1117, loss_G_cycle: 2.526
```

```
4, loss_D: 0.2245
TEST -  loss_G: 5.5286, loss_G_identity: 1.4296, loss_G_GAN: 0.8329, loss_G_cycle: 3.266
2, loss_D: 0.3691
Epoch - 47
TRAIN - loss_G: 4.8926, loss_G_identity: 1.1490, loss_G_GAN: 1.1600, loss_G_cycle: 2.583
5, loss_D: 0.2154
TEST -  loss_G: 5.8293, loss_G_identity: 1.7547, loss_G_GAN: 1.0304, loss_G_cycle: 3.044
2, loss_D: 0.3496
Epoch - 48
TRAIN - loss_G: 5.1983, loss_G_identity: 1.1866, loss_G_GAN: 1.2460, loss_G_cycle: 2.765
7, loss_D: 0.1803
TEST -  loss_G: 5.0421, loss_G_identity: 1.3661, loss_G_GAN: 0.9401, loss_G_cycle: 2.735
9, loss_D: 0.4131
Epoch - 49
TRAIN - loss_G: 4.8171, loss_G_identity: 1.0980, loss_G_GAN: 1.1694, loss_G_cycle: 2.549
7, loss_D: 0.2294
TEST -  loss_G: 5.4486, loss_G_identity: 1.2417, loss_G_GAN: 1.3708, loss_G_cycle: 2.836
1, loss_D: 0.2717
Epoch - 50
TRAIN - loss_G: 4.9003, loss_G_identity: 1.1133, loss_G_GAN: 1.1954, loss_G_cycle: 2.591
6, loss_D: 0.2026
TEST -  loss_G: 5.2573, loss_G_identity: 1.3092, loss_G_GAN: 1.0063, loss_G_cycle: 2.941
8, loss_D: 0.3479
```


```
Epoch - 51
TRAIN - loss_G: 4.8225, loss_G_identity: 1.0901, loss_G_GAN: 1.1796, loss_G_cycle: 2.552
8, loss_D: 0.2098
TEST -  loss_G: 5.7882, loss_G_identity: 1.4027, loss_G_GAN: 0.8138, loss_G_cycle: 3.571
7, loss_D: 0.3611
Epoch - 52
TRAIN - loss_G: 4.9513, loss_G_identity: 1.0917, loss_G_GAN: 1.2319, loss_G_cycle: 2.627
7, loss_D: 0.1796
TEST -  loss_G: 6.1620, loss_G_identity: 1.4378, loss_G_GAN: 1.1970, loss_G_cycle: 3.527
1, loss_D: 0.3223
Epoch - 53
TRAIN - loss_G: 5.4046, loss_G_identity: 1.1560, loss_G_GAN: 1.3564, loss_G_cycle: 2.892
1, loss_D: 0.1733
TEST -  loss_G: 5.9895, loss_G_identity: 1.3610, loss_G_GAN: 1.6297, loss_G_cycle: 2.998
9, loss_D: 0.4426
Epoch - 54
TRAIN - loss_G: 4.6584, loss_G_identity: 1.0392, loss_G_GAN: 1.1557, loss_G_cycle: 2.463
5, loss_D: 0.1927
TEST -  loss_G: 5.3853, loss_G_identity: 1.3532, loss_G_GAN: 1.0313, loss_G_cycle: 3.000
7, loss_D: 0.3836
Epoch - 55
TRAIN - loss_G: 4.8960, loss_G_identity: 1.0907, loss_G_GAN: 1.2395, loss_G_cycle: 2.565
8, loss_D: 0.1862
TEST -  loss_G: 5.5448, loss_G_identity: 1.2714, loss_G_GAN: 1.3014, loss_G_cycle: 2.972
0, loss_D: 0.3733
Epoch - 56
TRAIN - loss_G: 4.7961, loss_G_identity: 1.0611, loss_G_GAN: 1.1843, loss_G_cycle: 2.550
8, loss_D: 0.2212
TEST -  loss_G: 4.7324, loss_G_identity: 1.2095, loss_G_GAN: 0.9341, loss_G_cycle: 2.588
8, loss_D: 0.3657
Epoch - 57
TRAIN - loss_G: 4.6430, loss_G_identity: 1.0335, loss_G_GAN: 1.1634, loss_G_cycle: 2.446
```

```
2, loss_D: 0.2130
TEST -  loss_G: 5.1768, loss_G_identity: 1.4916, loss_G_GAN: 0.6158, loss_G_cycle: 3.069
4, loss_D: 0.4800
Epoch - 58
TRAIN - loss_G: 4.8252, loss_G_identity: 1.0569, loss_G_GAN: 1.2387, loss_G_cycle: 2.529
6, loss_D: 0.2055
TEST -  loss_G: 5.5731, loss_G_identity: 1.3025, loss_G_GAN: 1.1692, loss_G_cycle: 3.101
4, loss_D: 0.3789
Epoch - 59
TRAIN - loss_G: 4.6288, loss_G_identity: 1.0208, loss_G_GAN: 1.1596, loss_G_cycle: 2.448
3, loss_D: 0.2183
TEST -  loss_G: 5.9895, loss_G_identity: 1.4399, loss_G_GAN: 1.1846, loss_G_cycle: 3.365
0, loss_D: 0.4146
Epoch - 60
TRAIN - loss_G: 4.7240, loss_G_identity: 1.0326, loss_G_GAN: 1.1520, loss_G_cycle: 2.539
3, loss_D: 0.2190
TEST -  loss_G: 5.7036, loss_G_identity: 1.3953, loss_G_GAN: 1.4536, loss_G_cycle: 2.854
6, loss_D: 0.3563
```


```
Epoch - 61
TRAIN - loss_G: 4.6031, loss_G_identity: 0.9806, loss_G_GAN: 1.1185, loss_G_cycle: 2.504
0, loss_D: 0.2017
TEST -  loss_G: 6.0252, loss_G_identity: 1.3694, loss_G_GAN: 1.3299, loss_G_cycle: 3.325
9, loss_D: 0.4091
Epoch - 62
TRAIN - loss_G: 4.5946, loss_G_identity: 0.9820, loss_G_GAN: 1.1588, loss_G_cycle: 2.453
8, loss_D: 0.1936
TEST -  loss_G: 6.1147, loss_G_identity: 1.2271, loss_G_GAN: 2.0510, loss_G_cycle: 2.836
7, loss_D: 0.7039
Epoch - 63
TRAIN - loss_G: 4.4862, loss_G_identity: 0.9453, loss_G_GAN: 1.1883, loss_G_cycle: 2.352
6, loss_D: 0.2107
TEST -  loss_G: 4.6752, loss_G_identity: 1.2927, loss_G_GAN: 0.8918, loss_G_cycle: 2.490
7, loss_D: 0.3726
Epoch - 64
TRAIN - loss_G: 4.5211, loss_G_identity: 0.9538, loss_G_GAN: 1.1749, loss_G_cycle: 2.392
4, loss_D: 0.2150
TEST -  loss_G: 5.7641, loss_G_identity: 1.2370, loss_G_GAN: 1.4842, loss_G_cycle: 3.042
9, loss_D: 0.4088
Epoch - 65
TRAIN - loss_G: 4.4859, loss_G_identity: 0.9656, loss_G_GAN: 1.1210, loss_G_cycle: 2.399
2, loss_D: 0.2008
TEST -  loss_G: 5.7755, loss_G_identity: 1.1892, loss_G_GAN: 1.5728, loss_G_cycle: 3.013
5, loss_D: 0.4861
Epoch - 66
TRAIN - loss_G: 4.3527, loss_G_identity: 0.9062, loss_G_GAN: 1.1187, loss_G_cycle: 2.327
8, loss_D: 0.2183
TEST -  loss_G: 5.5772, loss_G_identity: 1.2538, loss_G_GAN: 1.2708, loss_G_cycle: 3.052
6, loss_D: 0.5487
Epoch - 67
TRAIN - loss_G: 4.3816, loss_G_identity: 0.9278, loss_G_GAN: 1.1408, loss_G_cycle: 2.313
1, loss_D: 0.2261
TEST -  loss_G: 5.5170, loss_G_identity: 1.2343, loss_G_GAN: 1.3564, loss_G_cycle: 2.926
3, loss_D: 0.4020
Epoch - 68
TRAIN - loss_G: 4.2744, loss_G_identity: 0.9137, loss_G_GAN: 1.1045, loss_G_cycle: 2.256
```

```
2, loss_D: 0.2206
TEST -  loss_G: 6.2736, loss_G_identity: 1.3020, loss_G_GAN: 1.9449, loss_G_cycle: 3.026
7, loss_D: 0.6077
Epoch - 69
TRAIN - loss_G: 4.5316, loss_G_identity: 0.9227, loss_G_GAN: 1.2099, loss_G_cycle: 2.398
9, loss_D: 0.2405
TEST -  loss_G: 6.4466, loss_G_identity: 1.3406, loss_G_GAN: 1.8038, loss_G_cycle: 3.302
2, loss_D: 0.4404
Epoch - 70
TRAIN - loss_G: 4.6284, loss_G_identity: 0.9242, loss_G_GAN: 1.2939, loss_G_cycle: 2.410
3, loss_D: 0.1848
TEST -  loss_G: 5.0194, loss_G_identity: 1.3537, loss_G_GAN: 0.7998, loss_G_cycle: 2.865
8, loss_D: 0.4468
```


```
Epoch - 71
TRAIN - loss_G: 4.3193, loss_G_identity: 0.9068, loss_G_GAN: 1.1048, loss_G_cycle: 2.307
7, loss_D: 0.2035
TEST -  loss_G: 5.6752, loss_G_identity: 1.2929, loss_G_GAN: 1.3109, loss_G_cycle: 3.071
4, loss_D: 0.5107
Epoch - 72
TRAIN - loss_G: 4.5218, loss_G_identity: 0.9373, loss_G_GAN: 1.2051, loss_G_cycle: 2.379
4, loss_D: 0.2144
TEST -  loss_G: 5.3358, loss_G_identity: 1.4813, loss_G_GAN: 1.0024, loss_G_cycle: 2.852
1, loss_D: 0.3846
Epoch - 73
TRAIN - loss_G: 4.6729, loss_G_identity: 0.9966, loss_G_GAN: 1.2941, loss_G_cycle: 2.382
3, loss_D: 0.1702
TEST -  loss_G: 5.4385, loss_G_identity: 1.4270, loss_G_GAN: 1.1328, loss_G_cycle: 2.878
6, loss_D: 0.3594
Epoch - 74
TRAIN - loss_G: 4.2704, loss_G_identity: 0.8763, loss_G_GAN: 1.1733, loss_G_cycle: 2.220
7, loss_D: 0.2160
TEST -  loss_G: 5.0856, loss_G_identity: 1.2494, loss_G_GAN: 0.9089, loss_G_cycle: 2.927
3, loss_D: 0.3623
Epoch - 75
TRAIN - loss_G: 4.2938, loss_G_identity: 0.8928, loss_G_GAN: 1.1538, loss_G_cycle: 2.247
2, loss_D: 0.1886
TEST -  loss_G: 5.2852, loss_G_identity: 1.1802, loss_G_GAN: 1.3363, loss_G_cycle: 2.768
7, loss_D: 0.5928
Epoch - 76
TRAIN - loss_G: 4.3869, loss_G_identity: 0.9009, loss_G_GAN: 1.1681, loss_G_cycle: 2.318
0, loss_D: 0.2192
TEST -  loss_G: 5.3851, loss_G_identity: 1.4013, loss_G_GAN: 0.7950, loss_G_cycle: 3.188
8, loss_D: 0.4907
Epoch - 77
TRAIN - loss_G: 4.2538, loss_G_identity: 0.8616, loss_G_GAN: 1.1232, loss_G_cycle: 2.269
1, loss_D: 0.2187
TEST -  loss_G: 5.0446, loss_G_identity: 1.1971, loss_G_GAN: 1.2180, loss_G_cycle: 2.629
4, loss_D: 0.5447
Epoch - 78
TRAIN - loss_G: 4.2824, loss_G_identity: 0.8897, loss_G_GAN: 1.1656, loss_G_cycle: 2.227
1, loss_D: 0.2257
TEST -  loss_G: 5.5269, loss_G_identity: 1.1968, loss_G_GAN: 1.5126, loss_G_cycle: 2.817
6, loss_D: 0.4809
Epoch - 79
TRAIN - loss_G: 4.1098, loss_G_identity: 0.8501, loss_G_GAN: 1.0997, loss_G_cycle: 2.159
```

```
9, loss_D: 0.2132
TEST -  loss_G: 5.2390, loss_G_identity: 1.2322, loss_G_GAN: 1.0404, loss_G_cycle: 2.966
4, loss_D: 0.5002
Epoch - 80
TRAIN - loss_G: 4.2471, loss_G_identity: 0.8662, loss_G_GAN: 1.1309, loss_G_cycle: 2.250
0, loss_D: 0.2153
TEST -  loss_G: 5.5301, loss_G_identity: 1.2531, loss_G_GAN: 1.0818, loss_G_cycle: 3.195
2, loss_D: 0.4843
```


```
Epoch - 81
TRAIN - loss_G: 4.2775, loss_G_identity: 0.8677, loss_G_GAN: 1.1718, loss_G_cycle: 2.238
0, loss_D: 0.2208
TEST -  loss_G: 5.2643, loss_G_identity: 1.3788, loss_G_GAN: 1.0006, loss_G_cycle: 2.884
9, loss_D: 0.4475
Epoch - 82
TRAIN - loss_G: 4.1478, loss_G_identity: 0.8296, loss_G_GAN: 1.1570, loss_G_cycle: 2.161
2, loss_D: 0.1997
TEST -  loss_G: 4.9336, loss_G_identity: 1.2570, loss_G_GAN: 1.0167, loss_G_cycle: 2.660
0, loss_D: 0.4185
Epoch - 83
TRAIN - loss_G: 4.1332, loss_G_identity: 0.8400, loss_G_GAN: 1.1524, loss_G_cycle: 2.140
9, loss_D: 0.2012
TEST -  loss_G: 4.9226, loss_G_identity: 1.2461, loss_G_GAN: 0.8849, loss_G_cycle: 2.791
5, loss_D: 0.5690
Epoch - 84
TRAIN - loss_G: 4.1157, loss_G_identity: 0.8375, loss_G_GAN: 1.1449, loss_G_cycle: 2.133
4, loss_D: 0.2203
TEST -  loss_G: 5.0959, loss_G_identity: 1.3303, loss_G_GAN: 1.0108, loss_G_cycle: 2.754
8, loss_D: 0.4818
Epoch - 85
TRAIN - loss_G: 4.0391, loss_G_identity: 0.8097, loss_G_GAN: 1.1417, loss_G_cycle: 2.087
7, loss_D: 0.2215
TEST -  loss_G: 5.0064, loss_G_identity: 1.1893, loss_G_GAN: 1.2080, loss_G_cycle: 2.609
1, loss_D: 0.4861
Epoch - 86
TRAIN - loss_G: 4.3336, loss_G_identity: 0.8604, loss_G_GAN: 1.2115, loss_G_cycle: 2.261
7, loss_D: 0.1977
TEST -  loss_G: 5.1366, loss_G_identity: 1.2704, loss_G_GAN: 0.8457, loss_G_cycle: 3.020
5, loss_D: 0.4629
Epoch - 87
TRAIN - loss_G: 4.1692, loss_G_identity: 0.8257, loss_G_GAN: 1.2007, loss_G_cycle: 2.142
7, loss_D: 0.2040
TEST -  loss_G: 5.3117, loss_G_identity: 1.4253, loss_G_GAN: 1.0051, loss_G_cycle: 2.881
2, loss_D: 0.4220
Epoch - 88
TRAIN - loss_G: 4.2089, loss_G_identity: 0.8386, loss_G_GAN: 1.1941, loss_G_cycle: 2.176
2, loss_D: 0.1907
TEST -  loss_G: 5.0191, loss_G_identity: 1.2179, loss_G_GAN: 1.1239, loss_G_cycle: 2.677
4, loss_D: 0.5939
Epoch - 89
TRAIN - loss_G: 4.2770, loss_G_identity: 0.8343, loss_G_GAN: 1.2336, loss_G_cycle: 2.209
1, loss_D: 0.1828
TEST -  loss_G: 5.2286, loss_G_identity: 1.2275, loss_G_GAN: 1.1785, loss_G_cycle: 2.822
7, loss_D: 0.4082
Epoch - 90
TRAIN - loss_G: 4.4230, loss_G_identity: 0.8838, loss_G_GAN: 1.2479, loss_G_cycle: 2.291
```

```
3, loss_D: 0.1819
TEST -  loss_G: 5.1453, loss_G_identity: 1.4382, loss_G_GAN: 0.9251, loss_G_cycle: 2.782
0, loss_D: 0.4417
```


```
Epoch - 91
TRAIN - loss_G: 4.1300, loss_G_identity: 0.8346, loss_G_GAN: 1.1710, loss_G_cycle: 2.124
5, loss_D: 0.2106
TEST -  loss_G: 4.9998, loss_G_identity: 1.2138, loss_G_GAN: 0.9937, loss_G_cycle: 2.792
3, loss_D: 0.4682
Epoch - 92
TRAIN - loss_G: 4.2492, loss_G_identity: 0.8412, loss_G_GAN: 1.2240, loss_G_cycle: 2.184
0, loss_D: 0.1961
TEST -  loss_G: 5.1622, loss_G_identity: 1.2931, loss_G_GAN: 1.0546, loss_G_cycle: 2.814
5, loss_D: 0.5612
Epoch - 93
TRAIN - loss_G: 4.1008, loss_G_identity: 0.8089, loss_G_GAN: 1.2092, loss_G_cycle: 2.082
8, loss_D: 0.1761
TEST -  loss_G: 5.5924, loss_G_identity: 1.2627, loss_G_GAN: 1.2375, loss_G_cycle: 3.092
2, loss_D: 0.4815
Epoch - 94
TRAIN - loss_G: 4.0279, loss_G_identity: 0.7904, loss_G_GAN: 1.1704, loss_G_cycle: 2.067
2, loss_D: 0.2251
TEST -  loss_G: 5.4627, loss_G_identity: 1.2678, loss_G_GAN: 1.4939, loss_G_cycle: 2.700
9, loss_D: 0.5396
Epoch - 95
TRAIN - loss_G: 4.1814, loss_G_identity: 0.8309, loss_G_GAN: 1.1889, loss_G_cycle: 2.161
7, loss_D: 0.1961
TEST -  loss_G: 5.1043, loss_G_identity: 1.3736, loss_G_GAN: 0.9731, loss_G_cycle: 2.757
7, loss_D: 0.4552
Epoch - 96
TRAIN - loss_G: 3.9717, loss_G_identity: 0.7813, loss_G_GAN: 1.1629, loss_G_cycle: 2.027
5, loss_D: 0.1930
TEST -  loss_G: 5.3436, loss_G_identity: 1.3337, loss_G_GAN: 1.3316, loss_G_cycle: 2.678
3, loss_D: 0.5178
Epoch - 97
TRAIN - loss_G: 4.8790, loss_G_identity: 0.9489, loss_G_GAN: 1.4038, loss_G_cycle: 2.526
3, loss_D: 0.1669
TEST -  loss_G: 5.7155, loss_G_identity: 1.2711, loss_G_GAN: 1.4922, loss_G_cycle: 2.952
3, loss_D: 0.3981
Epoch - 98
TRAIN - loss_G: 4.3668, loss_G_identity: 0.8536, loss_G_GAN: 1.2774, loss_G_cycle: 2.235
8, loss_D: 0.1727
TEST -  loss_G: 5.5408, loss_G_identity: 1.2566, loss_G_GAN: 1.5653, loss_G_cycle: 2.718
9, loss_D: 0.6211
Epoch - 99
TRAIN - loss_G: 4.0428, loss_G_identity: 0.7872, loss_G_GAN: 1.1952, loss_G_cycle: 2.060
4, loss_D: 0.1868
TEST -  loss_G: 5.4919, loss_G_identity: 1.2768, loss_G_GAN: 1.5873, loss_G_cycle: 2.627
7, loss_D: 0.6527
Epoch - 100
TRAIN - loss_G: 4.0512, loss_G_identity: 0.7795, loss_G_GAN: 1.2143, loss_G_cycle: 2.057
3, loss_D: 0.1824
TEST -  loss_G: 5.0963, loss_G_identity: 1.2985, loss_G_GAN: 0.8321, loss_G_cycle: 2.965
8, loss_D: 0.4646
```


```
Epoch - 101
TRAIN - loss_G: 4.0465, loss_G_identity: 0.7768, loss_G_GAN: 1.2244, loss_G_cycle: 2.045
3, loss_D: 0.1852
TEST -  loss_G: 4.9101, loss_G_identity: 1.3229, loss_G_GAN: 1.1139, loss_G_cycle: 2.473
3, loss_D: 0.5256
Epoch - 102
TRAIN - loss_G: 3.9641, loss_G_identity: 0.7644, loss_G_GAN: 1.1804, loss_G_cycle: 2.019
3, loss_D: 0.2008
TEST -  loss_G: 5.9851, loss_G_identity: 1.3179, loss_G_GAN: 1.5340, loss_G_cycle: 3.133
1, loss_D: 0.6207
Epoch - 103
TRAIN - loss_G: 4.0012, loss_G_identity: 0.7615, loss_G_GAN: 1.2092, loss_G_cycle: 2.030
5, loss_D: 0.2090
TEST -  loss_G: 5.5662, loss_G_identity: 1.1477, loss_G_GAN: 1.7742, loss_G_cycle: 2.644
3, loss_D: 0.6971
Epoch - 104
TRAIN - loss_G: 3.9666, loss_G_identity: 0.7634, loss_G_GAN: 1.1716, loss_G_cycle: 2.031
6, loss_D: 0.2083
TEST -  loss_G: 5.0231, loss_G_identity: 1.3469, loss_G_GAN: 1.0469, loss_G_cycle: 2.629
3, loss_D: 0.4321
Epoch - 105
TRAIN - loss_G: 4.1236, loss_G_identity: 0.7706, loss_G_GAN: 1.2628, loss_G_cycle: 2.090
1, loss_D: 0.1544
TEST -  loss_G: 5.7616, loss_G_identity: 1.4314, loss_G_GAN: 1.6142, loss_G_cycle: 2.715
9, loss_D: 0.6111
Epoch - 106
TRAIN - loss_G: 3.9989, loss_G_identity: 0.7494, loss_G_GAN: 1.2246, loss_G_cycle: 2.024
9, loss_D: 0.1584
TEST -  loss_G: 4.8377, loss_G_identity: 1.3494, loss_G_GAN: 0.7522, loss_G_cycle: 2.736
1, loss_D: 0.5105
Epoch - 107
TRAIN - loss_G: 3.9261, loss_G_identity: 0.7544, loss_G_GAN: 1.1963, loss_G_cycle: 1.975
4, loss_D: 0.2054
TEST -  loss_G: 5.5509, loss_G_identity: 1.2077, loss_G_GAN: 1.6405, loss_G_cycle: 2.702
7, loss_D: 0.5698
Epoch - 108
TRAIN - loss_G: 3.9119, loss_G_identity: 0.7523, loss_G_GAN: 1.1726, loss_G_cycle: 1.986
9, loss_D: 0.1820
TEST -  loss_G: 4.7163, loss_G_identity: 1.2511, loss_G_GAN: 0.7800, loss_G_cycle: 2.685
2, loss_D: 0.3978
Epoch - 109
TRAIN - loss_G: 4.0493, loss_G_identity: 0.7670, loss_G_GAN: 1.2719, loss_G_cycle: 2.010
4, loss_D: 0.1810
TEST -  loss_G: 5.5313, loss_G_identity: 1.2400, loss_G_GAN: 1.4856, loss_G_cycle: 2.805
6, loss_D: 0.5142
Epoch - 110
TEST -  loss_G: 4.7902, loss_G_identity: 1.1963, loss_G_GAN: 0.9964, loss_G_cycle: 2.597
4, loss_D: 0.4679
```


```
Epoch - 111
TRAIN - loss_G: 3.9783, loss_G_identity: 0.7515, loss_G_GAN: 1.2321, loss_G_cycle: 1.994
6, loss_D: 0.1858
TEST -  loss_G: 5.2414, loss_G_identity: 1.1842, loss_G_GAN: 1.5331, loss_G_cycle: 2.524
1, loss_D: 0.6228
Epoch - 112
TRAIN - loss_G: 3.8768, loss_G_identity: 0.7279, loss_G_GAN: 1.1824, loss_G_cycle: 1.966
5, loss_D: 0.1776
TEST -  loss_G: 4.8514, loss_G_identity: 1.1455, loss_G_GAN: 1.0897, loss_G_cycle: 2.616
3, loss_D: 0.4890
Epoch - 113
TRAIN - loss_G: 3.8753, loss_G_identity: 0.7252, loss_G_GAN: 1.1753, loss_G_cycle: 1.974
8, loss_D: 0.1690
TEST -  loss_G: 5.5714, loss_G_identity: 1.4523, loss_G_GAN: 1.4074, loss_G_cycle: 2.711
7, loss_D: 0.5782
Epoch - 114
TRAIN - loss_G: 3.7860, loss_G_identity: 0.6929, loss_G_GAN: 1.1874, loss_G_cycle: 1.905
7, loss_D: 0.1844
TEST -  loss_G: 4.5937, loss_G_identity: 1.2282, loss_G_GAN: 0.7363, loss_G_cycle: 2.629
2, loss_D: 0.4068
Epoch - 115
TRAIN - loss_G: 3.9324, loss_G_identity: 0.7318, loss_G_GAN: 1.2499, loss_G_cycle: 1.950
7, loss_D: 0.1688
TEST -  loss_G: 5.7569, loss_G_identity: 1.2847, loss_G_GAN: 1.7063, loss_G_cycle: 2.765
9, loss_D: 0.5919
Epoch - 116
TRAIN - loss_G: 3.8080, loss_G_identity: 0.7062, loss_G_GAN: 1.1949, loss_G_cycle: 1.906
9, loss_D: 0.1788
TEST -  loss_G: 4.7881, loss_G_identity: 1.1145, loss_G_GAN: 1.2537, loss_G_cycle: 2.419
8, loss_D: 0.5321
Epoch - 117
TRAIN - loss_G: 3.8653, loss_G_identity: 0.7287, loss_G_GAN: 1.1985, loss_G_cycle: 1.938
1, loss_D: 0.1831
TEST -  loss_G: 5.1401, loss_G_identity: 1.0533, loss_G_GAN: 1.3658, loss_G_cycle: 2.721
0, loss_D: 0.4945
Epoch - 118
TRAIN - loss_G: 3.6577, loss_G_identity: 0.6727, loss_G_GAN: 1.1671, loss_G_cycle: 1.817
9, loss_D: 0.1881
TEST -  loss_G: 5.3063, loss_G_identity: 1.2813, loss_G_GAN: 1.2502, loss_G_cycle: 2.774
8, loss_D: 0.4940
Epoch - 119
TRAIN - loss_G: 3.7831, loss_G_identity: 0.7001, loss_G_GAN: 1.1884, loss_G_cycle: 1.894
6, loss_D: 0.1685
TEST -  loss_G: 5.2732, loss_G_identity: 1.3293, loss_G_GAN: 1.3161, loss_G_cycle: 2.627
8, loss_D: 0.5037
Epoch - 120
TRAIN - loss_G: 3.7150, loss_G_identity: 0.6775, loss_G_GAN: 1.1971, loss_G_cycle: 1.840
4, loss_D: 0.1632
TEST -  loss_G: 5.1107, loss_G_identity: 1.4066, loss_G_GAN: 1.0726, loss_G_cycle: 2.631
5, loss_D: 0.4009
```


```
Epoch - 121
TRAIN - loss_G: 3.8173, loss_G_identity: 0.6874, loss_G_GAN: 1.2362, loss_G_cycle: 1.893
7, loss_D: 0.1540
TEST -  loss_G: 5.1356, loss_G_identity: 1.2808, loss_G_GAN: 1.2866, loss_G_cycle: 2.568
2, loss_D: 0.4990
Epoch - 122
TRAIN - loss_G: 3.8932, loss_G_identity: 0.6932, loss_G_GAN: 1.2669, loss_G_cycle: 1.933
0, loss_D: 0.1575
TEST -  loss_G: 5.2826, loss_G_identity: 1.5207, loss_G_GAN: 0.8589, loss_G_cycle: 2.903
0, loss_D: 0.5054
Epoch - 123
TRAIN - loss_G: 3.7629, loss_G_identity: 0.6857, loss_G_GAN: 1.1956, loss_G_cycle: 1.881
7, loss_D: 0.1559
TEST -  loss_G: 5.1806, loss_G_identity: 1.3283, loss_G_GAN: 1.2214, loss_G_cycle: 2.630
8, loss_D: 0.4196
Epoch - 124
TRAIN - loss_G: 3.7545, loss_G_identity: 0.6857, loss_G_GAN: 1.2043, loss_G_cycle: 1.864
6, loss_D: 0.1546
TEST -  loss_G: 4.9839, loss_G_identity: 1.3525, loss_G_GAN: 0.9067, loss_G_cycle: 2.724
7, loss_D: 0.3668
Epoch - 125
TRAIN - loss_G: 3.8826, loss_G_identity: 0.6896, loss_G_GAN: 1.2992, loss_G_cycle: 1.893
9, loss_D: 0.1651
TEST -  loss_G: 4.8675, loss_G_identity: 1.2524, loss_G_GAN: 0.8195, loss_G_cycle: 2.795
5, loss_D: 0.5678
Epoch - 126
TRAIN - loss_G: 3.6975, loss_G_identity: 0.6553, loss_G_GAN: 1.2244, loss_G_cycle: 1.817
7, loss_D: 0.1685
TEST -  loss_G: 5.5405, loss_G_identity: 1.3479, loss_G_GAN: 1.3119, loss_G_cycle: 2.880
7, loss_D: 0.5795
Epoch - 127
TRAIN - loss_G: 3.6787, loss_G_identity: 0.6594, loss_G_GAN: 1.2057, loss_G_cycle: 1.813
6, loss_D: 0.1783
TEST -  loss_G: 5.6165, loss_G_identity: 1.2716, loss_G_GAN: 1.6788, loss_G_cycle: 2.666
1, loss_D: 0.6037
Epoch - 128
TRAIN - loss_G: 3.6846, loss_G_identity: 0.6558, loss_G_GAN: 1.2224, loss_G_cycle: 1.806
3, loss_D: 0.1579
TEST -  loss_G: 4.7791, loss_G_identity: 1.2211, loss_G_GAN: 1.0564, loss_G_cycle: 2.501
6, loss_D: 0.4375
Epoch - 129
TRAIN - loss_G: 3.7521, loss_G_identity: 0.6593, loss_G_GAN: 1.2466, loss_G_cycle: 1.846
3, loss_D: 0.1524
TEST -  loss_G: 5.0744, loss_G_identity: 1.1866, loss_G_GAN: 1.3530, loss_G_cycle: 2.534
8, loss_D: 0.5097
Epoch - 130
TRAIN - loss_G: 3.7740, loss_G_identity: 0.6766, loss_G_GAN: 1.2471, loss_G_cycle: 1.850
3, loss_D: 0.1479
TEST -  loss_G: 5.4912, loss_G_identity: 1.2947, loss_G_GAN: 1.4860, loss_G_cycle: 2.710
5, loss_D: 0.6275
```


```
Epoch - 131
TRAIN - loss_G: 3.6973, loss_G_identity: 0.6533, loss_G_GAN: 1.2098, loss_G_cycle: 1.834
1, loss_D: 0.1494
TEST -  loss_G: 5.2058, loss_G_identity: 1.1548, loss_G_GAN: 1.5385, loss_G_cycle: 2.512
5, loss_D: 0.6413
Epoch - 132
TRAIN - loss_G: 3.5449, loss_G_identity: 0.6251, loss_G_GAN: 1.1907, loss_G_cycle: 1.729
1, loss_D: 0.1649
TEST -  loss_G: 4.8232, loss_G_identity: 1.2742, loss_G_GAN: 1.0205, loss_G_cycle: 2.528
5, loss_D: 0.4499
Epoch - 133
TRAIN - loss_G: 3.7338, loss_G_identity: 0.6400, loss_G_GAN: 1.2531, loss_G_cycle: 1.840
8, loss_D: 0.1516
TEST -  loss_G: 5.2220, loss_G_identity: 1.4020, loss_G_GAN: 1.1502, loss_G_cycle: 2.669
8, loss_D: 0.5295
Epoch - 134
TRAIN - loss_G: 3.6861, loss_G_identity: 0.6345, loss_G_GAN: 1.2538, loss_G_cycle: 1.797
8, loss_D: 0.1439
TEST -  loss_G: 5.3556, loss_G_identity: 1.3198, loss_G_GAN: 1.4848, loss_G_cycle: 2.551
0, loss_D: 0.6122
Epoch - 135
TRAIN - loss_G: 3.5221, loss_G_identity: 0.6144, loss_G_GAN: 1.1686, loss_G_cycle: 1.739
1, loss_D: 0.1526
TEST -  loss_G: 5.3741, loss_G_identity: 1.1534, loss_G_GAN: 1.6115, loss_G_cycle: 2.609
2, loss_D: 0.6216
Epoch - 136
TRAIN - loss_G: 3.5954, loss_G_identity: 0.6133, loss_G_GAN: 1.2328, loss_G_cycle: 1.749
3, loss_D: 0.1490
TEST -  loss_G: 5.7466, loss_G_identity: 1.4264, loss_G_GAN: 1.7736, loss_G_cycle: 2.546
5, loss_D: 0.5679
Epoch - 137
TRAIN - loss_G: 3.6120, loss_G_identity: 0.6106, loss_G_GAN: 1.2413, loss_G_cycle: 1.760
0, loss_D: 0.1490
TEST -  loss_G: 4.9577, loss_G_identity: 1.3077, loss_G_GAN: 0.9789, loss_G_cycle: 2.671
1, loss_D: 0.5521
Epoch - 138
TRAIN - loss_G: 3.4913, loss_G_identity: 0.6000, loss_G_GAN: 1.1679, loss_G_cycle: 1.723
4, loss_D: 0.1529
TEST -  loss_G: 5.2068, loss_G_identity: 1.2072, loss_G_GAN: 1.3940, loss_G_cycle: 2.605
7, loss_D: 0.5479
Epoch - 139
TRAIN - loss_G: 3.8846, loss_G_identity: 0.6679, loss_G_GAN: 1.3427, loss_G_cycle: 1.874
0, loss_D: 0.1376
TEST -  loss_G: 5.1811, loss_G_identity: 1.3992, loss_G_GAN: 1.0102, loss_G_cycle: 2.771
7, loss_D: 0.5197
Epoch - 140
TRAIN - loss_G: 3.6464, loss_G_identity: 0.6450, loss_G_GAN: 1.2202, loss_G_cycle: 1.781
2, loss_D: 0.1405
TEST -  loss_G: 5.4475, loss_G_identity: 1.4251, loss_G_GAN: 1.3711, loss_G_cycle: 2.651
3, loss_D: 0.6018
```


```
Epoch - 141
TRAIN - loss_G: 3.5241, loss_G_identity: 0.5947, loss_G_GAN: 1.2332, loss_G_cycle: 1.696
2, loss_D: 0.1432
TEST -  loss_G: 5.9132, loss_G_identity: 1.3971, loss_G_GAN: 1.8407, loss_G_cycle: 2.675
4, loss_D: 0.6127
Epoch - 142
TRAIN - loss_G: 3.4549, loss_G_identity: 0.5945, loss_G_GAN: 1.1937, loss_G_cycle: 1.666
8, loss_D: 0.1442
TEST -  loss_G: 4.7525, loss_G_identity: 1.3107, loss_G_GAN: 0.9076, loss_G_cycle: 2.534
2, loss_D: 0.5801
Epoch - 143
TRAIN - loss_G: 3.5721, loss_G_identity: 0.6203, loss_G_GAN: 1.2141, loss_G_cycle: 1.737
7, loss_D: 0.1494
TEST -  loss_G: 6.1862, loss_G_identity: 1.4128, loss_G_GAN: 1.9691, loss_G_cycle: 2.804
4, loss_D: 0.6832
Epoch - 144
TRAIN - loss_G: 3.6185, loss_G_identity: 0.6119, loss_G_GAN: 1.2601, loss_G_cycle: 1.746
4, loss_D: 0.1356
TEST -  loss_G: 5.7216, loss_G_identity: 1.3610, loss_G_GAN: 1.6876, loss_G_cycle: 2.673
0, loss_D: 0.6662
Epoch - 145
TRAIN - loss_G: 3.5824, loss_G_identity: 0.5970, loss_G_GAN: 1.2582, loss_G_cycle: 1.727
2, loss_D: 0.1458
TEST -  loss_G: 5.1706, loss_G_identity: 1.3519, loss_G_GAN: 1.3245, loss_G_cycle: 2.494
2, loss_D: 0.5421
Epoch - 146
TRAIN - loss_G: 3.6403, loss_G_identity: 0.6046, loss_G_GAN: 1.3033, loss_G_cycle: 1.732
5, loss_D: 0.1225
TEST -  loss_G: 5.7458, loss_G_identity: 1.4032, loss_G_GAN: 1.5199, loss_G_cycle: 2.822
7, loss_D: 0.6924
Epoch - 147
TRAIN - loss_G: 3.5084, loss_G_identity: 0.5864, loss_G_GAN: 1.2179, loss_G_cycle: 1.704
1, loss_D: 0.1352
TEST -  loss_G: 5.1479, loss_G_identity: 1.3005, loss_G_GAN: 1.2291, loss_G_cycle: 2.618
2, loss_D: 0.5323
Epoch - 148
TRAIN - loss_G: 3.5823, loss_G_identity: 0.5869, loss_G_GAN: 1.2967, loss_G_cycle: 1.698
6, loss_D: 0.1346
TEST -  loss_G: 5.2169, loss_G_identity: 1.2172, loss_G_GAN: 1.2917, loss_G_cycle: 2.708
0, loss_D: 0.4988
Epoch - 149
TRAIN - loss_G: 3.4620, loss_G_identity: 0.5703, loss_G_GAN: 1.2504, loss_G_cycle: 1.641
3, loss_D: 0.1402
TEST -  loss_G: 5.5098, loss_G_identity: 1.3209, loss_G_GAN: 1.5086, loss_G_cycle: 2.680
3, loss_D: 0.5375
Epoch - 150
TRAIN - loss_G: 3.5220, loss_G_identity: 0.5765, loss_G_GAN: 1.2748, loss_G_cycle: 1.670
7, loss_D: 0.1215
TEST -  loss_G: 5.7343, loss_G_identity: 1.4704, loss_G_GAN: 1.5521, loss_G_cycle: 2.711
8, loss_D: 0.7041
```


```
Epoch - 151
TRAIN - loss_G: 3.4336, loss_G_identity: 0.5629, loss_G_GAN: 1.2377, loss_G_cycle: 1.633
0, loss_D: 0.1266
TEST -  loss_G: 5.7265, loss_G_identity: 1.3133, loss_G_GAN: 1.7752, loss_G_cycle: 2.638
0, loss_D: 0.6415
Epoch - 152
TRAIN - loss_G: 3.4863, loss_G_identity: 0.5692, loss_G_GAN: 1.2703, loss_G_cycle: 1.646
8, loss_D: 0.1389
TEST -  loss_G: 5.3972, loss_G_identity: 1.2452, loss_G_GAN: 1.5824, loss_G_cycle: 2.569
7, loss_D: 0.6783
Epoch - 153
TRAIN - loss_G: 3.3827, loss_G_identity: 0.5530, loss_G_GAN: 1.2447, loss_G_cycle: 1.585
0, loss_D: 0.1337
TEST -  loss_G: 5.3470, loss_G_identity: 1.2157, loss_G_GAN: 1.4970, loss_G_cycle: 2.634
3, loss_D: 0.6310
Epoch - 154
TRAIN - loss_G: 3.4127, loss_G_identity: 0.5569, loss_G_GAN: 1.2576, loss_G_cycle: 1.598
1, loss_D: 0.1319
TEST -  loss_G: 5.3494, loss_G_identity: 1.2330, loss_G_GAN: 1.4903, loss_G_cycle: 2.626
1, loss_D: 0.5396
Epoch - 155
TRAIN - loss_G: 3.3351, loss_G_identity: 0.5402, loss_G_GAN: 1.2331, loss_G_cycle: 1.561
8, loss_D: 0.1392
TEST -  loss_G: 6.0231, loss_G_identity: 1.4028, loss_G_GAN: 1.9996, loss_G_cycle: 2.620
7, loss_D: 0.7277
Epoch - 156
TRAIN - loss_G: 3.4668, loss_G_identity: 0.5546, loss_G_GAN: 1.2952, loss_G_cycle: 1.617
1, loss_D: 0.1218
TEST -  loss_G: 5.5148, loss_G_identity: 1.1376, loss_G_GAN: 1.8253, loss_G_cycle: 2.551
9, loss_D: 0.7734
Epoch - 157
TRAIN - loss_G: 3.4504, loss_G_identity: 0.5461, loss_G_GAN: 1.2903, loss_G_cycle: 1.614
0, loss_D: 0.1202
TEST -  loss_G: 6.5578, loss_G_identity: 1.2791, loss_G_GAN: 2.7878, loss_G_cycle: 2.490
9, loss_D: 1.0446
Epoch - 158
TRAIN - loss_G: 3.3239, loss_G_identity: 0.5317, loss_G_GAN: 1.2609, loss_G_cycle: 1.531
3, loss_D: 0.1272
TEST -  loss_G: 5.1893, loss_G_identity: 1.3790, loss_G_GAN: 1.2626, loss_G_cycle: 2.547
8, loss_D: 0.5313
Epoch - 159
TRAIN - loss_G: 3.4264, loss_G_identity: 0.5501, loss_G_GAN: 1.2825, loss_G_cycle: 1.593
7, loss_D: 0.1214
TEST -  loss_G: 6.3071, loss_G_identity: 1.3286, loss_G_GAN: 2.3710, loss_G_cycle: 2.607
5, loss_D: 1.0578
Epoch - 160
TRAIN - loss_G: 3.4070, loss_G_identity: 0.5517, loss_G_GAN: 1.2573, loss_G_cycle: 1.598
0, loss_D: 0.1385
TEST -  loss_G: 5.1189, loss_G_identity: 1.1185, loss_G_GAN: 1.4889, loss_G_cycle: 2.511
5, loss_D: 0.6797
```


```
Epoch - 161
TRAIN - loss_G: 3.3336, loss_G_identity: 0.5371, loss_G_GAN: 1.2461, loss_G_cycle: 1.550
4, loss_D: 0.1274
TEST -  loss_G: 5.9531, loss_G_identity: 1.3375, loss_G_GAN: 2.0921, loss_G_cycle: 2.523
5, loss_D: 0.7853
Epoch - 162
TRAIN - loss_G: 3.4459, loss_G_identity: 0.5426, loss_G_GAN: 1.3085, loss_G_cycle: 1.594
8, loss_D: 0.1158
TEST -  loss_G: 5.5799, loss_G_identity: 1.3416, loss_G_GAN: 1.7051, loss_G_cycle: 2.533
2, loss_D: 0.7094
Epoch - 163
TRAIN - loss_G: 3.4015, loss_G_identity: 0.5500, loss_G_GAN: 1.2476, loss_G_cycle: 1.603
8, loss_D: 0.1313
TEST -  loss_G: 5.4508, loss_G_identity: 1.3519, loss_G_GAN: 1.8193, loss_G_cycle: 2.279
6, loss_D: 0.7063
Epoch - 164
TRAIN - loss_G: 3.3609, loss_G_identity: 0.5356, loss_G_GAN: 1.2657, loss_G_cycle: 1.559
6, loss_D: 0.1185
TEST -  loss_G: 5.4664, loss_G_identity: 1.3141, loss_G_GAN: 1.5559, loss_G_cycle: 2.596
4, loss_D: 0.6778
Epoch - 165
TRAIN - loss_G: 3.2727, loss_G_identity: 0.5088, loss_G_GAN: 1.2609, loss_G_cycle: 1.503
0, loss_D: 0.1254
TEST -  loss_G: 6.5059, loss_G_identity: 1.3468, loss_G_GAN: 2.6049, loss_G_cycle: 2.554
1, loss_D: 1.0740
Epoch - 166
TRAIN - loss_G: 3.2137, loss_G_identity: 0.5113, loss_G_GAN: 1.2145, loss_G_cycle: 1.487
9, loss_D: 0.1267
TEST -  loss_G: 5.7447, loss_G_identity: 1.3634, loss_G_GAN: 1.9250, loss_G_cycle: 2.456
3, loss_D: 0.8259
Epoch - 167
TRAIN - loss_G: 3.2768, loss_G_identity: 0.5169, loss_G_GAN: 1.2621, loss_G_cycle: 1.497
8, loss_D: 0.1232
TEST -  loss_G: 6.0365, loss_G_identity: 1.4156, loss_G_GAN: 2.1223, loss_G_cycle: 2.498
6, loss_D: 0.8408
Epoch - 168
TRAIN - loss_G: 3.2167, loss_G_identity: 0.4990, loss_G_GAN: 1.2574, loss_G_cycle: 1.460
3, loss_D: 0.1212
TEST -  loss_G: 6.0216, loss_G_identity: 1.3410, loss_G_GAN: 2.1426, loss_G_cycle: 2.538
0, loss_D: 0.8809
Epoch - 169
TRAIN - loss_G: 3.2674, loss_G_identity: 0.5104, loss_G_GAN: 1.2532, loss_G_cycle: 1.503
8, loss_D: 0.1207
TEST -  loss_G: 5.8097, loss_G_identity: 1.2947, loss_G_GAN: 1.9023, loss_G_cycle: 2.612
7, loss_D: 0.6650
Epoch - 170
TRAIN - loss_G: 3.1848, loss_G_identity: 0.4989, loss_G_GAN: 1.2245, loss_G_cycle: 1.461
3, loss_D: 0.1273
TEST -  loss_G: 6.1008, loss_G_identity: 1.3395, loss_G_GAN: 2.2136, loss_G_cycle: 2.547
7, loss_D: 0.9441
```


```
Epoch - 171
TRAIN - loss_G: 3.2489, loss_G_identity: 0.5072, loss_G_GAN: 1.2571, loss_G_cycle: 1.484
6, loss_D: 0.1252
TEST -  loss_G: 6.0689, loss_G_identity: 1.3643, loss_G_GAN: 1.9784, loss_G_cycle: 2.726
2, loss_D: 0.7349
Epoch - 172
TRAIN - loss_G: 3.1989, loss_G_identity: 0.4904, loss_G_GAN: 1.2678, loss_G_cycle: 1.440
7, loss_D: 0.1258
TEST -  loss_G: 5.6748, loss_G_identity: 1.2776, loss_G_GAN: 1.8311, loss_G_cycle: 2.566
0, loss_D: 0.6558
Epoch - 173
TRAIN - loss_G: 3.1875, loss_G_identity: 0.4877, loss_G_GAN: 1.2831, loss_G_cycle: 1.416
6, loss_D: 0.1175
TEST -  loss_G: 6.0085, loss_G_identity: 1.3837, loss_G_GAN: 2.2207, loss_G_cycle: 2.404
1, loss_D: 0.8645
Epoch - 174
TRAIN - loss_G: 3.1567, loss_G_identity: 0.4877, loss_G_GAN: 1.2694, loss_G_cycle: 1.399
7, loss_D: 0.1181
TEST -  loss_G: 5.7456, loss_G_identity: 1.3554, loss_G_GAN: 1.9320, loss_G_cycle: 2.458
3, loss_D: 0.8475
Epoch - 175
TRAIN - loss_G: 3.2648, loss_G_identity: 0.5026, loss_G_GAN: 1.2916, loss_G_cycle: 1.470
6, loss_D: 0.1150
TEST -  loss_G: 5.8001, loss_G_identity: 1.3414, loss_G_GAN: 1.9243, loss_G_cycle: 2.534
4, loss_D: 0.6942
Epoch - 176
TRAIN - loss_G: 3.1424, loss_G_identity: 0.4839, loss_G_GAN: 1.2416, loss_G_cycle: 1.416
9, loss_D: 0.1151
TEST -  loss_G: 6.0426, loss_G_identity: 1.2988, loss_G_GAN: 2.2916, loss_G_cycle: 2.452
2, loss_D: 0.9560
Epoch - 177
TRAIN - loss_G: 3.1756, loss_G_identity: 0.4869, loss_G_GAN: 1.2709, loss_G_cycle: 1.417
8, loss_D: 0.1170
TEST -  loss_G: 5.6142, loss_G_identity: 1.3651, loss_G_GAN: 1.8017, loss_G_cycle: 2.447
5, loss_D: 0.7091
Epoch - 178
TEST -  loss_G: 5.6163, loss_G_identity: 1.3453, loss_G_GAN: 1.9077, loss_G_cycle: 2.363
3, loss_D: 0.7745
Epoch - 179
TRAIN - loss_G: 3.1650, loss_G_identity: 0.4885, loss_G_GAN: 1.2572, loss_G_cycle: 1.419
3, loss_D: 0.1164
TEST -  loss_G: 6.6721, loss_G_identity: 1.3842, loss_G_GAN: 2.7548, loss_G_cycle: 2.533
1, loss_D: 1.1263
Epoch - 180
TRAIN - loss_G: 3.1297, loss_G_identity: 0.4784, loss_G_GAN: 1.2707, loss_G_cycle: 1.380
6, loss_D: 0.1240
TEST -  loss_G: 5.9882, loss_G_identity: 1.3045, loss_G_GAN: 2.3093, loss_G_cycle: 2.374
4, loss_D: 0.9739
```


```
Epoch - 181
TRAIN - loss_G: 3.1074, loss_G_identity: 0.4679, loss_G_GAN: 1.2819, loss_G_cycle: 1.357
6, loss_D: 0.1136
TEST -  loss_G: 6.1308, loss_G_identity: 1.3982, loss_G_GAN: 2.2058, loss_G_cycle: 2.526
8, loss_D: 0.8828
Epoch - 182
TRAIN - loss_G: 3.0807, loss_G_identity: 0.4823, loss_G_GAN: 1.2144, loss_G_cycle: 1.384
0, loss_D: 0.1236
TEST -  loss_G: 6.1137, loss_G_identity: 1.3995, loss_G_GAN: 2.2294, loss_G_cycle: 2.484
8, loss_D: 0.9260
Epoch - 183
TRAIN - loss_G: 3.1215, loss_G_identity: 0.4739, loss_G_GAN: 1.2772, loss_G_cycle: 1.370
3, loss_D: 0.1125
TEST -  loss_G: 6.1913, loss_G_identity: 1.3664, loss_G_GAN: 2.4292, loss_G_cycle: 2.395
7, loss_D: 0.9701
Epoch - 184
TRAIN - loss_G: 3.1160, loss_G_identity: 0.4789, loss_G_GAN: 1.2564, loss_G_cycle: 1.380
8, loss_D: 0.1186
TEST -  loss_G: 6.3466, loss_G_identity: 1.4381, loss_G_GAN: 2.3110, loss_G_cycle: 2.597
5, loss_D: 0.9129
Epoch - 185
TRAIN - loss_G: 3.1366, loss_G_identity: 0.4836, loss_G_GAN: 1.2659, loss_G_cycle: 1.387
2, loss_D: 0.1155
TEST -  loss_G: 6.4126, loss_G_identity: 1.4092, loss_G_GAN: 2.5778, loss_G_cycle: 2.425
5, loss_D: 1.0102
Epoch - 186
TRAIN - loss_G: 3.0221, loss_G_identity: 0.4558, loss_G_GAN: 1.2416, loss_G_cycle: 1.324
7, loss_D: 0.1157
TEST -  loss_G: 6.3792, loss_G_identity: 1.3590, loss_G_GAN: 2.5410, loss_G_cycle: 2.479
2, loss_D: 0.9556
Epoch - 187
TRAIN - loss_G: 3.0430, loss_G_identity: 0.4610, loss_G_GAN: 1.2477, loss_G_cycle: 1.334
3, loss_D: 0.1192
TEST -  loss_G: 6.4156, loss_G_identity: 1.4161, loss_G_GAN: 2.4042, loss_G_cycle: 2.595
3, loss_D: 0.9154
Epoch - 188
TRAIN - loss_G: 3.0908, loss_G_identity: 0.4679, loss_G_GAN: 1.2729, loss_G_cycle: 1.349
9, loss_D: 0.1151
TEST -  loss_G: 6.5094, loss_G_identity: 1.3948, loss_G_GAN: 2.7348, loss_G_cycle: 2.379
7, loss_D: 0.9974
Epoch - 189
TRAIN - loss_G: 3.0246, loss_G_identity: 0.4574, loss_G_GAN: 1.2477, loss_G_cycle: 1.319
5, loss_D: 0.1178
TEST -  loss_G: 6.5977, loss_G_identity: 1.3709, loss_G_GAN: 2.8243, loss_G_cycle: 2.402
5, loss_D: 1.0940
Epoch - 190
TRAIN - loss_G: 3.0237, loss_G_identity: 0.4668, loss_G_GAN: 1.2361, loss_G_cycle: 1.320
9, loss_D: 0.1208
TEST -  loss_G: 6.7720, loss_G_identity: 1.4155, loss_G_GAN: 2.8745, loss_G_cycle: 2.481
9, loss_D: 1.0811
```


```
Epoch - 191
TRAIN - loss_G: 3.0043, loss_G_identity: 0.4515, loss_G_GAN: 1.2481, loss_G_cycle: 1.304
7, loss_D: 0.1175
TEST -  loss_G: 6.4363, loss_G_identity: 1.3850, loss_G_GAN: 2.5171, loss_G_cycle: 2.534
2, loss_D: 0.9726
Epoch - 192
TRAIN - loss_G: 3.0556, loss_G_identity: 0.4619, loss_G_GAN: 1.2592, loss_G_cycle: 1.334
6, loss_D: 0.1131
TEST -  loss_G: 6.5350, loss_G_identity: 1.3678, loss_G_GAN: 2.7348, loss_G_cycle: 2.432
4, loss_D: 1.0100
Epoch - 193
TRAIN - loss_G: 3.0250, loss_G_identity: 0.4600, loss_G_GAN: 1.2502, loss_G_cycle: 1.314
9, loss_D: 0.1179
TEST -  loss_G: 6.6796, loss_G_identity: 1.3950, loss_G_GAN: 2.6935, loss_G_cycle: 2.591
1, loss_D: 1.0086
Epoch - 194
TRAIN - loss_G: 3.0320, loss_G_identity: 0.4597, loss_G_GAN: 1.2613, loss_G_cycle: 1.311
0, loss_D: 0.1153
TEST -  loss_G: 6.5987, loss_G_identity: 1.3310, loss_G_GAN: 2.8825, loss_G_cycle: 2.385
2, loss_D: 1.1444
Epoch - 195
TRAIN - loss_G: 3.0200, loss_G_identity: 0.4620, loss_G_GAN: 1.2611, loss_G_cycle: 1.296
9, loss_D: 0.1228
TEST -  loss_G: 6.7293, loss_G_identity: 1.4108, loss_G_GAN: 2.8208, loss_G_cycle: 2.497
7, loss_D: 1.0626
Epoch - 196
TRAIN - loss_G: 2.9829, loss_G_identity: 0.4546, loss_G_GAN: 1.2354, loss_G_cycle: 1.292
9, loss_D: 0.1189
TEST -  loss_G: 6.7264, loss_G_identity: 1.4216, loss_G_GAN: 2.8497, loss_G_cycle: 2.455
2, loss_D: 1.1339
Epoch - 197
TRAIN - loss_G: 3.0131, loss_G_identity: 0.4593, loss_G_GAN: 1.2349, loss_G_cycle: 1.318
9, loss_D: 0.1214
TEST -  loss_G: 6.7815, loss_G_identity: 1.4077, loss_G_GAN: 2.8630, loss_G_cycle: 2.510
8, loss_D: 1.0751
Epoch - 198
TRAIN - loss_G: 2.9551, loss_G_identity: 0.4416, loss_G_GAN: 1.2347, loss_G_cycle: 1.278
8, loss_D: 0.1274
TEST -  loss_G: 6.6883, loss_G_identity: 1.3375, loss_G_GAN: 2.9274, loss_G_cycle: 2.423
4, loss_D: 1.1073
Epoch - 199
TRAIN - loss_G: 3.0423, loss_G_identity: 0.4654, loss_G_GAN: 1.2638, loss_G_cycle: 1.313
1, loss_D: 0.1166
TEST -  loss_G: 6.6826, loss_G_identity: 1.3443, loss_G_GAN: 2.9370, loss_G_cycle: 2.401
3, loss_D: 1.1231
Epoch - 200
TRAIN - loss_G: 3.0070, loss_G_identity: 0.4539, loss_G_GAN: 1.2645, loss_G_cycle: 1.288
6, loss_D: 0.1166
TEST -  loss_G: 6.6944, loss_G_identity: 1.3834, loss_G_GAN: 2.9248, loss_G_cycle: 2.386
3, loss_D: 1.1622
```


```
CPU times: user 1h 46min 30s, sys: 1h 8min 19s, total: 2h 54min 50s
Wall time: 2h 58min 9s
```

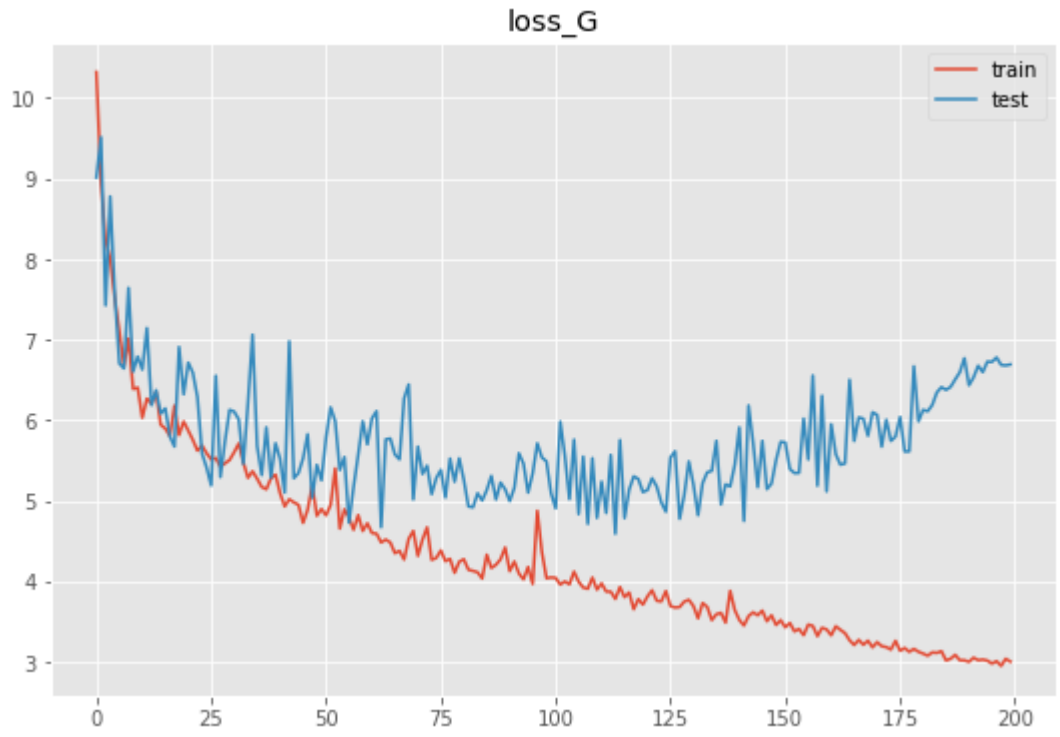
Losses

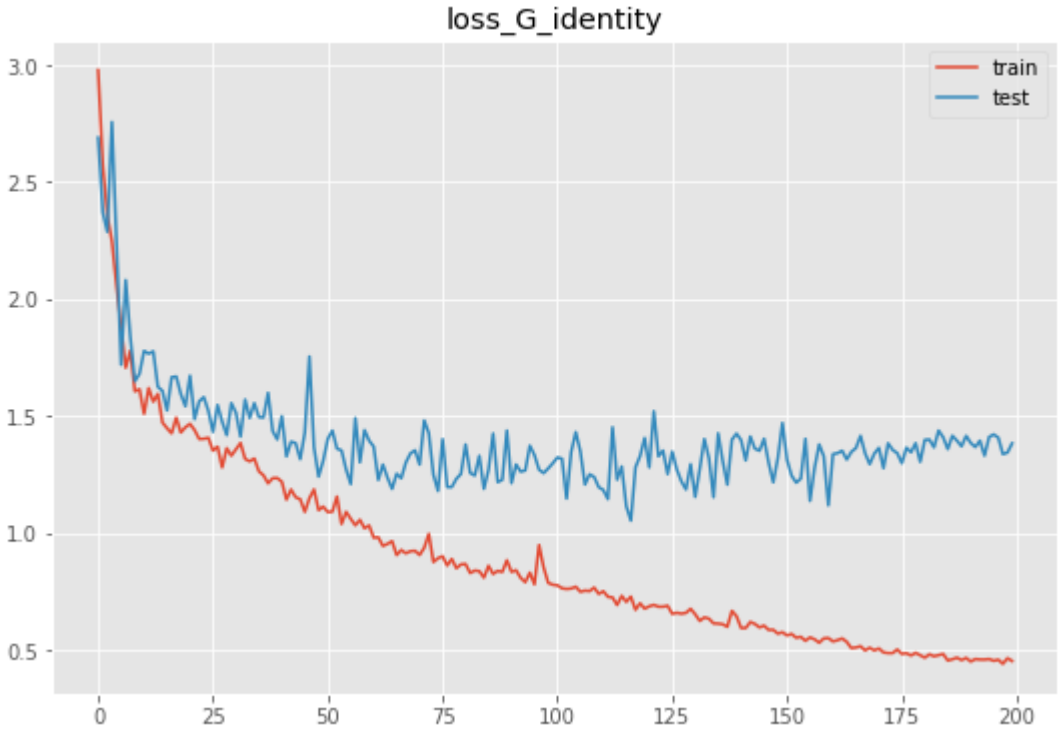
Посмотрим, как выглядят трейн и тест лоссы на обучении

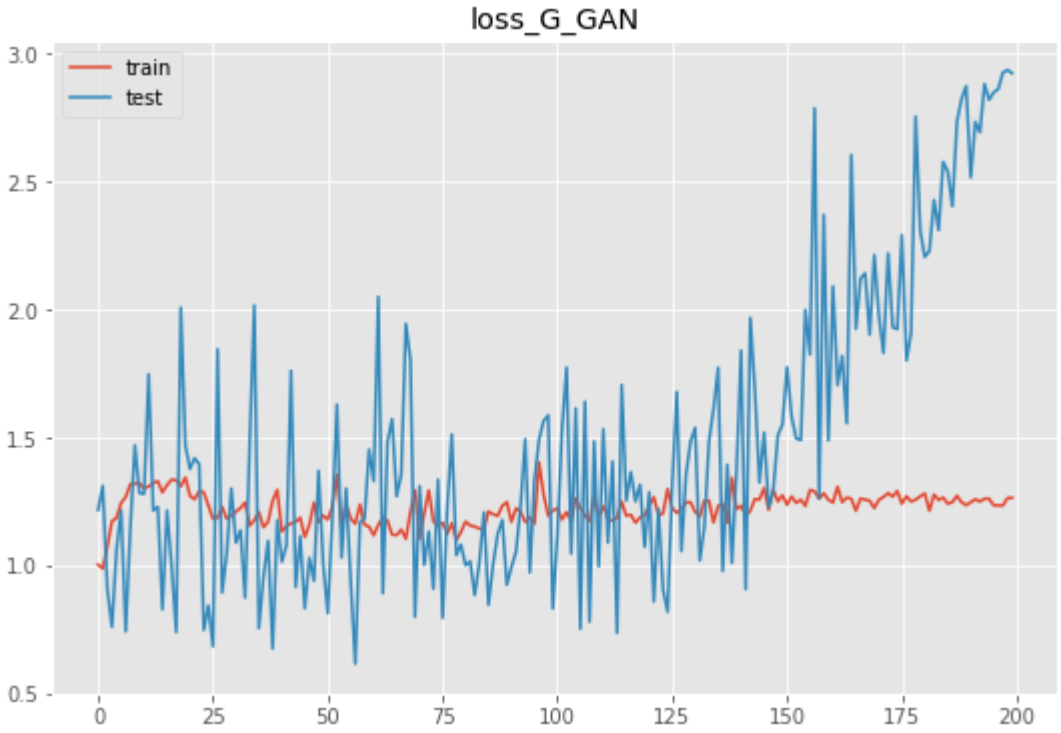
In [ ]:
```
logs = {}
for mode in ["train", "test"]:
    with open(f"output/{output_name}/{mode}_log.json", "r") as file:
        logs[mode] = json.load(file)
```

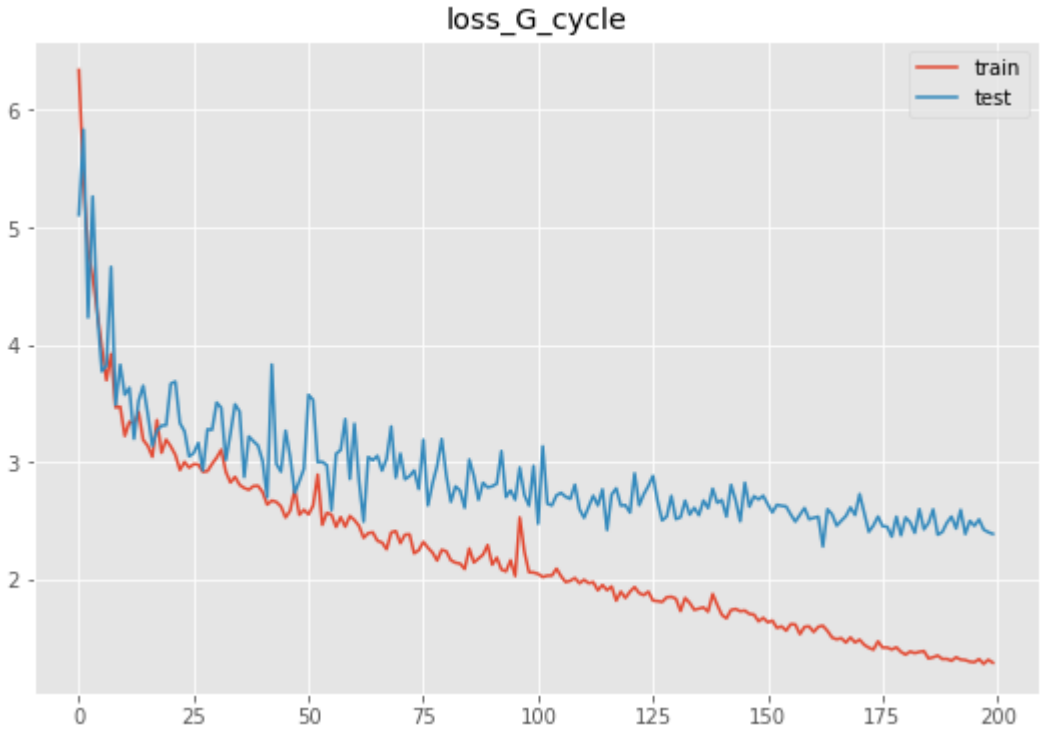
In [ ]:
```
def plot_logs(train_log: dict, test_log: dict):
    for loss_name in train_log:
        plt.style.use('ggplot')
        plt.figure(figsize=(9, 6))
        plt.plot(train_log[loss_name], label="train")
        plt.plot(test_log[loss_name], label="test")
        plt.title(loss_name)
        plt.legend()
        plt.show()
```

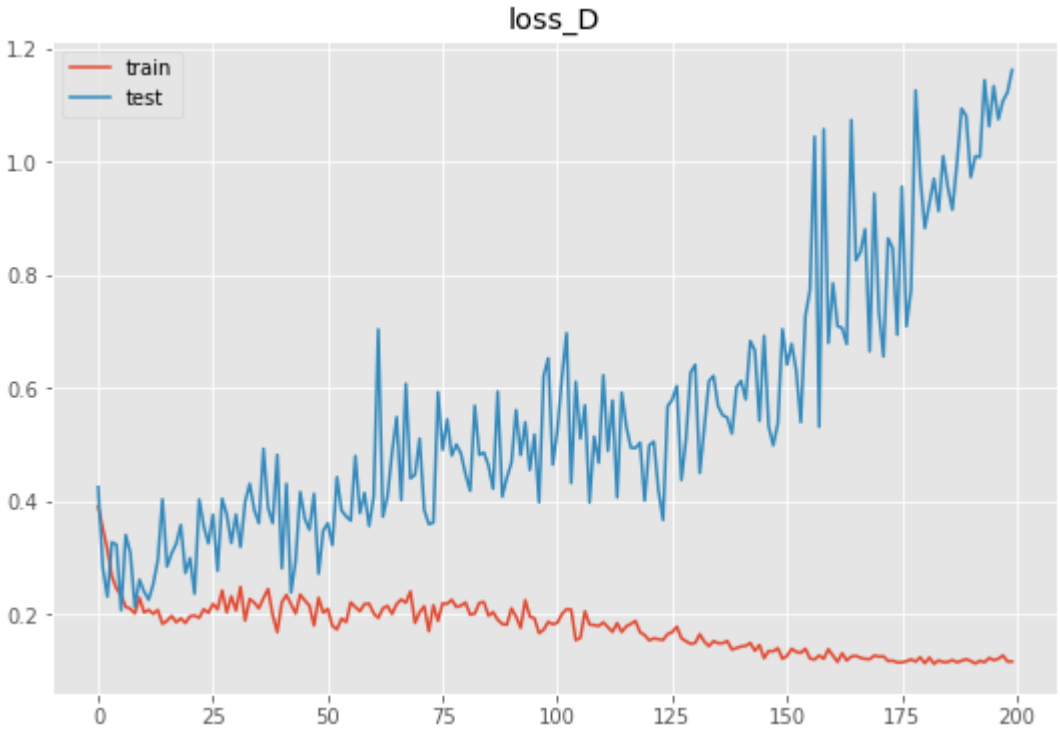
In [ ]:
```
plot_logs(logs["train"], logs["test"])
```




loss_G_identity
train
test
3.0
2.5
2.0
1.5
1.0
0.5
0
25
50
75
100
125
150
175
200


loss_G_GAN
train
test
3.0
2.5
2.0
1.5
1.0
0.5
0
25
50
75
100
125
150
175
200





Применение (тестирование) обученных и сохраненных генераторов

инициализация параметров для теста

In [ ]:
```
# test params init
batchSize = 1  # size of the batches
dataroot = f'datasets/{dataset_name}/'  # root directory of the dataset
input_nc = 3  # number of channels of input data
output_nc = 3  # number of channels of output data
```

```
n_cpu = 8  # number of cpu threads to use during batch generation
device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
```

инициализация моделей генераторов и загрузчика данных

In [ ]:
```
netG_A2B = Generator(input_nc, output_nc).to(device)
netG_B2A = Generator(output_nc, input_nc).to(device)

netG_A2B_path = f'output/{output_name}/best/netG_A2B.pth'
netG_B2A_path = f'output/{output_name}/best/netG_B2A.pth'

# load state dicts
netG_A2B.load_state_dict(torch.load(netG_A2B_path, map_location=device))
netG_B2A.load_state_dict(torch.load(netG_B2A_path, map_location=device))

# set test mode
netG_A2B.eval()
netG_B2A.eval()

# Dataset loader
test_transforms = [transforms.ToTensor(),
                   transforms.Normalize((0.5,0.5,0.5), (0.5,0.5,0.5))]
test_dataloader = DataLoader(ImageDataset(dataroot, transforms_=test_transforms, mode='
                        batch_size=batchSize, shuffle=False, num_workers=n_cpu)
```

применяем генераторы к тестовой выборке и сохраняем результат (вместе с оригиналами)

In [ ]:
```
# Create output dirs if they don't exist
if not os.path.exists(f'output/{output_name}/A'):
    os.makedirs(f'output/{output_name}/A')
if not os.path.exists(f'output/{output_name}/B'):
    os.makedirs(f'output/{output_name}/B')

for i, batch in enumerate(test_dataloader):
    # Set model input
    real_A = batch['A']
    real_B = batch['B']

    real_A_img = 0.5* (real_A.data + 1.0)
    real_B_img = 0.5* (real_B.data + 1.0)

    # Generate output
    fake_A2B_img = 0.5*(netG_A2B(real_A.to(device)).data + 1.0)
    fake_B2A_img = 0.5*(netG_B2A(real_B.to(device)).data + 1.0)

    # Save image files
    save_image(real_A_img, f'output/{output_name}/A/{i}_realA.png')
    save_image(real_B_img, f'output/{output_name}/B/{i}_realB.png')
    save_image(fake_A2B_img, f'output/{output_name}/A/{i}_fakeA2B.png')
    save_image(fake_B2A_img, f'output/{output_name}/B/{i}_fakeB2A.png')
```

Визуализация результата

In [ ]:
```
class PlotResult:
    def __init__(self, mode="A2B", root=f"output/{output_name}"):
        self.root = root
        self.mode = mode

    def plot(self, num=4, figsize=(24, 12)):
```

```
        real_liter = self.mode[0]
        self.real_files = sorted(glob.glob(os.path.join(self.root, real_liter, f'*_real
        self.fake_files = sorted(glob.glob(os.path.join(self.root, real_liter, f'*_fake

        inds = random.choices(list(range(len(self.real_files))), k=num)

        plt.figure(figsize=figsize)
        for i, ind in enumerate(inds):
            plt.subplot(2, num, i+1)
            plt.axis("off")
            plt.imshow(Image.open(self.real_files[ind]))
            plt.title(f"real{real_liter}")

            plt.subplot(2, num, num+i+1)
            plt.axis("off")
            plt.imshow(Image.open(self.fake_files[ind]))
            plt.title(f"fake{self.mode}")
        plt.show()
```

Визуализация результата **A2B**

(несколько случайных вариантов)

In [ ]:
```
PlotResult("A2B").plot()
```

Визуализация результата **B2A**

(несколько случайных вариантов)

In [ ]:
```
PlotResult("B2A").plot()
```

Некоторые выводы

- видно, что модель обучается
- примерно со 125 эпохи наблюдается переобучение (для применения каждый раз сохранялись лучшие генераторы по критерию loss_G -> min)
- думаю, переобучение связано с недостаточным объемом датасета. В трейне было 130 изображений, в тесте - 20
- малый объем датасета связан с тем, что пришлось делать его вручную. Изображения были просто скачены руками из яндекс-картинок
- думаю, можно было бы результат улучшить, в частности, путем увеличения датасета