In [1]:
```
import os
import random
import itertools
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from tqdm import tqdm
import torchvision


import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import torchvision.transforms as T
from torchvision.utils import save_image
from torchvision import transforms
```

In [2]:
```
class ResidualBlock(nn.Module):
    def __init__(self, channels):
        super().__init__()
        self.block = nn.Sequential(
            nn.ReflectionPad2d(1),
            nn.Conv2d(channels, channels, kernel_size=3),
            nn.InstanceNorm2d(channels),
            nn.ReLU(inplace=True),
            nn.ReflectionPad2d(1),
            nn.Conv2d(channels, channels, kernel_size=3),
            nn.InstanceNorm2d(channels),
        )

    def forward(self, x):
        return x + self.block(x)
```

In [3]:
```
class Generator(nn.Module):
    def __init__(self, in_channels=3, out_channels=3, n_residual_blocks=9):
        super().__init__()

        model = [
            nn.ReflectionPad2d(3),
            nn.Conv2d(in_channels, 64, kernel_size=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, kernel_size=3, stride=2, paddi
                nn.InstanceNorm2d(out_features),
                nn.ReLU(inplace=True),
            ]
            in_features = out_features
            out_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, kernel_size=3, stride
                nn.InstanceNorm2d(out_features),
                nn.ReLU(inplace=True),
            ]
            in_features = out_features
            out_features //= 2

        model += [
            nn.ReflectionPad2d(3),
            nn.Conv2d(64, out_channels, kernel_size=7),
            nn.Tanh()
        ]

        self.model = nn.Sequential(*model)

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

In [4]:
```
# Assuming you have a Generator class defined
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
# Load the pre-trained generator model
G_AB = Generator().to(device)
G_AB.load_state_dict(torch.load("G_BA_monet.pth", map_location=device))
G_AB.eval()  # Set model to evaluation mode

transform = T.Compose([
    T.Resize(256),
    T.ToTensor(),
    T.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
```

In [7]:
```
input_folder = "datasets/real_images/photo_jpg"
output_folder = "outputs/monet_style"
os.makedirs(output_folder, exist_ok=True)
```

In [8]:
```
for filename in os.listdir(input_folder):
    if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
        img_path = os.path.join(input_folder, filename)
        img = Image.open(img_path).convert('RGB')
        img_tensor = transform(img).unsqueeze(0).to(device)

        with torch.no_grad():
            fake_img = G_AB(img_tensor)
```

```python
        # Denormalize for saving
        fake_img = 0.5 * (fake_img + 1.0)
        save_path = os.path.join(output_folder, filename)
        save_image(fake_img, save_path)
        print(f"Saved stylized image to: {save_path}")
```

```
Saved stylized image to: outputs/monet_style\00068bc07f.jpg
Saved stylized image to: outputs/monet_style\000910d219.jpg
Saved stylized image to: outputs/monet_style\000ded5c41.jpg
Saved stylized image to: outputs/monet_style\00104fd531.jpg
Saved stylized image to: outputs/monet_style\001158d595.jpg
Saved stylized image to: outputs/monet_style\0033c5f971.jpg
Saved stylized image to: outputs/monet_style\0039ebb598.jpg
Saved stylized image to: outputs/monet_style\003aab6fdd.jpg
Saved stylized image to: outputs/monet_style\003c6c30e0.jpg
Saved stylized image to: outputs/monet_style\00479e2a21.jpg
Saved stylized image to: outputs/monet_style\005f987f56.jpg
Saved stylized image to: outputs/monet_style\0080f94ebc.jpg
Saved stylized image to: outputs/monet_style\00882b7e1d.jpg
Saved stylized image to: outputs/monet_style\009d534136.jpg
Saved stylized image to: outputs/monet_style\009ddaed1f.jpg
Saved stylized image to: outputs/monet_style\00aeb60e25.jpg
Saved stylized image to: outputs/monet_style\00c6a0ad1e.jpg
Saved stylized image to: outputs/monet_style\00dcf0f1e3.jpg
Saved stylized image to: outputs/monet_style\00dff09ebe.jpg
Saved stylized image to: outputs/monet_style\00e1798585.jpg
Saved stylized image to: outputs/monet_style\00e64e1b2c.jpg
Saved stylized image to: outputs/monet_style\00f78547f0.jpg
Saved stylized image to: outputs/monet_style\00fcff630e.jpg
Saved stylized image to: outputs/monet_style\01135e4771.jpg
Saved stylized image to: outputs/monet_style\011f46de73.jpg
Saved stylized image to: outputs/monet_style\012df5ce29.jpg
Saved stylized image to: outputs/monet_style\01416e7ada.jpg
Saved stylized image to: outputs/monet_style\01480da48f.jpg
Saved stylized image to: outputs/monet_style\0159685c51.jpg
Saved stylized image to: outputs/monet_style\015f79b672.jpg
Saved stylized image to: outputs/monet_style\01622039ef.jpg
Saved stylized image to: outputs/monet_style\0162322d2d.jpg
Saved stylized image to: outputs/monet_style\016d408833.jpg
Saved stylized image to: outputs/monet_style\016f524878.jpg
Saved stylized image to: outputs/monet_style\017018f377.jpg
Saved stylized image to: outputs/monet_style\01771d99c3.jpg
Saved stylized image to: outputs/monet_style\017aacad6e.jpg
Saved stylized image to: outputs/monet_style\018be60883.jpg
Saved stylized image to: outputs/monet_style\018db0f253.jpg
Saved stylized image to: outputs/monet_style\01952c7673.jpg
Saved stylized image to: outputs/monet_style\01971e48e5.jpg
Saved stylized image to: outputs/monet_style\019cf7682d.jpg
Saved stylized image to: outputs/monet_style\01ae8be57e.jpg
Saved stylized image to: outputs/monet_style\01af5f8623.jpg
Saved stylized image to: outputs/monet_style\01bce47d3f.jpg
Saved stylized image to: outputs/monet_style\01c4f86e3e.jpg
Saved stylized image to: outputs/monet_style\01f7a6398e.jpg
Saved stylized image to: outputs/monet_style\02001e59af.jpg
Saved stylized image to: outputs/monet_style\0212257854.jpg
Saved stylized image to: outputs/monet_style\021df2c226.jpg
Saved stylized image to: outputs/monet_style\022ca16b0f.jpg
Saved stylized image to: outputs/monet_style\023637f8fb.jpg
Saved stylized image to: outputs/monet_style\023eed0385.jpg
Saved stylized image to: outputs/monet_style\023f3cbb75.jpg
Saved stylized image to: outputs/monet_style\0240a597f5.jpg
Saved stylized image to: outputs/monet_style\024d98032b.jpg
Saved stylized image to: outputs/monet_style\fea227771f.jpg
Saved stylized image to: outputs/monet_style\fea42e4505.jpg
Saved stylized image to: outputs/monet_style\fea4d4f76c.jpg
Saved stylized image to: outputs/monet_style\fea5afe679.jpg
Saved stylized image to: outputs/monet_style\feb132e4c1.jpg
Saved stylized image to: outputs/monet_style\fee0c39907.jpg
Saved stylized image to: outputs/monet_style\fefb86c469.jpg
Saved stylized image to: outputs/monet_style\fefcd07a5e.jpg
Saved stylized image to: outputs/monet_style\ff03c98f0e.jpg
Saved stylized image to: outputs/monet_style\ff2152ac2a.jpg
Saved stylized image to: outputs/monet_style\ff258e79ce.jpg
Saved stylized image to: outputs/monet_style\ff2cb6658c.jpg
Saved stylized image to: outputs/monet_style\ff2f57429f.jpg
Saved stylized image to: outputs/monet_style\ff3867d938.jpg
Saved stylized image to: outputs/monet_style\ff46474c6d.jpg
Saved stylized image to: outputs/monet_style\ff48e8580b.jpg
Saved stylized image to: outputs/monet_style\ff58af0f3b.jpg
Saved stylized image to: outputs/monet_style\ff5c15ab50.jpg
Saved stylized image to: outputs/monet_style\ff6ed45562.jpg
Saved stylized image to: outputs/monet_style\ff757d7db3.jpg
Saved stylized image to: outputs/monet_style\ff769c35c9.jpg
Saved stylized image to: outputs/monet_style\ff7d83bc1d.jpg
Saved stylized image to: outputs/monet_style\ff853f993b.jpg
Saved stylized image to: outputs/monet_style\ff9ca0e1bb.jpg
Saved stylized image to: outputs/monet_style\ff9e94857b.jpg
Saved stylized image to: outputs/monet_style\ffa1ee4875.jpg
Saved stylized image to: outputs/monet_style\ffa5d376be.jpg
Saved stylized image to: outputs/monet_style\ffbbb24b43.jpg
Saved stylized image to: outputs/monet_style\ffbdd43bfb.jpg
Saved stylized image to: outputs/monet_style\ffc5b52a77.jpg
Saved stylized image to: outputs/monet_style\ffcc20463a.jpg
Saved stylized image to: outputs/monet_style\ffcdd249dd.jpg
Saved stylized image to: outputs/monet_style\ffcf64f150.jpg
Saved stylized image to: outputs/monet_style\ffd71fce61.jpg
Saved stylized image to: outputs/monet_style\ffe1af8ca0.jpg
Saved stylized image to: outputs/monet_style\fff5c33050.jpg
Saved stylized image to: outputs/monet_style\fffaaaae65.jpg
Saved stylized image to: outputs/monet_style\fffc0836d7.jpg
```

In [10]:

```python
from torch.utils.data import Dataset
from PIL import Image
import os

class FlatFolderDataset(Dataset):
    def __init__(self, folder, transform=None):
        self.files = sorted([
            os.path.join(folder, f) for f in os.listdir(folder)
            if f.lower().endswith(('.jpg', '.jpeg', '.png'))
        ])
        self.transform = transform

    def __len__(self):
        return len(self.files)

    def __getitem__(self, idx):
        img = Image.open(self.files[idx]).convert('RGB')
        if self.transform:
            img = self.transform(img)
        return img, 0  # dummy label
```

In [13]:
```
import torch
from torch.utils.data import DataLoader
from torchmetrics.image.fid import FrechetInceptionDistance
import torchvision.transforms as T

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

transform = T.Compose([
    T.Resize((256, 256)),  # match InceptionNet input
    T.PILToTensor()
])

real_dataset = FlatFolderDataset("datasets/real_images/photo_jpg", transform)
fake_dataset = FlatFolderDataset("outputs/monet_style", transform)

real_loader = DataLoader(real_dataset, batch_size=16, shuffle=False, num_workers=0)
fake_loader = DataLoader(fake_dataset, batch_size=16, shuffle=False, num_workers=0)

fid = FrechetInceptionDistance(feature=2048).to(device)

# Feed real images
for batch, _ in real_loader:
    fid.update(batch.to(device), real=True)

# Feed fake images
for batch, _ in fake_loader:
    fid.update(batch.to(device), real=False)

# Compute FID score
fid_score = fid.compute().item()
print(f"FID Score: {fid_score:.4f}")
```

```
FID Score: 46.7764
```