# Simulation-based inference and generative neural networks. Early explorations.

Sofya (Sofi) Dymchenko, Bruno Raffin

DFKI-Inria ENGAGE project meeting

July 4-5, 2022

# Introduction

## Who am I

Background:

- 2019, BSc in Mathematics, Higher School of Economics, Moscow, Russia
- 2021, MSc in Data Science, Skoltech & HSE, Moscow, Russia
- wide experience in deep learning (audio, images, generative models)
- interested in probability, bayesian DL, generative models, HPC+DL, math of DL...

Current:

- PhD student at Inria and UGA supervised by Bruno Raffin
- started 4 months ago
- high-performance online deep learning models trained on synthetic data (keywords, yes)

## Motivations

DeepMelissa:

framework for training deep learning models on synthetic data (on-the-fly).

Questions:

1. data is serialized : how to overcome inductive bias?
   $\rightarrow$ how to give training points to NN (replay-buffer)?
2. data is not finite: how to overcome bad exploration of global minima?
   $\rightarrow$ how to control training of NN (learning rate)?
3. data is high-dimensional: how to get good generalization of NN fast?
   $\rightarrow$ how to get to know probability space of simulator (probabilistic programming)?

## Motivations

DeepMelissa:

framework for training deep learning models on synthetic data (on-the-fly).

Questions:

1. data is serialized : how to overcome inductive bias?
   $\rightarrow$ how to give training points to NN (replay-buffer)?
2. data is not finite: how to overcome bad exploration of global minima?
   $\rightarrow$ how to control training of NN (learning rate)?
3. data is high-dimensional: how to get good generalization of NN fast?
   $\rightarrow$ how to get to know probability space of simulator (probabilistic programming)?

## Motivations

DeepMelissa:

framework for training deep learning models on synthetic data (on-the-fly).

Questions:

1. data is serialized : how to overcome inductive bias?
   $\rightarrow$ how to give training points to NN (replay-buffer)?
2. data is not finite: how to overcome bad exploration of global minima?
   $\rightarrow$ how to control training of NN (learning rate)?
3. data is high-dimensional: how to get good generalization of NN fast?
   $\rightarrow$ how to get to know probability space of simulator (probabilistic programming)?

## Motivations

DeepMelissa:

framework for training deep learning models on synthetic data (on-the-fly).

Questions:

1. data is serialized : how to overcome inductive bias?
   $\rightarrow$ how to give training points to NN (replay-buffer)?
2. data is not finite: how to overcome bad exploration of global minima?
   $\rightarrow$ how to control training of NN (learning rate)?
3. data is high-dimensional: how to get good generalization of NN fast?
   $\rightarrow$ how to get to know probability space of simulator (probabilistic programming)?

## Motivations

DeepMelissa:

framework for training deep learning models on synthetic data (on-the-fly).

Questions:

1. data is serialized : how to overcome inductive bias?
   $\rightarrow$ how to give training points to NN (replay-buffer)?
2. data is not finite: how to overcome bad exploration of global minima?
   $\rightarrow$ how to control training of NN (learning rate)?
3. data is high-dimensional: how to get good generalization of NN fast?
   $\rightarrow$ how to get to know probability space of simulator (probabilistic programming)?

## Motivations

Goal: being in (some kind of) full probabilistic control of simulator in order to train NN efficiently.

NN can be trained for some DL task that uses simulator's data, specifically it can be **surrogate model that mimics simulator**.

# SBI

## What is SBI

**Simulation Based Inference [1]**

*Simulation-based* – data comes from simulator

*Inference* – getting parameters of distribution from data

## What is SBI

**Simulation Based Inference [1]**

*Simulation-based* – data comes from simulator

*Inference* – getting parameters of distribution from data

## What is SBI

### Simulation Based Inference [1]

*Simulation-based* – data comes from simulator
*Inference* – getting parameters of distribution from data

## Problem statement

*Simulator* – computer program $f : \theta \to X$, where $\theta$ is a vector of input parameters, which describes a mechanistic model (e.g. for CFD: size of tube, density of ink).

What we actually want is to use a simulator not as a black box but as a probabilistic model and to learn its distributions.

## Problem statement

*Simulator* – computer program $f : \theta \rightarrow X$, where $\theta$ is a vector of input parameters, which describes a mechanistic model (e.g. for CFD: size of tube, density of ink).

What we actually want is to use a simulator not as a black box but as a probabilistic model and to learn its distributions.

## Problem Statement of SBI

Infer $\theta$ from $X_{obs}$ – posterior $P(\theta|X_{obs}) = ?$

Known/proposed prior $P(\theta)$. Bayes theorem: $P(\theta|X_{obs}) \sim P(X_{obs}|\theta)P(\theta)$ ?

Problem: likelihood $P(X|\theta)$ – unknown / intractable / impossible to compute, because of simulator nature!

## Problem Statement of SBI

Infer $\theta$ from $X_{obs}$ – posterior $P(\theta|X_{obs}) = ?$

Known/proposed prior $P(\theta)$. Bayes theorem: $P(\theta|X_{obs}) \sim P(X_{obs}|\theta)P(\theta)$ ?

Problem: likelihood $P(X|\theta)$ – unknown / intractable / impossible to compute, because of simulator nature!

## Problem Statement of SBI

Infer $\theta$ from $X_{obs}$ – posterior $P(\theta|X_{obs}) = ?$

Known/proposed prior $P(\theta)$. Bayes theorem: $P(\theta|X_{obs}) \sim P(X_{obs}|\theta)P(\theta)$ ?

Problem: likelihood $P(X|\theta)$ – unknown / intractable / impossible to compute, because of simulator nature!

## Traditional approaches

### Traditional approaches

- ABC (1984, 2002). Approximate Bayesian Computation:
  $\theta_i \sim p(\theta), x_{sim} \sim p(\cdot|\theta_i)$, if $\mathrm{dist}(x_{obs}, x_{sim})$ small then $\theta_i$ is from posterior.
- DE (1984). Density estimation methods:
  estimate distribution with histograms or KDE using a lot of data;

Disadvantages:

curse of dimensionality

amortization

low-dimensional stats

poorly scales to HD

sample inefficiency

bad quality of inference

## New directions

Expansion of SBI toolbox by three forces:

- neural networks for probabilistic models (2015+)
- active learning - guide a solver
- internal integration with a solver

## Neural network approaches

Conditional neural density estimator - parametric model $q_\phi$ controlled by a set of parameters $\phi$ (weights of NN), which:

- takes a pair of data points $(u, v)$
- outputs a conditional probability density $q_\phi(u|v)$
- trains by optimizing $\sum_{n=1}^{N} \log q_\phi(u_n|v_n) \to \max_\phi$
- learns approximate conditional $p(u|v)$ (with flexible model, enough training data)

# NN Methods

## Methods

**It is all about Bayes**

$$p(\theta|x) \propto p(X|\theta)p(\theta)$$

SNLE [2]:
learning likelihood
$p(X|\theta)$

SNPE[3]:
learning posterior
$p(\theta|X)$

SNRE[4]:
learning
likelihood-ratio
$p(X|\theta_0)/p(X|\theta_1)$

SNVI[5]:
learning
likelihood(-ration) in
variational setting
(optimizing ELBO
with GNN)

# Learning likelihood (2019)

$X_{obs}$ - observed data
Estimator $q_w(x|\theta)$ – neural network (normalizing flow)
Set prior $p(\theta)$
Set approximate of posterior $p'_0(\theta|x_{obs})$ as prior
In every round:

1. sample N parameter vectors from last round approximate of posterior
2. get N simulations with this parameters
3. train NN on all the data (from previous rounds and current)
4. set posterior approximation as product of NN-likelihood on observed data and prior

**Algorithm 1** APT with per-round proposal updates

**Input:** simulator with (implicit) density $p(x|\theta)$, data $x_o$, prior $p(\theta)$, density family $q_\psi$, neural network $F(x,\phi)$, simulations per round $N$, number of rounds $R$.

$\tilde{p}_1(\theta) := p(\theta)$
**for** $r = 1$ **to** $R$ **do**
  **for** $j = 1$ **to** $N$ **do**
    Sample $\theta_{r,j} \sim \tilde{p}_r(\theta)$
    Simulate $x_{r,j} \sim p(x|\theta_{r,j})$
  **end for**
  $\phi \leftarrow \underset{\phi}{\operatorname{argmin}} \sum_{i=1}^{r} \sum_{j=1}^{N} -\log \tilde{q}_{x_{i,j},\phi}(\theta_{i,j})$ using (2)
  $\tilde{p}_{r+1}(\theta) := q_{F(x_o,\phi)}(\theta)$
**end for**
**return** $q_{F(x_o,\phi)}(\theta)$

## Learning posterior (2019)

Approximate directly posterior (rounds)

1. propose prior
2. automatically update
3. set posterior=prior (active learning concept)
4. converge

**Algorithm 1:** Sequential Neural Likelihood (SNL)

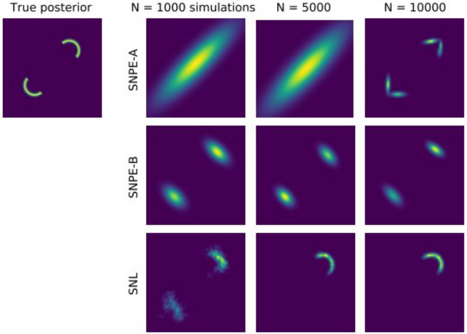
**Input** : observed data $\mathbf{x}_o$, estimator $q_\phi(\mathbf{x} \mid \boldsymbol{\theta})$, number of rounds $R$, simulations per round $N$

**Output** : approximate posterior $\hat{p}(\boldsymbol{\theta} \mid \mathbf{x}_o)$

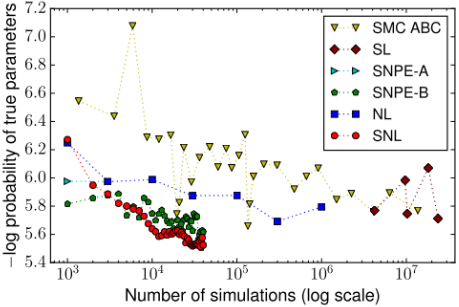
set $\hat{p}_0(\boldsymbol{\theta} \mid \mathbf{x}_o) = p(\boldsymbol{\theta})$ and $\mathcal{D} = \{\}$
**for** $r = 1 : R$ **do**
  **for** $n = 1 : N$ **do**
    sample $\boldsymbol{\theta}_n \sim \hat{p}_{r-1}(\boldsymbol{\theta} \mid \mathbf{x}_o)$ with MCMC
    simulate $\mathbf{x}_n \sim p(\mathbf{x} \mid \boldsymbol{\theta}_n)$
    add $(\boldsymbol{\theta}_n, \mathbf{x}_n)$ into $\mathcal{D}$
  (re-)train $q_\phi(\mathbf{x} \mid \boldsymbol{\theta})$ on $\mathcal{D}$ and set $\hat{p}_r(\boldsymbol{\theta} \mid \mathbf{x}_o) \propto q_\phi(\mathbf{x}_o \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})$
**return** $\hat{p}_R(\boldsymbol{\theta} \mid \mathbf{x}_o)$

# Methods comparison

SNPE

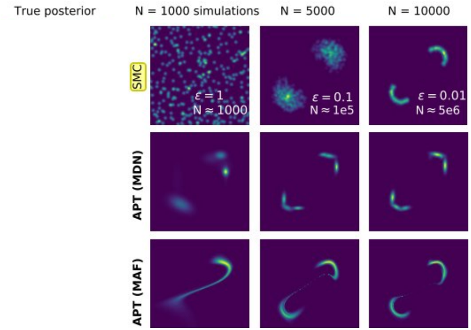



SNLE

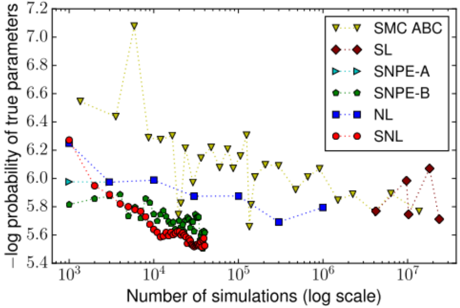


# Methods comparison

SNPE



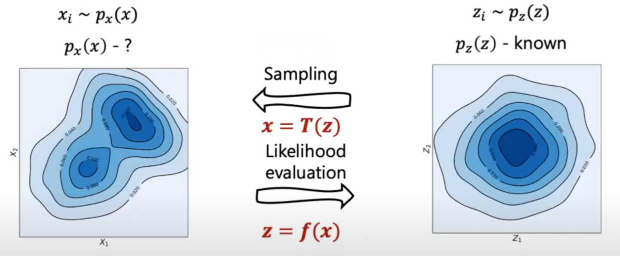

SNLE



# Normalizing Flows

## What generative model is used in both for approximating densities?

Normalizing flows.
Estimate complex distribution by map from latent space, e.g. $\mathcal{N}(0,1)$.




$$p_x(x_i) = p_z(f(x_i)) \left| \det \frac{\partial f(x_i)}{\partial x_i} \right|$$

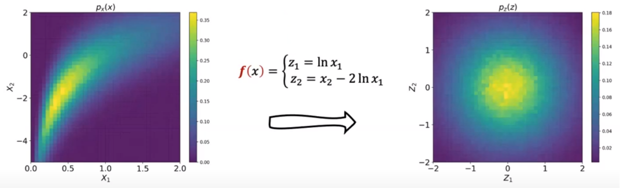
Generative model:

- likelihood evaluation $z = f(x)$
- sampling procedure $x = T(z)$

## What generative model is used in both?

Normalizing flows .

Estimate complex distribution by map from latent space, e.g. $\mathcal{N}(0,1)$.




$$p_x(x_i) = \frac{1}{2\pi} \exp(-\frac{1}{2}((\log x_1)^2 + \\ +(x_2 - 2\log x_1)^2)) * \frac{1}{x_1}$$

Generative model:

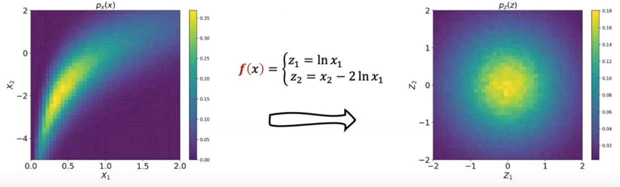
- likelihood evaluation $z = f(x)$
- sampling procedure $x = T(z)$

Can we have $T(z) = f^{-1}(z)$? Problem statement: find $f(x)$

## What generative model is used in both?

Normalizing flows .

Estimate complex distribution by map from latent space, e.g. $\mathcal{N}(0,1)$.

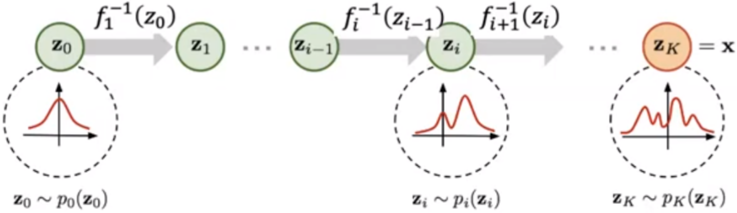



$$p_x(x_i) = \frac{1}{2\pi}\exp(-\frac{1}{2}((\log x_1)^2 + +(x_2 - 2\log x_1)^2)) * \frac{1}{x_1}$$

Generative model:

- likelihood evaluation $z = f(x)$
- sampling procedure $x = T(z)$

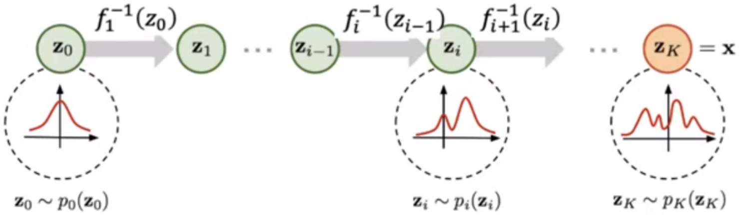
Can we have $T(z) = f^{-1}(z)$? Problem statement: find $f(x)$




$$p_x(x) = p_{z_0}(f_K(\dots(f_1(x)))) \left|\det \frac{\partial f_1(z_1)}{\partial z_1}\right| \dots \left|\det \frac{\partial f_K(z_k)}{\partial z_k}\right|.$$

To compute easily - low-triangular/block-triangular.

UPD Problem statement: how?




$$p_x(x) = p_{z_0}(f_K(\dots(f_1(x)))) \left|\det \frac{\partial f_1(z_1)}{\partial z_1}\right| \dots \left|\det \frac{\partial f_K(z_k)}{\partial z_k}\right|.$$

To compute easily - low-triangular/block-triangular.

UPD Problem statement: how?

## MAF: masked autoregressive flow

**MAF** [6]

All $z_i$ changes: low-triangular matrix. And $\mu_k s_k$ – neural networks.
Likelihood evaluation is fast.

$$z = f(x) = \begin{cases} z_1 = (x_1 - \mu_1)\exp(-s_1) \\ \dots \\ z_k = (x_k - \mu_k(x_{1:k-1})) \odot \exp(-s_k(x_{1:k-1})) \\ \dots \end{cases}$$

## Low-triangular matrix

$$\frac{\partial f(x)}{\partial x} = \begin{pmatrix} \exp(-s_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{\partial z_D}{\partial x_1} & \cdots & \exp(-s_D(x_{1:D-1})) \end{pmatrix}$$

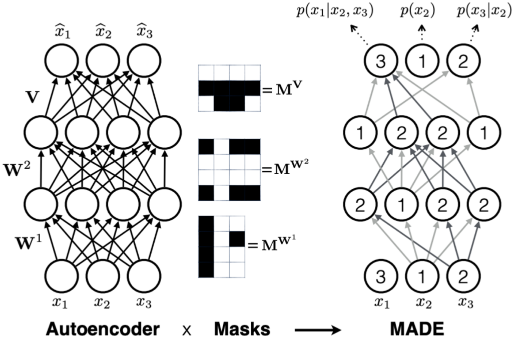
Denis Derkach

## Jacobian:

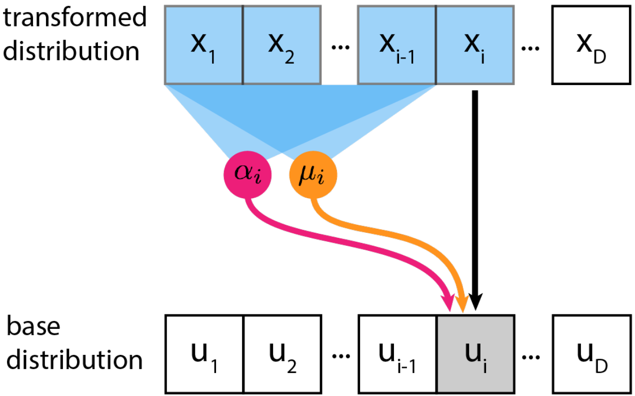
$$\left|\det \frac{\partial f(x)}{\partial x}\right| = \exp(-\sum_{j=1}^{D} s_d(x_{1:d-1}))$$

## MAF

1) Chain rule for sequential data (mesh/timesteps)
2) Masked NN



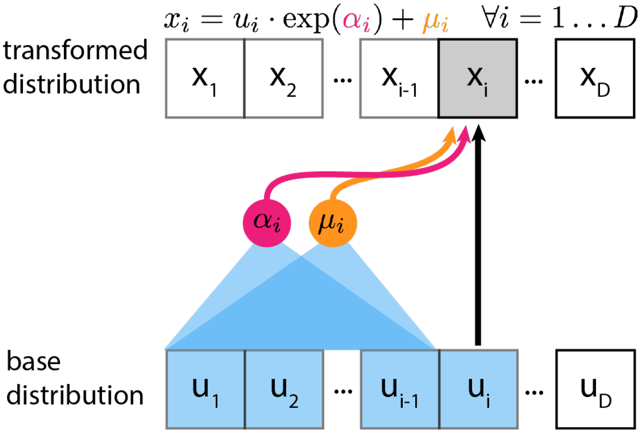
## Density estimation with MAF




$$u_i = (x_i - \mu_i) \cdot \exp(-\alpha_i) \quad \forall i = 1 \ldots D$$

Compute $\mu$ and $s$, evaluate $z$.
Fast, parallelizable.

## Sampling with IAF




All the $x_i$ can be computed in a single pass of D threads working in parallel.

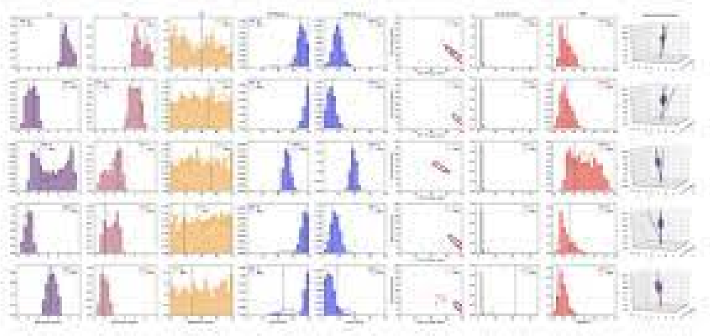
## MAF+IAF

| | Base distribution | Target distribution | Model | Data generation | Density estimation |
|---|---|---|---|---|---|
| MAF | $\mathbf{z} \sim \pi(\mathbf{z})$ | $\mathbf{x} \sim p(\mathbf{x})$ | $x_i = z_i \odot \sigma_i(\mathbf{x}_{1:i-1}) + \mu_i(\mathbf{x}_{1:i-1})$ | Sequential; slow | One pass; fast |
| IAF | $\tilde{\mathbf{z}} \sim \tilde{\pi}(\tilde{\mathbf{z}})$ | $\tilde{\mathbf{x}} \sim \tilde{p}(\tilde{\mathbf{x}})$ | $\tilde{x}_i = \tilde{z}_i \odot \tilde{\sigma}_i(\tilde{\mathbf{z}}_{1:i-1}) + \tilde{\mu}_i(\tilde{\mathbf{z}}_{1:i-1})$ | One pass; fast | Sequential; slow |

# Discussion

## Last week update

ETALUMIS [7]: large-scale simulator as a probabilistic program



## Future direction

1. Usually SBI experiments are low-scale
   $\longrightarrow$ Can we scale these to high-dimensional data (time-series meshes)?
2. Usually SBI is used directly for inverse problem on observations
   $\longrightarrow$ Can we use computed likelihood/posterior as part of framework to learn some NN on simulations in order to have control on data to simulate?
3. Concepts on active learning, probabilistic view on simulators, using normalizing flows seems prospective and interesting, should be done more state-of-the-art literature review in this direction.

## Future direction

1. Usually SBI experiments are low-scale
   $\longrightarrow$ Can we scale these to high-dimensional data (time-series meshes)?
2. Usually SBI is used directly for inverse problem on observations
   $\longrightarrow$ Can we use computed likelihood/posterior as part of framework to learn some NN on simulations in order to have control on data to simulate?
3. Concepts on active learning, probabilistic view on simulators, using normalizing flows seems prospective and interesting, should be done more state-of-the-art literature review in this direction.

## Future direction

1. Usually SBI experiments are low-scale
   ⟶ Can we scale these to high-dimensional data (time-series meshes)?
2. Usually SBI is used directly for inverse problem on observations
   ⟶ Can we use computed likelihood/posterior as part of framework to learn some NN on simulations in order to have control on data to simulate?
3. Concepts on active learning, probabilistic view on simulators, using normalizing flows seems prospective and interesting, should be done more state-of-the-art literature review in this direction.

## Future direction

1. Usually SBI experiments are low-scale
   $\longrightarrow$ Can we scale these to high-dimensional data (time-series meshes)?
2. Usually SBI is used directly for inverse problem on observations
   $\longrightarrow$ Can we use computed likelihood/posterior as part of framework to learn some NN on simulations in order to have control on data to simulate?
3. Concepts on active learning, probabilistic view on simulators, using normalizing flows seems prospective and interesting, should be done more state-of-the-art literature review in this direction.

## References I


[1] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences of the United States of America*, 117(48):30055–30062, 2020.

[2] George Papamakarios, David C. Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. *AISTATS*, 2020.

[3] David S. Greenberg, Marcel Nonnenmacher, and Jakob H. Macke. Automatic posterior transformation for likelihood-free inference. *ICML*, 2019.

[4] Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with amortized approximate ratio estimators. *ICML*, 2020.

[5] Manuel Glöckler, Michael Deistler, and Jakob H Macke. Variational Methods For Simulation-Based Inference. *ICLR*, 2022.

## References II

[6] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *NeurIPS*, 2017.

[7] Atilim Güne Baydin, Lei Shao, Wahid Bhimji, Lukas Heinrich, Lawrence Meadows, Jialin Liu, Andreas Munk, Saeid Naderiparizi, Bradley Gram-Hansen, Gilles Louppe, Mingfei Ma, Xiaohui Zhao, Philip Torr, Victor Lee, Kyle Cranmer, Prabhat, and Frank Wood. Etalumis: Bringing probabilistic programming to scientific simulators at scale. *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, jul 2019.