Exam project for PML 2025/2026

The exam project consists of parts A and B, reflecting different topics from the course. You are expected to work on the project in groups of 2-4 students. To avoid any misunderstandings, please read the following section about requirements and practicalities carefully before you begin.

Technical requirements and practicalities

Report length The project should be documented in a brief report (one report per group), which will determine your final grade in the course. The report should consist of maximum 6 pages in a reasonable font-size, including figures. You are allowed to include additional text, results and figures as appendices (appendices do not count to your total page number).

Report style We recommend using this template for your report: https://da.overleaf.com/latex/templates/style-and-template-for-preprints-arxiv-bio-arxiv/fxsnsrzpnvwc.

Should I include code in my report? It might make sense to include short snippets - but in general we do not expect you to include a lot of code in the report. You can optionally include it in an appendix, but the most useful approach is to provide a link to a relevant github repository.

Contribution from group members If members of the group contributed unequally to the different parts of the project, you should state this in the report, by specifying for each part the contribution percentage from the different members. If nothing is stated, we will assume that all members contributed equally. *Please try to avoid distributing the sub-questions of the project among the group members* - ***ideally, each group members should be involved equally in all aspects of the project.***

Handing in The reports should be handed in as a PDF through Digital Exam (eksamen.ku.dk), which will become available over the next week.

Questions Instead of the lecture on Tuesday, we will have a Q&A about the exam project (at the usual time: 10:00–12:00). We prefer that you save your questions for the in-class session, but if something urgent comes up, you are welcome to start a thread on the Absalon Discussion Board, and we'll try to answer it there. Please do not send us emails with questions about the project (since we then end up answering the same questions many times).

Generative AI declaration In accordance with the rules at UCPH, all exam project must include an "AI declaration". A template for this declaration is provided under your "Study Information" page under "Use of generative AI", so it should be easy to fill in. For details of how you are allowed to use AI, see this page: https://kunet.ku.dk/faculty-and-department/science/study-administration/examination/gai/. This declaration should be provided as an appendix to your report.

File locations All files mentioned in this document can be found under the "Final Project (exam)" module on the Absalon page for the course.

A Diffusion-based generative models

In the assignment for week 6 we coded a simple diffusion model on MNIST. In this first part of the project, you will explore different variations/extensions of this basic model.

Below are some ideas for things you could explore (although feel free to choose other variations/extensions if you want). For each variation/extension, implement it and compare its performance to the original model from the exercise session (we have provided a working solution to the exercise on Absalon: `mnist_ddpm_solution.ipynb`).

- Adjust the template code to convert the denoising diffusion MNIST code to a simple flow matching strategy. Describe which adjustments you had to make to make it work. Compare the training behavior and sampling quality of the models.

- Choose/Design a simple toy dataset where you expect DDPMs (and/or flow matching models) to struggle or fail. Change the template code to work with this data and investigate the failure modes, and whether/how they can be resolved. Note that it is important to be able to distinguish between faulty code and difficult data, so ideally, construct your data parametrically so that you can gradually increase the difficulty (and demonstrate that your model works as expected in the easier cases).

- One of the "Best Paper" award recipients at NeurIPS this year explored *memorization* in diffusion models: i.e. when model generate samples that are exactly identical to points from the training set[1]. Investigate whether you with your MNIST model can reproduce a setting when memorization occurs (e.g. by changing model capacity or training data size).

- One modeling approach that has been quite successful is to combine autoencoders (or VAEs) with diffusion models[2]. The basic idea is to first learn a AE/VAE on the dataset. We can now use this VAE to map inputs to a latent space, and train a diffusion model in this latent space. The fact that diffusion now happens in a much lower dimensional space has been reported to improve both speed and robustness of training. Check whether such an approach improves on the basic MNIST DDPM model. Remember that a problem with VAEs is that they tend to generate blurry images, so you might want to stick with standard autoencoders, or use a VAE variant that places greater emphasis on reconstruction[3].

- Implement a continuous-time version of the diffusion model, using the SDE approach Yang Song et al[4]. Does the continuous version provide any benefits

[1] Tony Bonnaire, Raphaël Urfin, Giulio Biroli, Marc Mezard, *Why Diffusion Models Don't Memorize: The Role of Implicit Dynamical Regularization in Training*, NeurIPS 2025, https://openreview.net/pdf?id=BSZqpqgqM0

[2] Stable Diffusion is an example of this: https://arxiv.org/abs/2112.10752

[3] The β-VAE is an example of this: https://openreview.net/forum?id=Sy2fzU9gl, and requires only minimal coding changes if you already have an implementation of a standard VAE.

[4] https://yang-song.net/blog/2021/score/

on its own, in terms of training time or image quality? You can experiment with some of the unique features of the continuous formulation - e.g. different noising processes, or the ODE-based deterministic reverse process.

Please use the template code we provided (`mnist_ddpm_solution.ipynb` **or the version you created from the original** `ddpm_template.ipynb` **template) as the basis for your implementations** to ensure that the models are run under similar conditions (and to make it easy to compare the changes made). If you prefer, you are of course welcome to move the code from the Jupyter notebook to a standard python file.

Deliverables:

1. Implementation of one or more extensions/variations of the basic MNIST diffusion model. For each, motivate your choice and briefly explain the theoretical background for the extension. Sketch what changes were necessary to make in the template code to make this work (you can do this in the appendix if you run out of space).

2. Conduct experiments with your models, assessing their performance. Describe potential failure modes that you observe during training/generation.

3. Compare the performance of the models to the basic model. Motivate your choice of metrics for these comparisons.

Note that Part A of the project allows you to set your ambition level to match your group size. If your group has 2 members, you are not expected to do more than 1-2 variation/extension. On the other hand, if there are 4 members in your group, we expect you to explore 2-4 different variations/extensions (depending on the difficulty of each of them).

B Function fitting with noisy inputs

Consider the case where we have a regression problems with a dataset $\mathcal{D}$ given by pairs (x_i, y_i), $i = 1, \ldots, \ell$, but we have uncertainty on both x_i and y_i. A case where this happens is if we want to model the volume of a liquid as a function of its temperature, where both quantities are measured. Thus, the true value of the input is $x_i^{\text{Truth}} = x_i - \Delta_i$ for some unknown value of Δ_i. We assume that the generative model of y_i is

$$y_i = f(x_i - \Delta_i) + \epsilon_i,\ \Delta_i \sim \mathcal{N}(0, \sigma_x^2),\ \epsilon_i \sim \mathcal{N}(0, \sigma_y^2)\ . \tag{1}$$

Here, $\sigma_x^2 = 0.01$ and $\sigma_y^2 = 0.0025$. This is a hard problem to solve, when Δ_i is unknown.

As ground truth function to learn, we employ:

$$f(x) = -x^2 + 2\frac{1}{1 + \exp(-10x)} \tag{2}$$

Training data is provided in the file `data_part_B.csv` where x_i is the first, y_i the second and Δ_i the third column. We further provide some code to implement multivariate normal distributions including some transformations that are helpful for this implementation. Further we provide code outlining how to sample arbitrary distributions using pyro.

B.1 Fitting a standard Gaussian Process

We will first consider the simpler problem of fitting a standard Gaussian Process assuming no uncertainty in x_i, i.e., $\Delta_i = 0$ and f follows a GP prior. This will form your baseline.

1. Select a suitable kernel. Make sure that the kernel is twice continuously differentiable (this will be necessary for part B.2). Identify the parameters of the model and decide which kernel parameters are fixed and which are variable. We will refer to the variable parameters as θ. Implement the marginal log-likelihood $\log p(y|X, \theta, \sigma_y^2)$.

2. Compute the maximum marginal log likelihood estimate of the variable parameters (θ and possible σ_y^2), for example using grid-search, of the following variations of the GP model:

   (a) σ_y^2 is unknown and must be fit as well. X is given by the noisy x_i

   (b) $\sigma_y^2 = 0.0025$ is fixed. X is given by the noisy x_i

   (c) $\sigma_y^2 = 0.0025$ is fixed. X is given by the true x_i^{Truth}

3. On an equally spaced grid of 100 values $x \in [-1, 1]$ plot the mean and 95% confidence interval of the posterior $p(f|x, \mathcal{D}, \theta, \sigma_y^2)$, that is, the posterior estimate of f at grid location x after conditioning on the dataset using the parameters found in step 2. (Hint: what is the difference between predicting f and y?)

Deliverables:

D1 A description of the kernel used and a description of the fixed and variable parameters as well as your optimization approach.

D2 A plot for each model containing the posterior mean and 95% confidence interval computed on the grid, the true function values, equation (2) on the grid, and a scatter plot of the data (use x_i^{Truth} for the scatter plot of the last model).

D3 Report the best kernel parameters found for the 3 models and report the found variance for model (a). Discuss the obtained fit and the quality of the models. Name differences, discuss the effect of the different parameters on the chosen kernel and discuss possible hypotheses for what could cause the differences.

B.2 Modeling Gaussian Process Regression with noisy inputs

We will now include uncertainty into the model. As directly using equation (1) is difficult, we will use an approximation. Assuming that Δ_i is small, we can use a Taylor expansion to approximate $f(x^{\text{truth}} - \Delta) \approx f(x) - \Delta f'(x)$ and assume that the generative model follows the approximation

$$y_i = f(x_i) - \Delta_i f'(x_i) + \epsilon_i$$

Assuming that f follows some GP prior and assuming that the kernel k is twice differentiable, it is possible to also obtain a joint prior over f and f' as:

$$\begin{bmatrix} f \\ f' \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K(X) & K_1(X)^T \\ K_1(X) & K_2(X) \end{bmatrix} \right) \ ,$$

where $K(X)_{ij} = k(x_i, x_j)$ is the usual kernel matrix, $K_1(X)_{ij} = \frac{\partial}{\partial x_i} k(x_i, x_j)$ and $K_2(X)_{ij} = \frac{\partial^2}{\partial x_i \partial x_j} k(x_i, x_j)$.

1. Compute the derivatives of your chosen kernel and compute the joint prior of $(f, f')|X$. Obtain a joint sample of $(f, f')|X$ on a grid of 100 equally spaced values X with samples $X_i \in [-1, 1]$. To verify correctness of f', use the samples of f to estimate f' using central differences.

$$f'_i \approx \tilde{f}'_i = \frac{f(X_{i+1}) - f(X_{i-1})}{X_{i+1} - X_{i-1}}, \ i = 2, \ldots, 99 \ .$$

2. Derive the mean and covariance matrix of the multivariate normal distribution of $p(y|X, \Delta, \theta, \sigma_y^2)$, that is, the value of y conditioned on a given value of the vector of measurement errors Δ. To simplify the derivation, you can define D as the diagonal matrix with $D_{ii} = \Delta_i$. Using the values of Δ_i given in the file, compute the maximum marginal log-likelihood estimate of $p(y|X, \Delta, \theta, \sigma_y^2)$ assuming that θ and σ_y^2 are unknown. Plot the posterior $p(f|x, \mathcal{D}, \Delta, \theta, \sigma_y^2)$ as in B.1.3. Compare your results to the ones obtained with the models from B.1. (Hint: for predictions at new points, take $\Delta = 0$)

3. (*High Ambition – e.g. for larger groups*) With Δ following the prior stated above and using your fitted kernel parameters from part B.2.2, use NUTS to sample from $p(\Delta|X, y, \theta, \sigma_y^2, \sigma_x^2)$, that is, the posterior distribution of Δ given the observations, where X is given by the x_i. Use Arviz to ensure that chains have converged. Provide a scatter plot of the pair of values (Δ_9, Δ_{10}) and compare to their true values $(-0.25, 0.25)$. Use the sampled values of Δ to estimate mean and variance of the marginal posterior $p(f|x, \mathcal{D}, \theta, \sigma_y^2)$.

Deliverables:

D4 Provide the derivatives of your chosen kernel and a derivation of the mean and covariance matrix of $p(y|X, \Delta, \theta, \sigma_y^2)$.
(*High Ambition*) A derivation of the mean and variance estimates of the marginal posterior predictive using samples of Δ.

D5 Provide a plot for the model(s) containing the posterior mean and 95% confidence interval computed on the grid, the true function values on the grid, and a scatter plot of the data. State the values of the optimized kernel parameters.
(*High Ambition*) Provide a scatter plot of (Δ_9, Δ_{10})

D6 Compare your results to the ones obtained in B1. Compare the fits and discuss the individual up and downsides of the approaches. What needs to be changed in the methodology to make this model usable in practice?