Exercises Week 3

Advanced Machine Learning (02460)
Technical University of Denmark
Jes Frellsen

February 2026
(Version 1.31)

This week's exercises are on **diffusion models** and consist of theoretical exercises that will prepare you for the written exam and a programming exercise that will prepare you for the project(s). We will focus on the *denoising diffusion probabilistic model* (DDPM) as described in section 5.5.3 of the textbook (Tomczak 2024) and in the paper by Ho et al. (2020).

1 Theoretical exercises

Exercise 3.1 From equation (5.98) in the textbook (Tomczak 2024), we have that

$$q(\mathbf{z}_t \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{z}_t \mid \sqrt{\bar{\alpha}_t}\mathbf{x}, (1-\bar{\alpha}_t)\mathbf{I}\right) \tag{1}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$. Show that for any noise variance schedule $0 < \beta_1 < \cdots < \beta_T < 1$, $q(\mathbf{z}_T \mid \mathbf{x})$ become as standard Gaussian as $T \to \infty$, i.e.,

$$q(\mathbf{z}_T \mid \mathbf{x}) \to \mathcal{N}(\mathbf{z}_T \mid \mathbf{0}, \mathbf{I}) \quad \text{as} \quad T \to \infty. \tag{2}$$

Exercise 3.2 Derive equation (5) from equation (3) in the paper by Ho et al. (2020). The derivation is given in Appendix A of the paper, but you should explain every step going from each equation to the next in equations (17)–(22).

Exercise 3.3 Consider the $(t-1)$th term in the DDPM ELBO, c.f., equation (5.102) by Tomczak (2024) or equation (5) by Ho et al. (2020),

$$L_{t-1} = \mathrm{KL}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) \,\|\, p(\mathbf{z}_{t-1}|\mathbf{z}_t)), \tag{25}$$

where

$$q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{x}) = \mathcal{N}(\mathbf{z}_{t-1} \mid \tilde{\mu}(\mathbf{z}_t, \mathbf{x}), \tilde{\beta}_t\mathbf{I}) \tag{26}$$

$$p(\mathbf{z}_{t-1}|\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_{t-1} \mid \mu_\theta(\mathbf{z}_t, t), \sigma_t^2\mathbf{I}). \tag{27}$$

Show that we write this term as

$$L_{t-1} = \frac{1}{2\sigma_t^2}\left\|\tilde{\mu}(\mathbf{z}_t, \mathbf{x}) - \mu_\theta(\mathbf{z}_t, t)\right\|^2 + C, \tag{28}$$

where C does not depend on θ.

Hint: We can write the KL divergence between two D-dimensional multivariate Gaussian distributions $\mathcal{N}_0(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and $\mathcal{N}_1(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ as

$$\mathrm{KL}(\mathcal{N}_0 \,\|\, \mathcal{N}_1) = \frac{1}{2}\left(\log\det(\boldsymbol{\Sigma}_1\boldsymbol{\Sigma}_0^{-1})) + (\boldsymbol{\mu}_0 - \boldsymbol{\mu}1)^\intercal\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}1) + \mathrm{tr}(\Sigma_1^{-1}\Sigma_0) - D\right), \tag{29}$$

see, e.g., Rasmussen and Williams (2005, equation (A.23)).

2 Programming exercises

The main task in this week's programming exercise is to implement the DDPM (Ho et al. 2020). We will start with the two simple toy datasets from week 2 (TwoGaussians and Chequerboard), and then we will move on to learning DDPMs on MNIST.

We have provided you with two files:

- `ddpm.py` contains an incomplete DDMP implementation for the toy datasets.
- `unet.py` contains the code for a U-Net predicting ϵ of reverse process on MNIST[1].

You will also need `ToyData.py` from week 2.

Exercise 3.4 Complete the DDPM implementation (`ddpm.py`), by implementing the following parts:

- `DDPM.negative_elbo(...)` should return the negative ELBO of equation (14) by Ho et al. (2020) by implementing Algorithm 1 in the paper.
- `DDPM.sample(shape)` should implement Algorithm 2 in the paper by Ho et al. (2020). For the covariance matrix of the reverse process use $\sigma_t^2\mathbf{I}$ with $\sigma_t^2 = \beta_t$. For example, to sample 5000 samples for the 2D toy examples, the method would be called with the argument `shape=(5000, 2)`.

A simple, fully connected network is implemented in the class `FcNetwork`. Note that the method `FcNetwork.forward(x, t)` takes as input a batch of data `x` of dimension `(batch_size, input_dim)` and the time step for each data point in the batch of dimension `(batch_size, 1)`. The method concatenates the data and the time step before inputting it to the network, and it is a good idea to normalize the time step to $[0, 1]$.

Test the implementation on both the TwoGaussians and Chequerboard datasets and answer the following questions:

[1]The architecture and the implementation of the U-Net is from https://github.com/mfkasim1/score-based-tutorial/blob/main/03-SGM-with-SDE-MNIST.ipynb.

- Can you improve the fit to the Chequerboard dataset by modifying the network architecture?
- How does the DDPM qualitatively compare to the Flow model from week 2 on the two toy datasets?

Exercise 3.5 Use the DPPM implementation from exercise 3.4 to learn a DDPM on MNIST. You *do not* need to implement a discrete likelihood function for the DDPM as suggested in section 3.3 by Ho et al. (2020). Instead, we will dequantized the pixel values (as we did for flows) and transform them to $[-1, 1]$, which can be done with the code:

```
from torchvision import transforms

transform=transforms.Compose([transforms.ToTensor(),
            transforms.Lambda(lambda x: x + torch.rand(x.shape)/255),
            transforms.Lambda(lambda x: (x-0.5)*2.0),
            transforms.Lambda(lambda x: x.flatten())]
            )

train_data = datasets.MNIST('data/',
                            train=True,
                            download=True,
                            transform=transform)
```

Remember to transform the pixel values back to $[0, 1]$ before displaying samples.

You should both test a fully connected architecture and the provided U-Net architecture (in `unet.py`). Please answer the following questions:

- Can you learn a DDPM on MNIST using a fully connected architecture?
- How do the samples from the DDPM qualitatively compare to the VAE and Flow models from week 1 and 2?

Hint: Remember to change the batch size to, e.g., 64, and you will need to train the model for around 50–100 epochs to get a good model, which takes around 15 minutes on a GPU.

References

Ho, J, A Jain, and P Abbeel (2020). "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 6840–6851. URL: https://arxiv.org/pdf/2006.11239.pdf.

Rasmussen, CE and CKI Williams (Nov. 2005). *Gaussian Processes for Machine Learning*. The MIT Press. URL: https://doi.org/10.7551/mitpress/3206.001.0001.

Tomczak, JM (2024). *Deep Generative Modeling*. Springer. URL: https://link.springer.com/book/10.1007/978-3-031-64087-2.