# Predicting LQCD probability densities with machine learning methods

Reinhold Kaiser

Institute for Theoretical Physics - University of Frankfurt

30/05/2025
Lattice Journal Club - SS 2025



# Outline

# Motivating the application of ML methods to LQCD data

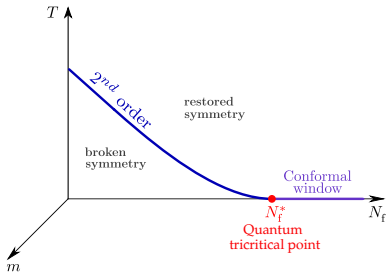



Fig. from (Klinger, Kaiser, and Philipsen 2025)

Challenging tasks:

- huge parameter space to sample with Monte Carlo (MC) simulations $\{\beta, am, N_\sigma, N_\tau, N_\mathrm{f}, \mu_i\}$
- required extrapolations: $am \to 0$, $N_\sigma \to \infty$, $N_\tau \to \infty$
- lattice bulk transition at large $N_\mathrm{f}$- and small $\beta$-values, see JP Klinger's talk on 13.06.2025
- critical slowing down near phase transitions

## Our current project:

- determine order of chiral phase transition as function of $N_\mathrm{f}$ up to conformal window
- possibly find the onset of the conformal window $N_\mathrm{f}^*$

## Joint project with group from Bielefeld:

Check if ML models can reliably predict LQCD observables based on lattice paramers as input.

# ML methods as an alternative for multiple histogram reweighting

- PhD project by M. Neumann: train ML model on MC data to interpolate in $\beta$
- $\beta$-reweighting with small statistics: smoothly connecting data points

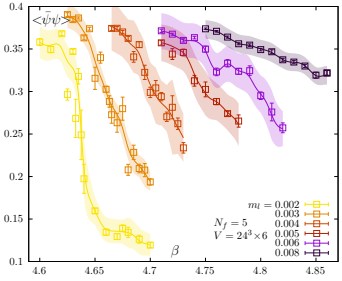
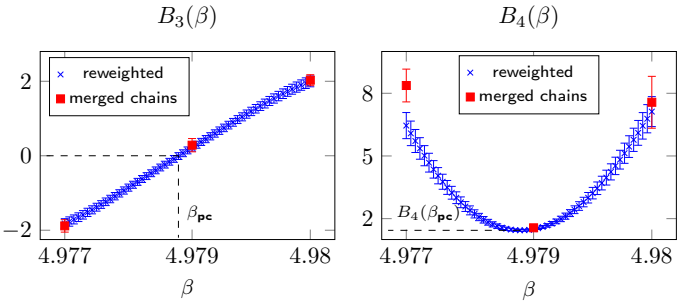



Fig. from (Neumann 2023)

- Jonas Schaible's talk on 20.06.25 about "Reweighting methods for Monte Carlo data"
- exact in the limit of infinite statistics
- only parameters for which conjugated energy in the action is known



# Outline of the ML approach by M. Neumann

**Goal**

- Localize the phase transition in $\beta$ and
- classify the transition with respect to its order in $am$

using machine learning methods.

Possible benefits:

- take all data into account to interpolate observables
- extrapolate to small values of $am$
- interpolate in $N_\sigma$

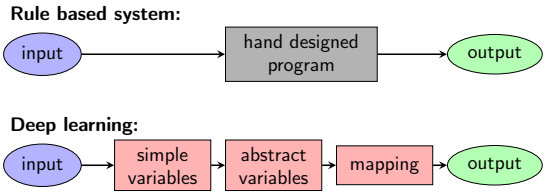
1. simulate several $am$-values at several $\beta$-values each for two volumes on $N_\tau = 6$ lattices. Measure gauge action $S_G$ and $\langle \bar{\psi}\psi \rangle$
2. train MAF model to learn probability distribution of $(\langle \bar{\psi}\psi \rangle, S_G)$
3. extract $\beta_{\mathrm{pc}}$ from $\langle \bar{\psi}\psi \rangle$ histograms
4. train ViT model to classify images of transitions with respect to their order
5. extract prediction of $am_{\mathrm{c}}$, which separates 1. order and crossover regions

# Literature

- Neuman: PhD thesis - Thermodynamics of 5-flavor QCD (Neumann 2023)
- Erdmann, Glombitza, Kasieczka, Klemradt: Deep Learning for Physics Research (Erdmann et al. 2021)
- Papamakarios, Pavlakou, Murray: Masked Autoregressive Flow for Density Estimation (Papamakarios, Pavlakou, and Murray 2017)

# Basics of ML methods

Data analysis:

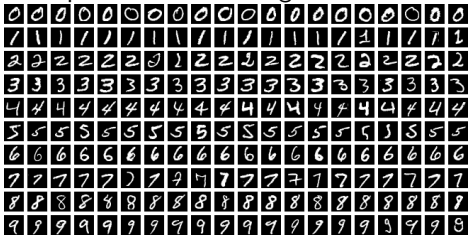



- deep learning is a subcategory of machine learning
- machine learning may be beneficial for problems with many observables

## Task of ML methods

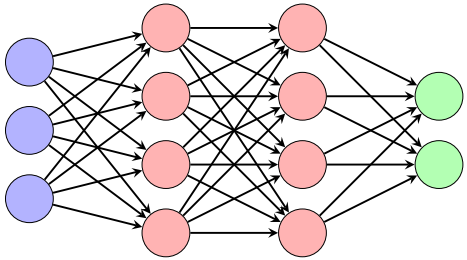
Find a function $f(\boldsymbol{x};\boldsymbol{\theta})$ to predict some output $\boldsymbol{y}$ based on the input $\boldsymbol{x}$. Learn the optimal parameters $\boldsymbol{\theta}$ based on the data set $\{\boldsymbol{x}_i\}$.

Example: handwritten digits



$$f\left(\quad ;\boldsymbol{\theta}\right) = 8$$

# The concept of deep learning



input layer hidden layers output layer

Transformations in neuron:

- linear transformations with weight matrix $\boldsymbol{W}$ and bias $\boldsymbol{b}$
- non-linear mapping: activation function $\sigma$
- $\sigma$ acts element-wise on its argument vector

Output of $i$-th layer:

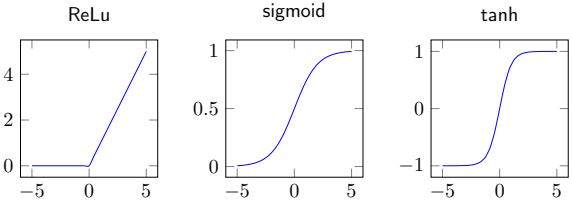
$$\boldsymbol{y}^{(i)} = \sigma\left(\boldsymbol{W}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}^{(i)}\right)$$

Examples of activation functions:

$$\sigma_{\text{ReLu}}(x) = x \cdot \Theta(x)$$

$$\sigma_{\text{sigmoid}}(x) = \frac{1}{1+e^{-x}}$$

$$\sigma_{\tanh}(x) = \tanh(x)$$



# The loss function

- scalar measure $\mathcal{L}$ of distance from prediction to true value
- is minimized with respect to the model parameters $\boldsymbol{W}$ and $\boldsymbol{b}$
- supervised learning: data $\boldsymbol{x}_i$ has labels $\boldsymbol{z}_i$
- or true probability $\boldsymbol{p}_i$ given for estimated $\boldsymbol{q}_i$

- 3 data sets: training, validation, testing
- difference of $\mathcal{L}$ evaluated on training and validation set: **generalization gap**
- in the end: testing set used to evaluate model after hyper-parameter tuning

## Mean-squared-error for regression

$$\mathcal{L} = \frac{1}{k} \sum_{i=1}^{k} [f(\boldsymbol{x}_i) - \boldsymbol{z}_i]^2$$

## Cross-entropy for classification

$$\mathcal{L} = -\frac{1}{k} \sum_{i=1}^{k} \sum_{j=1}^{m} p_{j,i} \log q_{j,i}$$

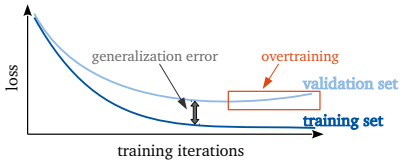



Fig. taken from (Glombitza 2022)

# Backpropagation

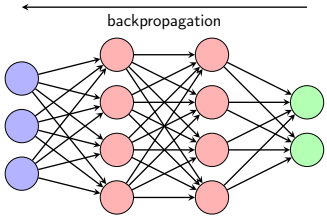
- to minimize the loss function gradients are calculated

$$\frac{\partial \mathcal{L}}{\partial W^{(i)}_{j,k}}, \quad \frac{\partial \mathcal{L}}{\partial b^{(i)}_{j}}$$

- chain rule used to traverse through the network backwards
- update parameters along steepest gradient descent with learning rate $\alpha$

$$W_{t+1} = W_t - \alpha \left\langle \frac{\partial \mathcal{L}}{\partial W} \right\rangle_t$$

$$b_{t+1} = b_t - \alpha \left\langle \frac{\partial \mathcal{L}}{\partial b} \right\rangle_t$$




$$\boldsymbol{y}^{(i)} = \sigma\left(\boldsymbol{W}^{(i)}\boldsymbol{y}^{(i-1)} + \boldsymbol{b}^{(i)}\right)$$

- stochastic gradient descent: take random subset of data for update: **(mini-)batch**
- **epoch**: one-time use of all training data

# General procedure to train a ML model

1. preprocess data (e.g. normalization, zero-centered)
2. split data into training, validation and test set
3. train network on training set, optimize $\boldsymbol{W}, \boldsymbol{b}$
4. check generalization gap with validation set
5. tune hyper-parameters of the network (e.g. number of layers, activation function, etc.)
6. repeat steps 3. to 5.
7. test the final model on the test set

# Learning probability distributions

- unsupervised learning: training data without labels
- generative models: generate new samples similar to training data
- find the underlying probability distribution of a data set $\{\boldsymbol{x}_i\}$

**Task**

- train a model to approximate the true distribution $p_r$ with $p_\theta$.
- generate new samples $\tilde{\boldsymbol{x}} \sim p_\theta$

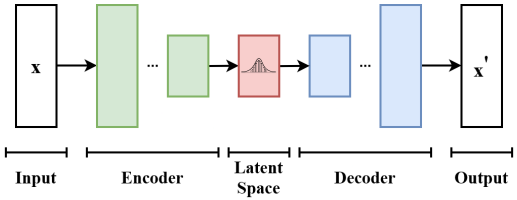



Fig. from (EugenioTL 2025)

- find lower dimensional representation of data
- encode abstract features in latent space

**Kullback-Leibler divergence**

$$\mathcal{D}_{\mathrm{KL}}(p||q) = \int \mathrm{d}x\, p(x) \log\left(\frac{p(x)}{q(x)}\right)$$

# Normalizing Flows

## Idea

Transform a simple distribution using invertible and differential mappings to obtain a complex distribution.

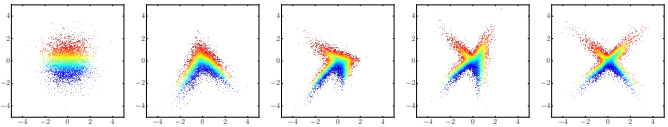


Fig. from (Papamakarios, Nalisnick, et al. 2021)

## Definition

A normalizing flow describes the transformation of a probability density through a sequence of invertible mappings. (Rezende and Mohamed 2015)

- use multi-dimensional Gaussian as simple distribution
- latent space has equal dimensionality as data due to invertibility
- density of data point can be calculated with model

# Transformations of the normalizing flow

- model represents invertible, smooth mapping

$$f : \mathbb{R}^d \to \mathbb{R}^d$$

- inverse function $g = f^{-1}$
- simple distribution $p(\boldsymbol{z})$ in latent space
- distribution of data $p(\boldsymbol{x})$
- new samples are generated through $\tilde{\boldsymbol{x}} = g(\boldsymbol{z})$

**Training goal**

Transform the distribution of the training data $\boldsymbol{x}$ into a Gaussian using $f(\boldsymbol{x})$.

- transformation of probability from $\boldsymbol{z}$ to $\boldsymbol{x}$

$$p(\boldsymbol{x}) = p(f(\boldsymbol{x}))|\det \boldsymbol{J}_f(\boldsymbol{x})|$$

with $f(\boldsymbol{x}) = \boldsymbol{z}$

- construction of model as a composition of $N$ bijective functions

$$f = f_1 \circ \cdots \circ f_{N-1} \circ f_N$$

- simple calculation of Jacobian

$$|\det \boldsymbol{J}_f| = \prod_{i=1}^{N} |\det \boldsymbol{J}_{f_i}|$$

# Introduction to masked autoregressive flows

- joint probability as a composition of conditional probabilities

$$p(\boldsymbol{x}) = \prod_{d=1}^{D} p(\boldsymbol{x}_d | \boldsymbol{x}_{<d})$$

- for a network to represent proper distribution: **autoregressive property**

**Autoregressive property**

Each output $\boldsymbol{y}_d$ only depends on the previous input units $\boldsymbol{x}_{<d}$ and not on $\boldsymbol{x}_{\geq d}$.

- use a mask on fully connected network to achieve autoregressive property

Masked autoencoder for distribution estimation (MADE)

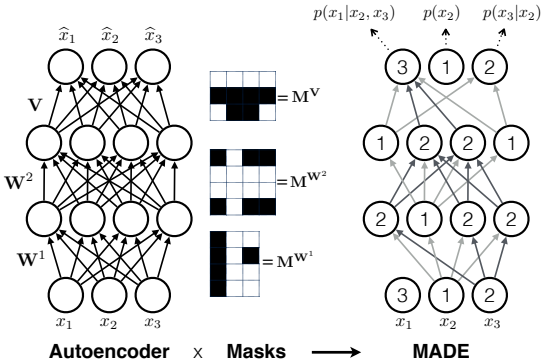



Fig. from (Germain et al. 2015)

# Autoregressive flows

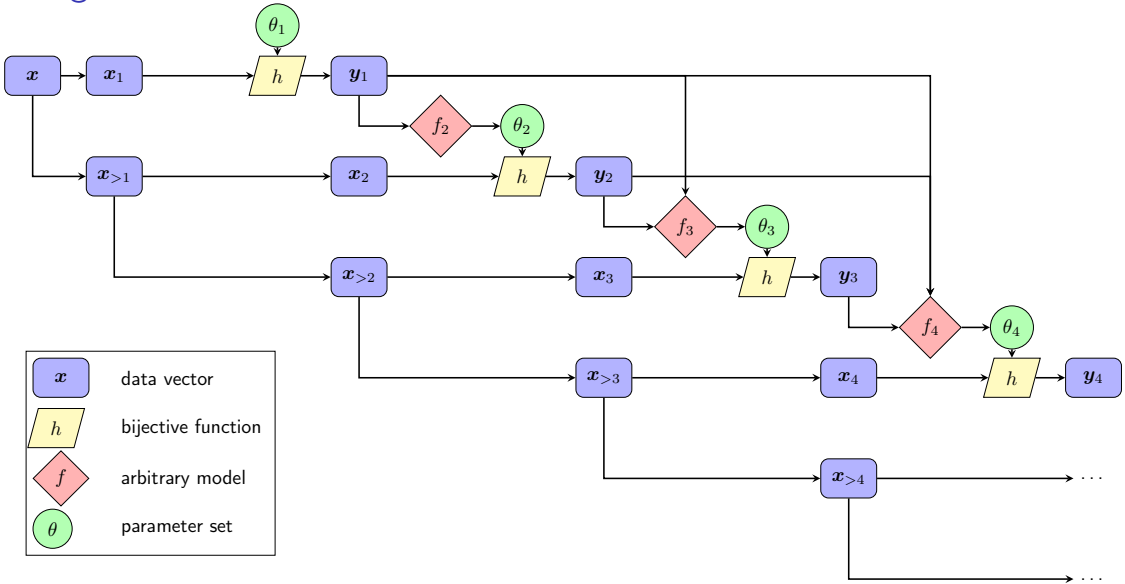
- invertible network
- bijective function $\boldsymbol{x} \mapsto \boldsymbol{y}$
- autoregressive property for probability density evaluation
- each output $\boldsymbol{y}_t$ only depends on the previous $\boldsymbol{x}_t$

Defining the autoregressive flow (Erdmann et al. 2021)

$$
\begin{aligned}
\boldsymbol{y}_1 &= h(\boldsymbol{x}_1, \theta_1) \\
\boldsymbol{y}_2 &= h(\boldsymbol{x}_2, \theta_2(\boldsymbol{y}_1)) \\
\boldsymbol{y}_3 &= h(\boldsymbol{x}_3, \theta_3(\boldsymbol{y}_1, \boldsymbol{y}_2)) \\
&\dots \\
\boldsymbol{y}_t &= h(\boldsymbol{x}_t, \theta_t(\boldsymbol{y}_1, \dots, \boldsymbol{y}_{t-1}))
\end{aligned}
$$

- $h$ is bijective function parameterized by $\theta_t$
- triangular Jacobian and simple calculation of determinant

# Autoregressive flows



# Masked autoregressive flow (MAF)

- autoregressive model with conditionals

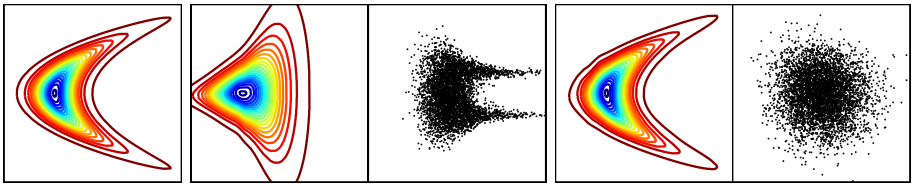
$$p(x_i|\boldsymbol{x}_{<i}) = \mathcal{N}(x_i|\mu_i, \sigma_i^2),$$

  where $\mu_i = f_{\mu_i}(\boldsymbol{x}_{<i})$ and $\sigma_i^2 = f_{\sigma_i^2}(\boldsymbol{x}_{<i})$.

- generate data with

$$x_i = z_i \cdot \sqrt{\sigma_i^2} + \mu_i, \text{ where } \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$

- $\boldsymbol{z}$ is alternative representation, can be obtained by inverse transformation
- functions $f_{\mu_i}$, $f_{\sigma_i^2}$ are implemented using MADE blocks
- conditional density estimation $p(\boldsymbol{x}|\boldsymbol{y})$: $\boldsymbol{y}$ is input for every layer



(a) Target density (b) MADE with Gaussian conditionals (c) MAF with 5 layers

Fig. from (Papamakarios, Pavlakou, and Murray 2017)

# Application of MAF to the chiral condensate

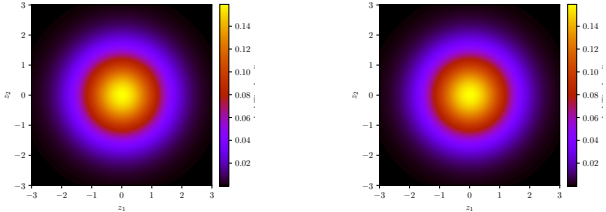
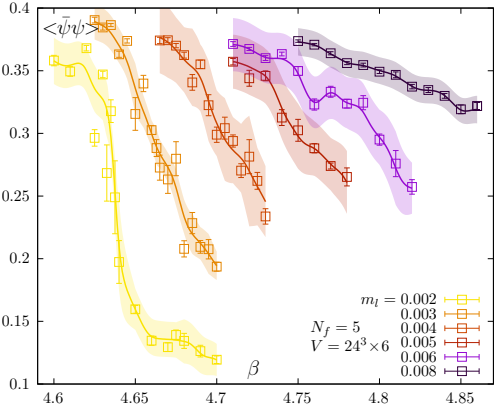
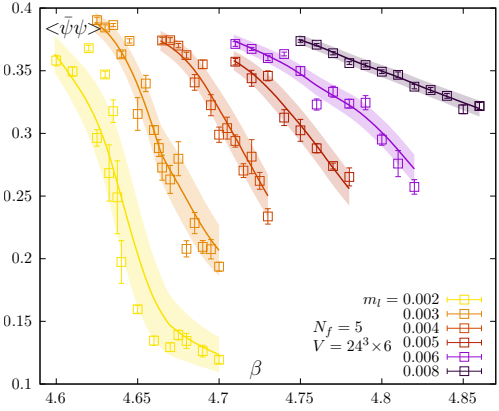



Fig. from (Karsch et al. 2023)

# Results of the chiral condensate interpolations




multiple histogram reweighting in $\beta$

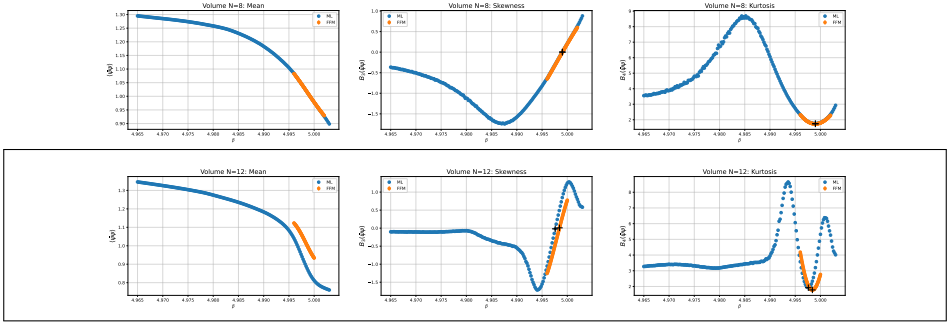



MAF interpolation trained on all data

Fig. from (Neumann 2023)

# Higher order cumulants of $\langle \bar{\psi}\psi \rangle$

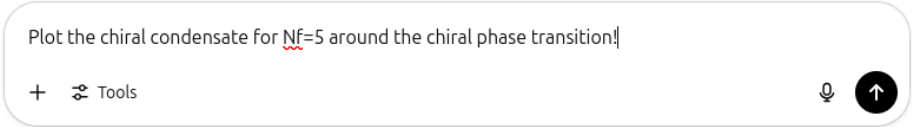
Interpolating the $\langle \bar{\psi}\psi \rangle$ in $N_\sigma$




Figs. courtesy of JP

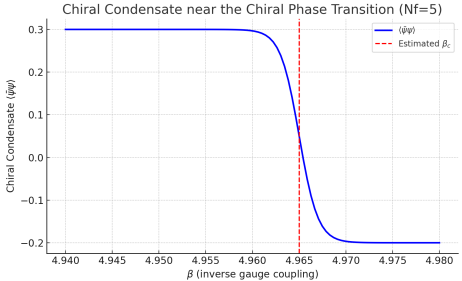
# Concluding Remarks

- ML is a complex optimization problem with potentially many local minima
- challenge: physics laws need to apply to generated samples
- true probability distribution might be part of model: no prescription how to find it
- learning the probability distribution away from the training data is challenging
- extrapolation to distant parameter values is problematic


Plot the chiral condensate for Nf=5 around the chiral phase transition!
Tools

Plot the chiral condensate for Nf=5 around the chiral phase transition!

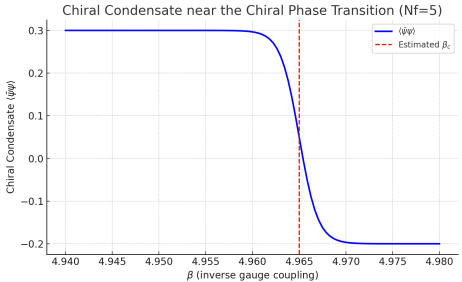
+ Tools




(*ChatGPT* 2025)

Plot the chiral condensate for Nf=5 around the chiral phase transition!

+ Tools




(*ChatGPT* 2025)

# Thanks!

# Bibliography I


- *ChatGPT* (2025). URL: https://chatgpt.com/ (cit. on pp. 23–25).
- Erdmann, Martin et al. (Feb. 2021). *Deep Learning for Physics Research*. World Scientific Publishing Co. Pte. Ltd. DOI: 10.1142/12294 (cit. on pp. 6, 16).
- EugenioTL (2025). *Variational autoencoder - Wikipedia — en.wikipedia.org*. [Accessed 30-05-2025]. URL: %7Bhttps://en.wikipedia.org/wiki/Variational_autoencoder#/media/File:VAE_Basic.png%7D (cit. on p. 12).
- Germain, Mathieu et al. (Feb. 2015). "MADE: Masked Autoencoder for Distribution Estimation". In: arXiv: 1502.03509 [cs.LG] (cit. on p. 15).
- Glombitza, Jonas (2022). "Deep-learning based measurement of the mass composition of ultra-high energy cosmic rays using the surface detector of the Pierre Auger Observatory". PhD thesis (cit. on p. 9).
- Karsch, Frithjof et al. (2023). "A machine learning approach to the classification of phase transitions in many flavor QCD". In: *PoS* LATTICE2022, p. 027. DOI: 10.22323/1.430.0027. arXiv: 2211.16232 [hep-lat] (cit. on p. 19).

# Bibliography II


- Klinger, Jan Philipp, Reinhold Kaiser, and Owe Philipsen (2025). "The order of the chiral phase transition in massless many-flavour lattice QCD". In: *PoS* LATTICE2024, p. 172. DOI: 10.22323/1.466.0172. arXiv: 2501.19251 [hep-lat] (cit. on p. 3).
- Neumann, Marius (2023). "Thermodynamics of 5-flavor QCD". PhD thesis. Bielefeld U. DOI: 10.4119/unibi/2983242 (cit. on pp. 4, 6, 20).
- Papamakarios, George, Eric Nalisnick, et al. (2021). "Normalizing Flows for Probabilistic Modeling and Inference". In: *J. Machine Learning Res.* 22.1, pp. 2617–2680. DOI: 10.5555/3546258.3546315. arXiv: 1912.02762 [stat.ML] (cit. on p. 13).
- Papamakarios, George, Theo Pavlakou, and Iain Murray (May 2017). "Masked Autoregressive Flow for Density Estimation". In: arXiv: 1705.07057 [stat.ML] (cit. on pp. 6, 18).
- Rezende, Danilo Jimenez and Shakir Mohamed (May 2015). "Variational Inference with Normalizing Flows". In: arXiv: 1505.05770 [stat.ML] (cit. on p. 13).