Assignment

July 25, 2023

1 DT8122 - Assignment

Deadline: 2023 August 15 AoE (Anywhere on Earth)

Send a zip file with the notebook both as a .ipynb and as a .pdf file to dt8122@idi.ntnu.no. Label the file with your full name.

The task is to implement conditional DDPM for MNIST images. Your implementation should take as input a digit and be able to generate 28x28 grayscale handwritten image of said digit.

You can add additional cells anywhere in the notebook to make your code more readable.

DDPM: https://arxiv.org/abs/2006.11239

Classifier-free conditional DDPM: https://arxiv.org/abs/2207.12598

The notebook should be run when it is turned in so all plots are visible. All code should be contained in the notebook.

1.0.1 Install necessary libraries

Any additional libraries you make use of should be installed in this cell.

Instructions to run this code: Create a new environment: conda create -n probai python=3.10

Then run this notebook in the environment. You might need to install ipykernel to run jupyter notebook.

[14]:
```
%pip install torch torchvision torchaudio --index-url https://download.pytorch.
↪org/whl/cu117
%pip install matplotlib
%pip install tqdm
%pip install ipykernel
%pip install pandoc
%pip install nbconvert
```

[15]:
```
import os
import random
import math
import numpy as np

import matplotlib.pyplot as plt
```

```
from tqdm import tqdm

import torch
import torch.nn as nn
from torch.utils.data import DataLoader
from torchvision.utils import save_image, make_grid
from torchvision.transforms import Compose, ToTensor, Lambda
from torchvision.datasets.mnist import MNIST
from matplotlib.animation import FuncAnimation, PillowWriter
```

1.0.2 Define constants here

Constants such as number of epochs, device, and learning rate and other hyperparameters should be defined here.

[16]:
```
SEED = 0
random.seed(SEED)
torch.manual_seed(SEED)
np.random.seed(SEED)


n_epoch = 25
batch_size = 256
n_T = 1000
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
n_classes = 10
n_feat = 128 #256 better but slower
lrate = 1e-4
save_model = False
save_dir = './data/diffusion/'
ws_test = [0.0, 0.5, 2.0] # strength of generative guidance
beta_start = 0.0001
beta_end = 0.02

os.makedirs(save_dir, exist_ok=True)
```

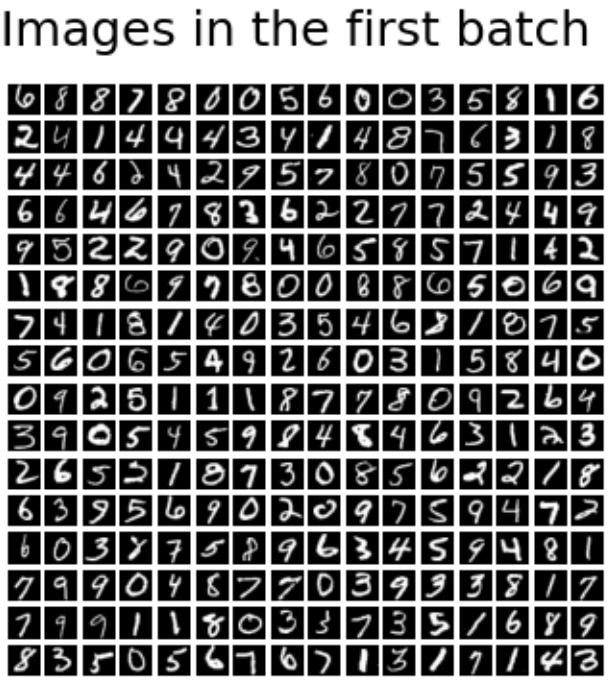
1.0.3 Functionality for loading and visualizing dataset

We have provided some functionality for loading and visualizing the dataset. You may add more cells/functions here.

[17]:
```
transform = Compose([
    ToTensor(),
    Lambda(lambda x: (x - 0.5) * 2)]
)

dataset = MNIST("./datasets", download=True, train=True, transform=transform)
dataloader = DataLoader(dataset, batch_size, shuffle=True)
```

[18]:
```
"""
This function plots images in a grid. Input is a Tensor.
See show_first_batch to see how it is used.
"""
def show_images(images, title=""):
    images = images.detach().cpu().numpy()
    fig = plt.figure(figsize=(4, 4))
    cols = math.ceil(len(images) ** (1 / 2))
    rows = math.ceil(len(images) / cols)
    for r in range(rows):
        for c in range(cols):
            idx = cols * r + c
            ax = fig.add_subplot(rows, cols, idx + 1)
            ax.axis('off')
            if idx < len(images):
                ax.imshow(images[idx][0], cmap="gray")
    fig.suptitle(title, fontsize=18)
    plt.show()
```

[19]:
```
def show_first_batch(dataloader):
    for batch in dataloader:
        show_images(batch[0], "Images in the first batch")
        break
show_first_batch(dataloader)
```



1.0.4 Your conditional DDPM implementation should go here

This includes functionality for adding noise to the image.

[20]:
```
def ddpm_schedules(beta1, beta2, T):

    beta_t = (beta2 - beta1) * torch.arange(0, T + 1, dtype=torch.float32) / T␣
 ↪+ beta1
    sqrt_beta_t = torch.sqrt(beta_t)
    alpha_t = 1 - beta_t
    alphabar_t = torch.cumprod(alpha_t, dim=0)

    sqrtab = torch.sqrt(alphabar_t)
    oneover_sqrta = 1 / torch.sqrt(alpha_t)

    sqrtmab = torch.sqrt(1 - alphabar_t)
    mab_over_sqrtmab_inv = (1 - alpha_t) / sqrtmab

    return {
        "alpha_t": alpha_t,  # \alpha_t
        "oneover_sqrta": oneover_sqrta,  # 1/\sqrt{\alpha_t}
        "sqrt_beta_t": sqrt_beta_t,  # \sqrt{\beta_t}
        "alphabar_t": alphabar_t,  # \bar{\alpha_t}
        "sqrtab": sqrtab,  # \sqrt{\bar{\alpha_t}}
        "sqrtmab": sqrtmab,  # \sqrt{1-\bar{\alpha_t}}
        "mab_over_sqrtmab": mab_over_sqrtmab_inv,  # (1-\alpha_t)/
 ↪\sqrt{1-\bar{\alpha_t}}
    }
```

[21]:
```
class DDPM(nn.Module):
    def __init__(self, nn_model, betas, n_T, device, drop_prob=0.1):
        super(DDPM, self).__init__()
        self.nn_model = nn_model.to(device)

        for k, v in ddpm_schedules(betas[0], betas[1], n_T).items():
            self.register_buffer(k, v)

        self.n_T = n_T
        self.device = device
        self.drop_prob = drop_prob
        self.loss_mse = nn.MSELoss()

    def forward(self, x, c):

        t = torch.randint(1, self.n_T+1, (x.shape[0],)).to(self.device)  # t ~␣
 ↪Uniform(0, n_T)
        noise = torch.randn_like(x)  # eps ~ N(0, 1)
```

```
        x_t = (
            self.sqrtab[t, None, None, None] * x
            + self.sqrtmab[t, None, None, None] * noise
        )  # This is the x_t, which is sqrt(alphabar) x_0 + sqrt(1-alphabar) *␣
↪eps

        # return MSE between added noise, and our predicted noise
        return self.loss_mse(noise, self.nn_model(x_t, c, t / self.n_T))

    def sample(self, n_sample, size, device, guide_w = 0.0):

        x_i = torch.randn(n_sample, *size).to(device)  # x_T ~ N(0, 1), sample␣
↪initial noise
        c_i = torch.arange(0,10).to(device) # labels to generate images
        c_i = c_i.repeat(int(n_sample/c_i.shape[0]))

        x_i_store = [] # keep track of generated samples for plotting
        # print()
        for i in range(self.n_T, 0, -1):
            print(f'sampling timestep {i}',end='\r')
            t_is = torch.tensor([i / self.n_T]).to(device)
            t_is = t_is.repeat(n_sample,1,1,1)

            z = torch.randn(n_sample, *size).to(device) if i > 1 else 0

            eps1 = self.nn_model(x_i, c_i, t_is)
            eps2 = self.nn_model(x_i, None, t_is)
            eps = (1+guide_w)*eps1 - guide_w*eps2 # guidance weight from CFG␣
↪paper
            x_i = (
                self.oneover_sqrta[i] * (x_i - eps * self.mab_over_sqrtmab[i])
                + self.sqrt_beta_t[i] * z
            )
            if i%20==0 or i==self.n_T or i<8: #storing images for gif
                x_i_store.append(x_i.detach().cpu().numpy())

        x_i_store = np.array(x_i_store)
        return x_i, x_i_store
```

1.0.5 The implementation of the neural network used to estimate the noise should go here

The network should make use of both time and context embedding.

Any functions/methods used for time and context embeddings should also go here.

[22]:
```
class ResidualConvBlock(nn.Module):
    def __init__(self, in_channels: int, out_channels: int):
        super(ResidualConvBlock, self).__init__()
        self.conv1 = nn.Sequential(
            nn.Conv2d(in_channels, out_channels, 3, 1, 1),
            nn.BatchNorm2d(out_channels),
            nn.GELU(),
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(out_channels, out_channels, 3, 1, 1),
            nn.BatchNorm2d(out_channels),
            nn.GELU(),
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x1 = self.conv1(x)
        x2 = self.conv2(x1)
        return x2


class UnetDown(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(UnetDown, self).__init__()
        self.model = nn.Sequential(
            ResidualConvBlock(in_channels, out_channels),
            nn.MaxPool2d(2),
            )

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


class UnetUp(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(UnetUp, self).__init__()
        self.model = nn.Sequential(
            nn.ConvTranspose2d(in_channels, out_channels, 2, 2),
            ResidualConvBlock(out_channels, out_channels),
            ResidualConvBlock(out_channels, out_channels),
            )

    def forward(self, x, skip):
        x = torch.cat((x, skip), 1)
        x = self.model(x)
        return x
```

```
class EmbedMLP(nn.Module):
    def __init__(self, in_dim, out_dim):
        super(EmbedMLP, self).__init__()
        self.in_dim = in_dim

        self.model = nn.Sequential(
            nn.Linear(in_dim, out_dim),
            nn.GELU(),
            nn.Linear(out_dim, out_dim),)

    def forward(self, x):
        x = x.view(-1, self.in_dim)
        return self.model(x)
```

[23]:
```
class Cond_Unet(nn.Module):
    def __init__(self, in_channels, n_feat = 256, n_classes=10):
        super(Cond_Unet, self).__init__()

        self.in_channels = in_channels
        self.n_feat = n_feat
        self.n_classes = n_classes

        self.init_conv = ResidualConvBlock(in_channels, n_feat)

        self.down1 = UnetDown(n_feat, n_feat)
        self.down2 = UnetDown(n_feat, 2 * n_feat)

        self.to_vec = nn.Sequential(nn.AvgPool2d(7), nn.GELU())

        self.timeembed1 = EmbedMLP(1, 2*n_feat)
        self.timeembed2 = EmbedMLP(1, 1*n_feat)
        self.contextembed1 = EmbedMLP(n_classes, 2*n_feat)
        self.contextembed2 = EmbedMLP(n_classes, 1*n_feat)

        self.up0 = nn.Sequential(
            nn.ConvTranspose2d(2 * n_feat, 2 * n_feat, 7, 7),
            nn.GroupNorm(8, 2 * n_feat),
            nn.ReLU(),
        )

        self.up1 = UnetUp(4 * n_feat, n_feat)
        self.up2 = UnetUp(2 * n_feat, n_feat)
        self.out = nn.Sequential(
            nn.Conv2d(2 * n_feat, n_feat, 3, 1, 1),
            nn.GroupNorm(8, n_feat),
            nn.ReLU(),
            nn.Conv2d(n_feat, self.in_channels, 3, 1, 1),
```

```
        )

    def forward(self, x, c, t):
        # x is (noisy) image, c is context label, t is timestep,

        x = self.init_conv(x)
        down1 = self.down1(x)
        down2 = self.down2(down1)
        hiddenvec = self.to_vec(down2)

        # embed context, time step
        temb1 = self.timeembed1(t).view(-1, self.n_feat * 2, 1, 1)
        temb2 = self.timeembed2(t).view(-1, self.n_feat, 1, 1)

        if c is not None:
            c = nn.functional.one_hot(c, num_classes=self.n_classes).type(torch.
↪float)
            cemb1 = self.contextembed1(c).view(-1, self.n_feat * 2, 1, 1)
            cemb2 = self.contextembed2(c).view(-1, self.n_feat, 1, 1)
        else:
            cemb1 = torch.ones_like(temb1)
            cemb2 = torch.ones_like(temb2)

        up1 = self.up0(hiddenvec)
        up2 = self.up1(cemb1*up1+ temb1, down2)  # add and multiply embeddings
        up3 = self.up2(cemb2*up2+ temb2, down1)
        out = self.out(torch.cat((up3, x), 1))
        return out
```

1.0.6 Show that you can generate images before the model is trained

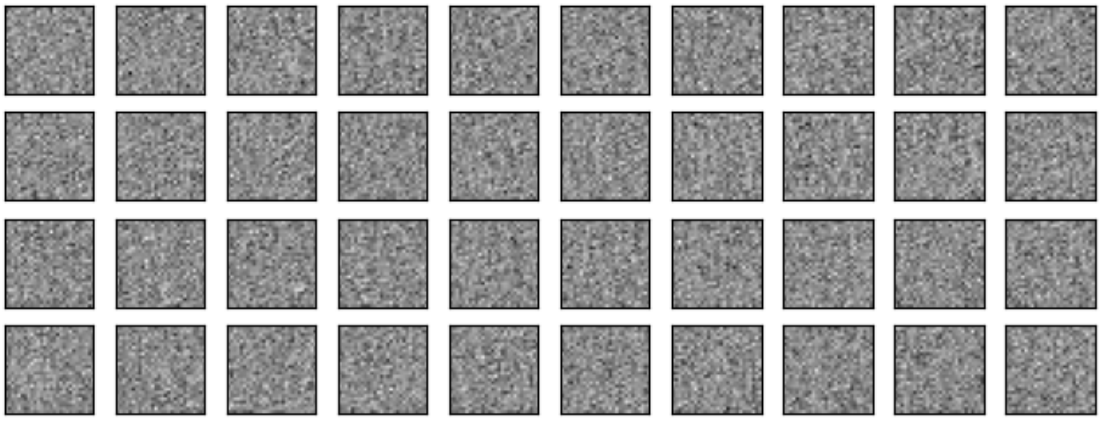
This should demonstrate that the backwards pass works. The generated images are expected to be noise.

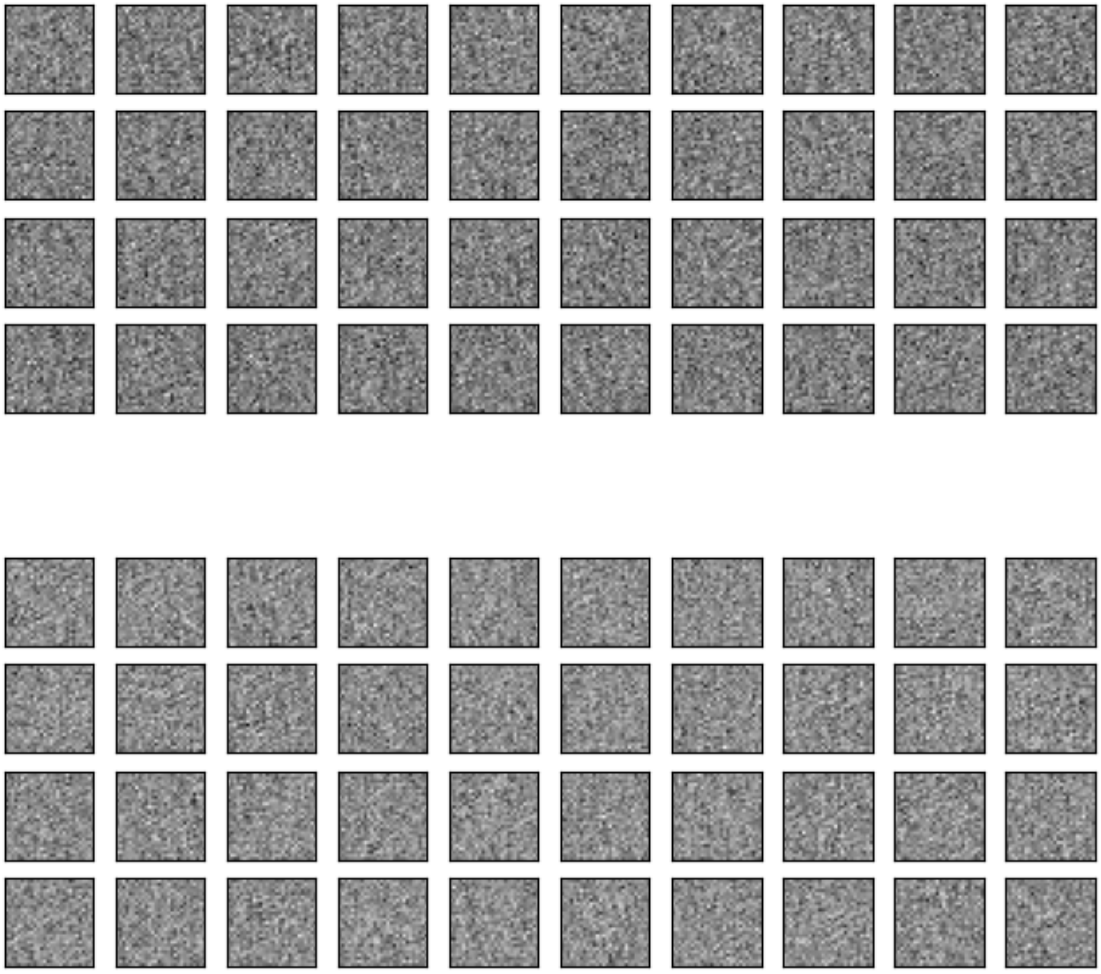
[24]:
```
def generate_images(ddpm, ep, n_classes, ws_test):
    with torch.no_grad():
            n_sample = 4*n_classes
            for w_i, w in enumerate(ws_test):
                x_gen, x_gen_store = ddpm.sample(n_sample, (1, 28, 28), device,␣
↪guide_w=w)

                grid = make_grid(x_gen*-1 + 1, nrow=10)
                save_image(grid, save_dir + f"image_ep{ep}_w{w}.png")
                print('saved image at ' + save_dir + f"image_ep{ep}_w{w}.png")

                # create gif of images evolving over time, based on x_gen_store
```

```
                fig, axs = plt.subplots(nrows=int(n_sample/n_classes),␣
↪ncols=n_classes,sharex=True,sharey=True,figsize=(8,3))
                def animate_diff(i, x_gen_store):
                    print(f'gif animating frame {i} of {x_gen_store.shape[0]}',␣
↪end='\r')
                    plots = []
                    for row in range(int(n_sample/n_classes)):
                        for col in range(n_classes):
                            axs[row, col].clear()
                            axs[row, col].set_xticks([])
                            axs[row, col].set_yticks([])
                            plots.append(axs[row, col].
↪imshow(-x_gen_store[i,(row*n_classes)+col,0],cmap='gray',vmin=(-x_gen_store[i]).
↪min(), vmax=(-x_gen_store[i]).max()))
                    return plots
                ani = FuncAnimation(fig, animate_diff, fargs=[x_gen_store], ␣
↪interval=200, blit=False, repeat=True, frames=x_gen_store.shape[0])
                ani.save(save_dir + f"gif_ep{ep}_w{w}.gif", dpi=100,␣
↪writer=PillowWriter(fps=5))
                print('saved image at ' + save_dir + f"gif_ep{ep}_w{w}.gif")
```

[25]:
```
ddpm = DDPM(nn_model=Cond_Unet(in_channels=1, n_feat=n_feat,␣
↪n_classes=n_classes), betas=(beta_start, beta_end), n_T=n_T, device=device,␣
↪drop_prob=0.1)
ddpm.to(device)

generate_images(ddpm, 'test', n_classes, ws_test)
```

```
saved image at ./data/diffusion/image_eptest_w0.0.png
saved image at ./data/diffusion/gif_eptest_w0.0.gif
saved image at ./data/diffusion/image_eptest_w0.5.png
saved image at ./data/diffusion/gif_eptest_w0.5.gif
saved image at ./data/diffusion/image_eptest_w2.0.png
saved image at ./data/diffusion/gif_eptest_w2.0.gif
```





1.0.7 Implement training loop

Train the model here. There should be some indication of how long the model took to train, both total and per epoch. For good results you will want to train the model for several epochs, but with a good implementation you should expect to see something that looks like digits after only a single epoch.

[26]:
```
def train(dataset, dataloader):

    ddpm = DDPM(nn_model=Cond_Unet(in_channels=1, n_feat=n_feat,␣
↪n_classes=n_classes), betas=(1e-4, 0.02), n_T=n_T, device=device,␣
↪drop_prob=0.1)
    ddpm.to(device)
```

```
    optim = torch.optim.Adam(ddpm.parameters(), lr=lrate)

    for ep in range(n_epoch):
        print(f'epoch {ep}')
        ddpm.train()

        # linear lrate decay
        optim.param_groups[0]['lr'] = lrate*(1-ep/n_epoch)

        pbar = tqdm(dataloader)
        for x, c in pbar:
            optim.zero_grad()
            x = x.to(device)
            c = c.to(device)
            if np.random.random() < ddpm.drop_prob:
                c = None
            loss = ddpm(x, c)
            loss.backward()
            pbar.set_description(f"loss: {loss:.4f}")
            optim.step()

        ddpm.eval()
        generate_images(ddpm, ep, n_classes, ws_test)

        # optionally save model
        if save_model and ep == int(n_epoch-1):
            torch.save(ddpm.state_dict(), save_dir + f"model_{ep}.pth")
            print('saved model at ' + save_dir + f"model_{ep}.pth")
```

1.0.8 Train and visualize the model

We want to see several generated examples of each digit.

The images you see below are generated for guidance weight [0, 0.5, 2] after each epoch.

[27]:
```
train(dataset, dataloader)
```

```
epoch 0
```

```
  0%|          | 0/235 [00:00<?, ?it/s]loss: 0.0474: 100%|     | 235/235
[03:57<00:00,  1.01s/it]
```

```
saved image at ./data/diffusion/image_ep0_w0.0.png
saved image at ./data/diffusion/gif_ep0_w0.0.gif
saved image at ./data/diffusion/image_ep0_w0.5.png
saved image at ./data/diffusion/gif_ep0_w0.5.gif
saved image at ./data/diffusion/image_ep0_w2.0.png
saved image at ./data/diffusion/gif_ep0_w2.0.gif
epoch 1
```

```
loss: 0.0390: 100%|      | 235/235 [02:19<00:00,  1.68it/s]

saved image at ./data/diffusion/image_ep1_w0.0.png
saved image at ./data/diffusion/gif_ep1_w0.0.gif
saved image at ./data/diffusion/image_ep1_w0.5.png
saved image at ./data/diffusion/gif_ep1_w0.5.gif
saved image at ./data/diffusion/image_ep1_w2.0.png
saved image at ./data/diffusion/gif_ep1_w2.0.gif
epoch 2

loss: 0.0282: 100%|      | 235/235 [05:52<00:00,  1.50s/it]

saved image at ./data/diffusion/image_ep2_w0.0.png
saved image at ./data/diffusion/gif_ep2_w0.0.gif
saved image at ./data/diffusion/image_ep2_w0.5.png
saved image at ./data/diffusion/gif_ep2_w0.5.gif
saved image at ./data/diffusion/image_ep2_w2.0.png
saved image at ./data/diffusion/gif_ep2_w2.0.gif
epoch 3

loss: 0.0310: 100%|      | 235/235 [06:07<00:00,  1.56s/it]

saved image at ./data/diffusion/image_ep3_w0.0.png
saved image at ./data/diffusion/gif_ep3_w0.0.gif
saved image at ./data/diffusion/image_ep3_w0.5.png
saved image at ./data/diffusion/gif_ep3_w0.5.gif
saved image at ./data/diffusion/image_ep3_w2.0.png
saved image at ./data/diffusion/gif_ep3_w2.0.gif
epoch 4

loss: 0.0360: 100%|      | 235/235 [06:08<00:00,  1.57s/it]

saved image at ./data/diffusion/image_ep4_w0.0.png
saved image at ./data/diffusion/gif_ep4_w0.0.gif
saved image at ./data/diffusion/image_ep4_w0.5.png
saved image at ./data/diffusion/gif_ep4_w0.5.gif
saved image at ./data/diffusion/image_ep4_w2.0.png
saved image at ./data/diffusion/gif_ep4_w2.0.gif
epoch 5

loss: 0.0287: 100%|      | 235/235 [06:10<00:00,  1.57s/it]

saved image at ./data/diffusion/image_ep5_w0.0.png
saved image at ./data/diffusion/gif_ep5_w0.0.gif
saved image at ./data/diffusion/image_ep5_w0.5.png
saved image at ./data/diffusion/gif_ep5_w0.5.gif
saved image at ./data/diffusion/image_ep5_w2.0.png
saved image at ./data/diffusion/gif_ep5_w2.0.gif
epoch 6

loss: 0.0310: 100%|      | 235/235 [06:13<00:00,  1.59s/it]
```

```
saved image at ./data/diffusion/image_ep6_w0.0.png
saved image at ./data/diffusion/gif_ep6_w0.0.gif
saved image at ./data/diffusion/image_ep6_w0.5.png
saved image at ./data/diffusion/gif_ep6_w0.5.gif
saved image at ./data/diffusion/image_ep6_w2.0.png

C:\Users\mamoo\AppData\Local\Temp\ipykernel_24692\2244048863.py:12:
RuntimeWarning: More than 20 figures have been opened. Figures created through
the pyplot interface (`matplotlib.pyplot.figure`) are retained until explicitly
closed and may consume too much memory. (To control this warning, see the
rcParam `figure.max_open_warning`). Consider using `matplotlib.pyplot.close()`.
  fig, axs = plt.subplots(nrows=int(n_sample/n_classes),
ncols=n_classes,sharex=True,sharey=True,figsize=(8,3))

saved image at ./data/diffusion/gif_ep6_w2.0.gif
epoch 7

loss: 0.0236: 100%|      | 235/235 [06:23<00:00,  1.63s/it]

saved image at ./data/diffusion/image_ep7_w0.0.png
saved image at ./data/diffusion/gif_ep7_w0.0.gif
saved image at ./data/diffusion/image_ep7_w0.5.png
saved image at ./data/diffusion/gif_ep7_w0.5.gif
saved image at ./data/diffusion/image_ep7_w2.0.png
saved image at ./data/diffusion/gif_ep7_w2.0.gif
epoch 8

loss: 0.0299: 100%|      | 235/235 [06:04<00:00,  1.55s/it]

saved image at ./data/diffusion/image_ep8_w0.0.png
saved image at ./data/diffusion/gif_ep8_w0.0.gif
saved image at ./data/diffusion/image_ep8_w0.5.png
saved image at ./data/diffusion/gif_ep8_w0.5.gif
saved image at ./data/diffusion/image_ep8_w2.0.png
saved image at ./data/diffusion/gif_ep8_w2.0.gif
epoch 9

loss: 0.0252: 100%|      | 235/235 [06:15<00:00,  1.60s/it]

saved image at ./data/diffusion/image_ep9_w0.0.png
saved image at ./data/diffusion/gif_ep9_w0.0.gif
saved image at ./data/diffusion/image_ep9_w0.5.png
saved image at ./data/diffusion/gif_ep9_w0.5.gif
saved image at ./data/diffusion/image_ep9_w2.0.png
saved image at ./data/diffusion/gif_ep9_w2.0.gif
epoch 10

loss: 0.0243: 100%|      | 235/235 [06:15<00:00,  1.60s/it]

saved image at ./data/diffusion/image_ep10_w0.0.png
saved image at ./data/diffusion/gif_ep10_w0.0.gif
saved image at ./data/diffusion/image_ep10_w0.5.png
saved image at ./data/diffusion/gif_ep10_w0.5.gif
```

```
saved image at ./data/diffusion/image_ep10_w2.0.png
saved image at ./data/diffusion/gif_ep10_w2.0.gif
epoch 11
loss: 0.0278: 100%|      | 235/235 [06:23<00:00,  1.63s/it]
saved image at ./data/diffusion/image_ep11_w0.0.png
saved image at ./data/diffusion/gif_ep11_w0.0.gif
saved image at ./data/diffusion/image_ep11_w0.5.png
saved image at ./data/diffusion/gif_ep11_w0.5.gif
saved image at ./data/diffusion/image_ep11_w2.0.png
saved image at ./data/diffusion/gif_ep11_w2.0.gif
epoch 12
loss: 0.0343: 100%|      | 235/235 [06:05<00:00,  1.55s/it]
saved image at ./data/diffusion/image_ep12_w0.0.png
saved image at ./data/diffusion/gif_ep12_w0.0.gif
saved image at ./data/diffusion/image_ep12_w0.5.png
saved image at ./data/diffusion/gif_ep12_w0.5.gif
saved image at ./data/diffusion/image_ep12_w2.0.png
saved image at ./data/diffusion/gif_ep12_w2.0.gif
epoch 13
loss: 0.0221: 100%|      | 235/235 [06:06<00:00,  1.56s/it]
saved image at ./data/diffusion/image_ep13_w0.0.png
saved image at ./data/diffusion/gif_ep13_w0.0.gif
saved image at ./data/diffusion/image_ep13_w0.5.png
saved image at ./data/diffusion/gif_ep13_w0.5.gif
saved image at ./data/diffusion/image_ep13_w2.0.png
saved image at ./data/diffusion/gif_ep13_w2.0.gif
epoch 14
loss: 0.0200: 100%|      | 235/235 [06:14<00:00,  1.59s/it]
saved image at ./data/diffusion/image_ep14_w0.0.png
saved image at ./data/diffusion/gif_ep14_w0.0.gif
saved image at ./data/diffusion/image_ep14_w0.5.png
saved image at ./data/diffusion/gif_ep14_w0.5.gif
saved image at ./data/diffusion/image_ep14_w2.0.png
saved image at ./data/diffusion/gif_ep14_w2.0.gif
epoch 15
loss: 0.0227: 100%|      | 235/235 [06:08<00:00,  1.57s/it]
saved image at ./data/diffusion/image_ep15_w0.0.png
saved image at ./data/diffusion/gif_ep15_w0.0.gif
saved image at ./data/diffusion/image_ep15_w0.5.png
saved image at ./data/diffusion/gif_ep15_w0.5.gif
saved image at ./data/diffusion/image_ep15_w2.0.png
saved image at ./data/diffusion/gif_ep15_w2.0.gif
epoch 16
```

```
loss: 0.0208: 100%|      | 235/235 [06:11<00:00,  1.58s/it]

saved image at ./data/diffusion/image_ep16_w0.0.png
saved image at ./data/diffusion/gif_ep16_w0.0.gif
saved image at ./data/diffusion/image_ep16_w0.5.png
saved image at ./data/diffusion/gif_ep16_w0.5.gif
saved image at ./data/diffusion/image_ep16_w2.0.png
saved image at ./data/diffusion/gif_ep16_w2.0.gif
epoch 17

loss: 0.0216: 100%|      | 235/235 [06:08<00:00,  1.57s/it]

saved image at ./data/diffusion/image_ep17_w0.0.png
saved image at ./data/diffusion/gif_ep17_w0.0.gif
saved image at ./data/diffusion/image_ep17_w0.5.png
saved image at ./data/diffusion/gif_ep17_w0.5.gif
saved image at ./data/diffusion/image_ep17_w2.0.png
saved image at ./data/diffusion/gif_ep17_w2.0.gif
epoch 18

loss: 0.0230: 100%|      | 235/235 [06:36<00:00,  1.69s/it]

saved image at ./data/diffusion/image_ep18_w0.0.png
saved image at ./data/diffusion/gif_ep18_w0.0.gif
saved image at ./data/diffusion/image_ep18_w0.5.png
saved image at ./data/diffusion/gif_ep18_w0.5.gif
saved image at ./data/diffusion/image_ep18_w2.0.png
saved image at ./data/diffusion/gif_ep18_w2.0.gif
epoch 19

loss: 0.0249: 100%|      | 235/235 [06:52<00:00,  1.75s/it]

saved image at ./data/diffusion/image_ep19_w0.0.png
saved image at ./data/diffusion/gif_ep19_w0.0.gif
saved image at ./data/diffusion/image_ep19_w0.5.png
saved image at ./data/diffusion/gif_ep19_w0.5.gif
saved image at ./data/diffusion/image_ep19_w2.0.png
saved image at ./data/diffusion/gif_ep19_w2.0.gif
epoch 20

loss: 0.0198: 100%|      | 235/235 [07:01<00:00,  1.79s/it]

saved image at ./data/diffusion/image_ep20_w0.0.png
saved image at ./data/diffusion/gif_ep20_w0.0.gif
saved image at ./data/diffusion/image_ep20_w0.5.png
saved image at ./data/diffusion/gif_ep20_w0.5.gif
saved image at ./data/diffusion/image_ep20_w2.0.png
saved image at ./data/diffusion/gif_ep20_w2.0.gif
epoch 21

loss: 0.0205: 100%|      | 235/235 [06:55<00:00,  1.77s/it]
```

```
saved image at ./data/diffusion/image_ep21_w0.0.png
saved image at ./data/diffusion/gif_ep21_w0.0.gif
saved image at ./data/diffusion/image_ep21_w0.5.png
saved image at ./data/diffusion/gif_ep21_w0.5.gif
saved image at ./data/diffusion/image_ep21_w2.0.png
saved image at ./data/diffusion/gif_ep21_w2.0.gif
epoch 22
```

```
loss: 0.0309: 100%|      | 235/235 [07:02<00:00,  1.80s/it]
```

```
saved image at ./data/diffusion/image_ep22_w0.0.png
saved image at ./data/diffusion/gif_ep22_w0.0.gif
saved image at ./data/diffusion/image_ep22_w0.5.png
saved image at ./data/diffusion/gif_ep22_w0.5.gif
saved image at ./data/diffusion/image_ep22_w2.0.png
saved image at ./data/diffusion/gif_ep22_w2.0.gif
epoch 23
```

```
loss: 0.0229: 100%|      | 235/235 [06:56<00:00,  1.77s/it]
```

```
saved image at ./data/diffusion/image_ep23_w0.0.png
saved image at ./data/diffusion/gif_ep23_w0.0.gif
saved image at ./data/diffusion/image_ep23_w0.5.png
saved image at ./data/diffusion/gif_ep23_w0.5.gif
saved image at ./data/diffusion/image_ep23_w2.0.png
saved image at ./data/diffusion/gif_ep23_w2.0.gif
epoch 24
```

```
loss: 0.0232: 100%|      | 235/235 [02:25<00:00,  1.62it/s]
```

```
saved image at ./data/diffusion/image_ep24_w0.0.png
saved image at ./data/diffusion/gif_ep24_w0.0.gif
saved image at ./data/diffusion/image_ep24_w0.5.png
saved image at ./data/diffusion/gif_ep24_w0.5.gif
saved image at ./data/diffusion/image_ep24_w2.0.png
saved image at ./data/diffusion/gif_ep24_w2.0.gif
```