Exercises Week 3

1.

```python
def get_betas(self, schedule='linear'):
    if schedule == 'linear':
        # HINT: use torch.linspace to create a linear schedule from beta
        # add your own (e.g. cosine)
        return torch.linspace(self.beta_start, self.beta_end, self.T)
    else:
        raise NotImplementedError('Not implemented!')
```

2.

```python
def q_sample(self, x, t):
    """
    x: input image (x0)
    t: timestep: should be torch.tensor

    Forward diffusion process
    q(x_t | x_0) = sqrt(alpha_hat_t) * x0 + sqrt(1-alpha_hat_t) * N(0,1)

    Should return q(x_t | x_0), noise
    """

    # TASK 2: Implement the forward process
    # HINT: use torch.sqrt to calculate the sqrt of alphas_bar at timestep t
    sqrt_alpha_bar = torch.sqrt(self.alphas_bar[t])
    sqrt_alpha_bar = sqrt_alpha_bar[:, None, None, None] # match image dimensions

    # HINT: calculate the sqrt of 1 - alphas_bar at time step t
    sqrt_one_minus_alpha_bar = torch.sqrt(1 - self.alphas_bar[t])
    sqrt_one_minus_alpha_bar = sqrt_one_minus_alpha_bar[:, None, None, None] # mat

    # HINT: sample noise from a normal distribution. It should match the shape of
    noise = torch.randn_like(x)
    assert noise.shape == x.shape, 'Invalid shape of noise'

    # HINT: Create the noisy version of x. See Eq. 4 in the ddpm paper at page 2
    x_noised = x * sqrt_alpha_bar + sqrt_one_minus_alpha_bar * noise
    return x_noised, noise
```

3.

```python
def p_mean_std(self, model, x_t, t):
    """
    Calculate mean and std of p(x_{t-1} | x_t) using the reverse process and model
    """
    alpha = self.alphas[t][:, None, None, None] # match image dimensions
    alpha_bar = self.alphas_bar[t][:, None, None, None] # match image dimensions
    beta = self.betas[t][:, None, None, None] # match image dimensions

    # TASK 3 : Implement the revese process
    # HINT: use model to predict noise
    predicted_noise = model(x_t, t)
    # HINT: calculate the mean of the distribution p(x_{t-1} | x_t). See Eq. 11 in the ddpm paper at page 4
    mean = (1 / torch.sqrt(alpha)) * (x_t - (beta / torch.sqrt(1 - alpha_bar)) * predicted_noise)
    std = torch.sqrt(beta)

    return mean, std

def p_sample(self, model, x_t, t):
    """
    Sample from p(x{t-1} | x_t) using the reverse process and model
    """
    # TASK 3: implement the reverse process
    mean, std = self.p_mean_std(model, x_t, t)

    # HINT: Having calculate the mean and std of p(x{x_t} | x_t), we sample noise from a normal distribution.
    # see line 3 of the Algorithm 2 (Sampling) at page 4 of the ddpm paper.
    noise = torch.randn_like(x_t) * std if t[0] > 1 else torch.zeros_like(x_t)

    # Calculate x_{t-1}, see line 4 of the Algorithm 2 (Sampling) at page 4 of the ddpm paper.
    x_t_prev = mean + noise
    return x_t_prev
```

4.

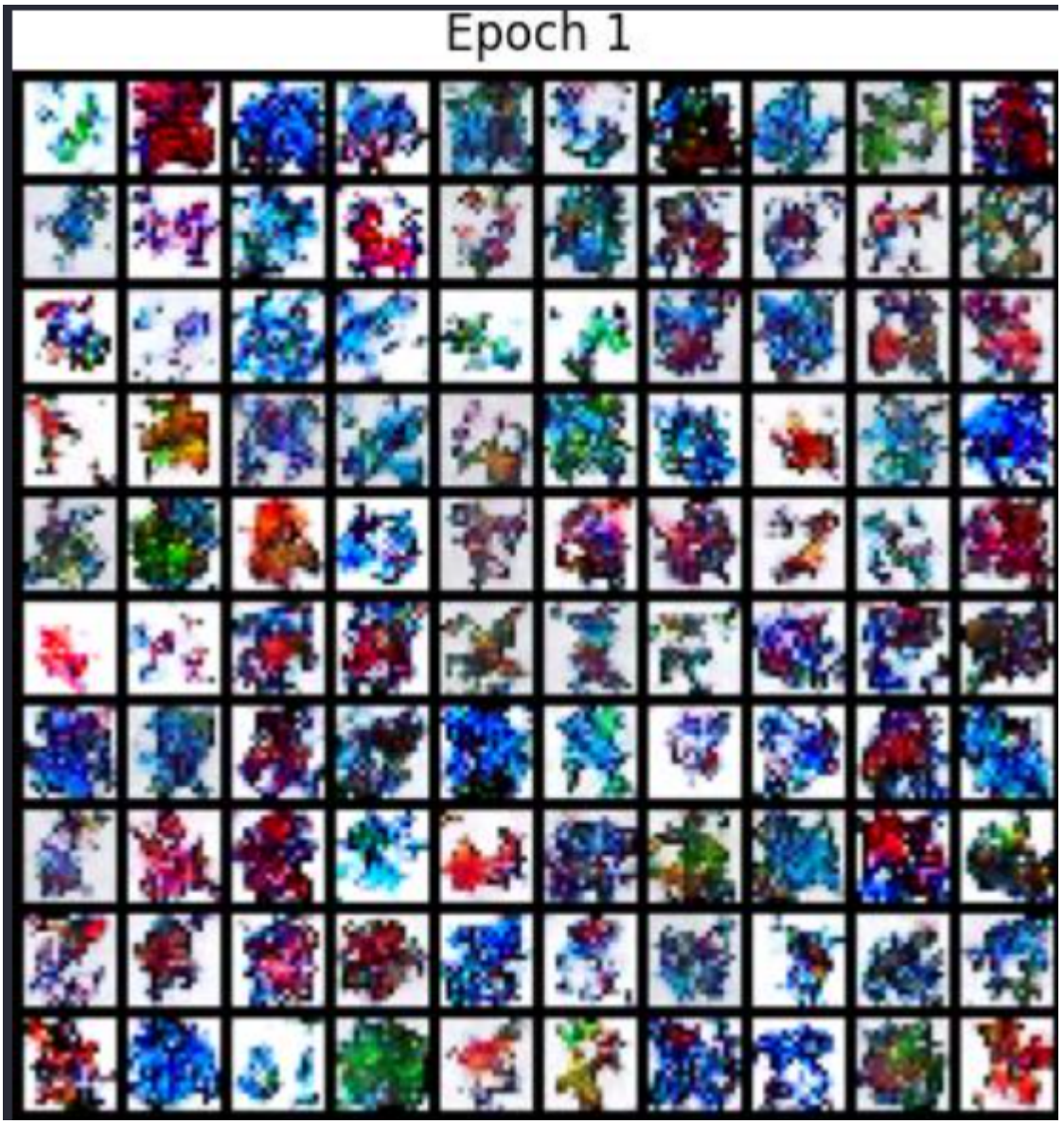
```python
for epoch in range(1, num_epochs + 1):
    logging.info(f"Starting epoch {epoch}:")
    pbar = tqdm(dataloader)

    for i, images in enumerate(pbar):
        images = images.to(device)

        # TASK 4: implement the training loop
        t = diffusion.sample_timesteps(images.shape[0]).to(device) # line 3 from
        # inject noise to the images (forward process), HINT: use q_sample
        x_t, noise = diffusion.q_sample(images, t)
        # predict noise of x_t using the UNet
        predicted_noise = model(x_t, t)
        # loss between noise and predicted noise
        loss = mse(noise, predicted_noise)


        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Results

The model, which is completed per the code implementations shown above, is trained for 30 epochs and the resulting generated images for some of these epochs are presented below.

For 1 and 5 epochs, there is clearly no concrete object in the generated image, however, it is obviously not just pure noise.

For epochs 10 and 15 the results are more stable, and for the 30’th epoch, the generated images are mostly clearly drawn objects or imaginary characters that resemble just that (I would mistake them for Pokémon any day at least).




Epoch 5


Epoch 10


Epoch 15


Epoch 30