Exercise Sheet #11

Due: 29.01.2024 11h00

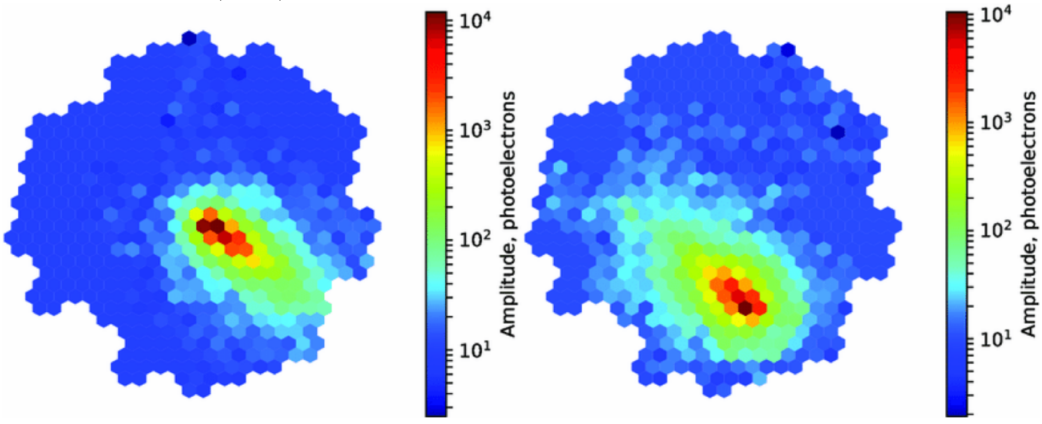
Jan Pawlowski, Tilman Plehn
Jonas Spinner

General Regulations.

- Please hand in your solutions in groups of three people. A mix of attendees from different tutorials is fine. We will not correct submissions from single students.
- Your solutions to theoretical exercises can be either handwritten notes (scanned to `pdf`), typeset using LaTeX, or directly in the jupyter notebook using Markdown.
- For the practical exercises, the data and a skeleton for your jupyter notebook are available at https://github.com/heidelberg-hepml/mlph2023-Exercises. Always provide the (commented) python code as well as the output, and don't forget to explain/interpret the latter, we do not give points for code that does not run. Please hand in both the notebook (`.ipynb`) and an exported pdf. Combine your the pdfs from theoretical and notebook exercises into a single pdf.
- Submit all your files in the Übungsgruppenverwaltung, only once for your group of three. Please list all names and tutorial numbers to simplify things for us.
- Starting from this sheet we put the tag **exam-like** to some exercises to give you a felling for how exercises in the exam might look like.

1 Cherenkov Telescope

A Cherenkov telescope gives images such as the ones below. The task is to differentiate gamma rays (left)from background particles (right).



Bychkov, Igor et al. (2018). Russian–German Astroparticle Data Life Cycle Initiative. Data. 3. 56. 10.3390/data3040056.

(a) (exam-like) What kind of architecture would you use? Argue briefly why it would be a good choice. (1 pts)

(b) (exam-like) Argue briefly which loss function you would use here, and give its formula. (1 pts)

(c) (exam-like) List three tricks that you would use to train this network successfully from a limited size training set. You want, in particular, to avoid overfitting. Give a one-sentence description or a formula for each of the tricks you use. (2 pts)

2 Event Generation with Diffusion Models

After building some intuition for how diffusion models work in exercise 1 on the last sheet, we will now focus on using diffusion models for LHC event generation. We consider two classes of diffusion models, DDPMs and CFM. You can find more information about those in the lecture notes which are based on a recent application of these models to LHC event generation[1] as well as in the original papers.

(a) (exam-like) Explain how the concept of diffusion can be used to construct a generative neural network. Discuss pros and cons of diffusion models compared to other generative architectures. (2 pts)

Denoising Diffusion Probabilistic Models (DDPMs)[2] are a class of discretized diffusion models, i.e. the diffusion process has a finite number of steps. This is the same model as in sheet 10, exercise 1.

(b) A formal ingredient for diffusion models is that all intermediate variables x_t are normally distributed. Following the notation in Section 5.5.1 of the lecture notes, check that DDPMs satisfy this condition by proving Eqn. (5.80)

$$p(x_t|x_0) = \int dx_1 \cdots dx_{t-1} \prod_{i=1}^{t} p(x_i|x_{i-1}) = \mathcal{N}(x_t; \sqrt{1-\bar{\beta}_t}x_0, \bar{\beta}_t)$$

$$\text{with} \qquad p(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t), \qquad 1-\bar{\beta}_t \equiv \prod_{i=1}^{t}(1-\beta_i).$$

Hint: The following relation might be helpful

$$\int dx_i \, \mathcal{N}(x_{i+1}; \sqrt{1-\beta_{i+1}}x_i, \beta_{i+1}) \, \mathcal{N}(x_i; \sqrt{1-\beta_i}x_{i-1}, \beta_i)$$
$$= \mathcal{N}\Big(x_{i+1}; \sqrt{1-\beta_{i+1}}\sqrt{1-\beta_i}x_{i-1}, 1-(1-\beta_{i+1})(1-\beta_i)\Big).$$

(2 pts)

(c) Fill the gaps in the given DDPM implementation and train the model on a toy dataset. (1 pts)

(d) Train the DDPM on the event generation dataset. You can directly use the improved preprocessing of sheet 9, exercise 3. Visualize the distributions of the generated events. (2 pts)

Another highly expressive generative network architecture is Conditional Flow Matching[3]. The diffusion process in this model is continuous, i.e. there is an infinite number of diffusion steps and one has to integrate over them to sample from the model. This architecture is very young, it was proposed in late 2022.

(e) Fill the gaps in the given CFM implementation and train the model on a toy dataset. (1 pts)

(f) Train the CFM on the event generation dataset. You can again directly use the improved preprocessing of sheet 9, exercise 3. Visualize the distributions of the generated events. (2 pts)

[1] https://arxiv.org/pdf/2305.10475.pdf
[2] https://arxiv.org/pdf/2006.11239.pdf
[3] https://arxiv.org/pdf/2210.02747.pdf

3 Unfolding with Conditional INNs

Over the last sheets we have learned about different generative architectures. However, in practice one often wants to generate events based on certain conditions to get more control over the generation process. Examples are the generation of an image conditioned on a prompt, the generation of a translated text conditioned on the original text or generating the detector response of a particle conditioned on its energy. It is straight-forward to extend the architectures studied on the last few sheets by adding extra inputs to the respective networks.

After the hard process, LHC events undergo a series of processes like parton showering and detector simulation that are well-understood, but strongly modify the distribution of events. Inverting or *unfolding* these effects is challenging, but rewarding as it simplifies the comparison of results across experiments and reduces the effort required for analyses. In this exercise we will train an INN conditioned on detector-level events to generate the underlying parton-level events. Conditional generative networks are very well-suited for this application, as they allow statistically meaningful unfolding even for single events.[4]

(a) For this exercise we will use a toy detector simulation as implemented in the notebook. Apply this detector simulation to the $Z \to \mu^+\mu^-$ dataset and preprocess the resulting events with a simple standardization. For the parton-level events, apply the same preprocessing as in the previous exercises on event generation. Then construct a dataloader that returns pairs of parton-level and detector-level events. (2 pts)

(b) Construct a conditional INN and train it on the unfolding dataset. Unfold the full detector-level distribution by unfolding each event in the test set once. Compare the resulting distribution of unfolded events to the true parton-level distribution using the same plotting code as for the other event generation exercises. (2 pts)

(c) In contrast to classical unfolding methods, conditional generative networks can also unfold single events in a statistically well-defined way. Unfold the first 10 events in the test set 1000 times and visualize the distribution of unfolded events compared to the true event for the energy of the first muon. (2 pts)

[4] https://arxiv.org/pdf/2006.06685.pdf