Code:

```
import torch
import torchvision.transforms as transforms
from torchvision.utils import save_image
from torch.utils.data import DataLoader
from torchvision.datasets import CelebA
from torch.autograd import Variable
import numpy as np
import os
import imageio

# Mount Google Drive to access files
from google.colab import drive
drive.mount('/content/drive')

# Device configuration
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Hyperparameters
latent_size = 100
batch_size = 64
num_epochs = 10
sample_dir = '/content/drive/My Drive/generated_images'
video_file = '/content/drive/My Drive/generated_video.mp4'

# Create directories if not exists
if not os.path.exists(sample_dir):
    os.makedirs(sample_dir)

# CelebA dataset
transform = transforms.Compose([
    transforms.Resize((64, 64)),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

celeba_dataset = CelebA(root='/content/data/',
                        split='all',
                        transform=transform,
                        download=True)

# Data loader
data_loader = DataLoader(dataset=celeba_dataset,
                         batch_size=batch_size,
```

```
                    shuffle=True)

# Generator model
class Generator(torch.nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.main = torch.nn.Sequential(
            torch.nn.ConvTranspose2d(latent_size, 512, 4, 1, 0, bias=False),
            torch.nn.BatchNorm2d(512),
            torch.nn.ReLU(True),
            torch.nn.ConvTranspose2d(512, 256, 4, 2, 1, bias=False),
            torch.nn.BatchNorm2d(256),
            torch.nn.ReLU(True),
            torch.nn.ConvTranspose2d(256, 128, 4, 2, 1, bias=False),
            torch.nn.BatchNorm2d(128),
            torch.nn.ReLU(True),
            torch.nn.ConvTranspose2d(128, 64, 4, 2, 1, bias=False),
            torch.nn.BatchNorm2d(64),
            torch.nn.ReLU(True),
            torch.nn.ConvTranspose2d(64, 3, 4, 2, 1, bias=False),
            torch.nn.Tanh()
        )

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

# Create generator
generator = Generator().to(device)

# Load pretrained model
generator.load_state_dict(torch.load('/content/drive/My Drive/generator.pth',
map_location=torch.device(device)))
generator.eval()

# Generate video frames
images = []
with torch.no_grad():
    for i in range(100):  # Generate 100 frames
        noise = Variable(torch.randn(batch_size, latent_size, 1, 1)).to(device)
        fake_images = generator(noise)
        images.append(fake_images)

# Combine images into video
with imageio.get_writer(video_file, mode='I') as writer:
```

```
    for image in images:
        for img in image:
            img = img.cpu().numpy()
            img = np.transpose(img, (1, 2, 0))
            img = ((img + 1) * 255 / 2).astype(np.uint8)
            writer.append_data(img)

print("Video generation completed!")
```

Output: