Machine Learning for HEP

Lecture II — Bayesian Neural Networks

UNIVERSITÀ
DEGLI STUDI
DI MILANO

BND Graduate School — Blankenberge 2024

Ramon Winterhalder

Lecture II

Bayesian Neural Networks

Reminder — Likelihood-Loss

Training should **optimize** probability that **network parameters** ω describe the **data** x

This is given by: $p(\omega \mid x)$ → Usually intractable ☹

Bayes' Theorem

$$p(\omega \mid x) = \frac{p(x \mid \omega)\, p(\omega)}{p(x)}$$

Likelihood → Known from simulation ☺

Prior

Normalization

Posterior

Reminder — Likelihood-Loss

Bayes' Theorem

$$p(\omega \mid x) = \frac{p(x \mid \omega)\, p(\omega)}{p(x)}$$

Likelihood-loss

Weight regularization

$$\begin{aligned}\mathscr{L} &= -\log p(\omega \mid x) \\ &= -\log p(x \mid \omega) - \log p(\omega) + \text{const}\end{aligned}$$

Regression with Gaussian assumption

with point-wise uncertainty

Heteroscedastic loss

Normalization term

$$\mathscr{L}_{\text{heteroscedastic}} = \sum_i \frac{|f(x_i) - f_\omega(x_i)|^2}{2\sigma(x_i)^2} + \log \sigma(x_i) + \ldots$$

Same uncertainty

$\sigma(x) \to \sigma = \text{const}$

Homoscedastic loss = MSE

$$\mathscr{L}_{\text{MSE}} = \frac{1}{2\sigma} \sum_i |f(x_i) - f_\omega(x_i)|^2 + \ldots$$

Is there more we can do?

Yes!

Bayesian Neural Networks

→ There is nothing really Bayesian about them!

What we have *vs* what we want

So far: NNs are deterministic → **fixed** input x → **fixed** output y

However, in **particle physics:**

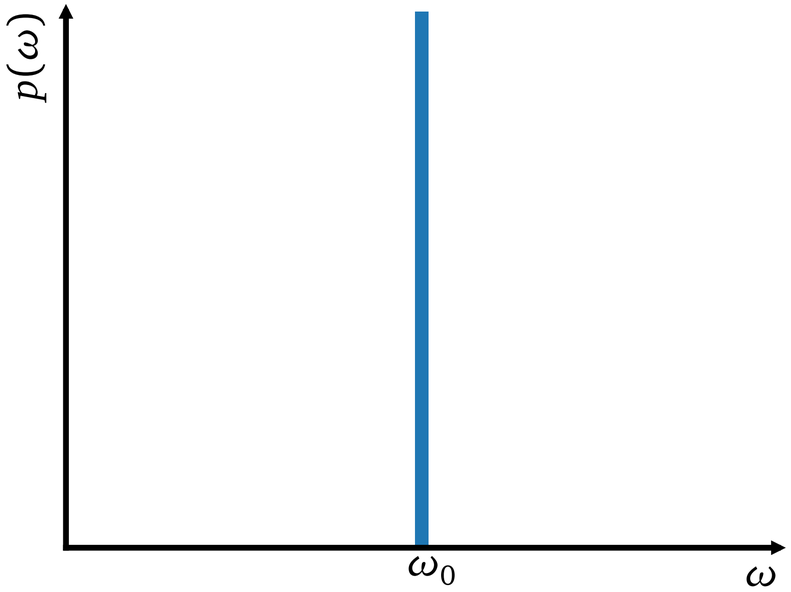
→ we are not only interested in **results**, but also in **errorbars**

We need: **fixed** input x → **probabilistic** output $\langle y \rangle \pm \sigma_y$ → $y \sim p(y \,|\, x)$

How do we achieve this?

Bayesian neural networks

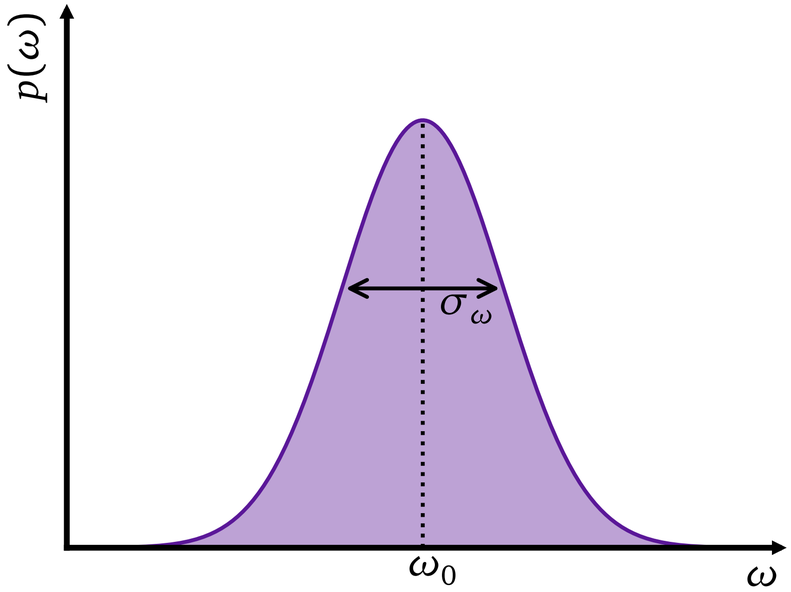
Neural network



Network weights are deterministic

$$\omega = \omega_0$$

Bayesian Neural network



Network weights are drawn from distribution

$$\omega \sim p(\omega)$$

Basics of BNNs

Predictive distribution in Classification

$$p(y\,|\,x) = \int \mathrm{d}\omega\, p(y\,|\,\omega, x)\, p(\omega\,|\,x)$$

Classifier output

→ average over $p(\omega\,|\,x)$ → Intractable ☹

Bayes' theorem

$$p(\omega\,|\,x) = \frac{p(x\,|\,\omega)\, p(\omega)}{p(x)}$$

Need to Calculate normalization

$$p(x) = \int \mathrm{d}\omega\, p(x\,|\,\omega)\, p(\omega)$$

→ Also Intractable ☹

Prediction in Regression

$$\begin{aligned} y(x) \equiv \langle y \rangle &= \int \mathrm{d}y\, p(y\,|\,x)\, y \\ &= \int \mathrm{d}\omega\, p(\omega\,|\,x) \int \mathrm{d}y\, p(y\,|\,x, \omega)\, y \\ &= \int \mathrm{d}\omega\, p(\omega\,|\,x)\ \bar{y}(x, \omega) \end{aligned}$$

Regression output

So what do we do instead?

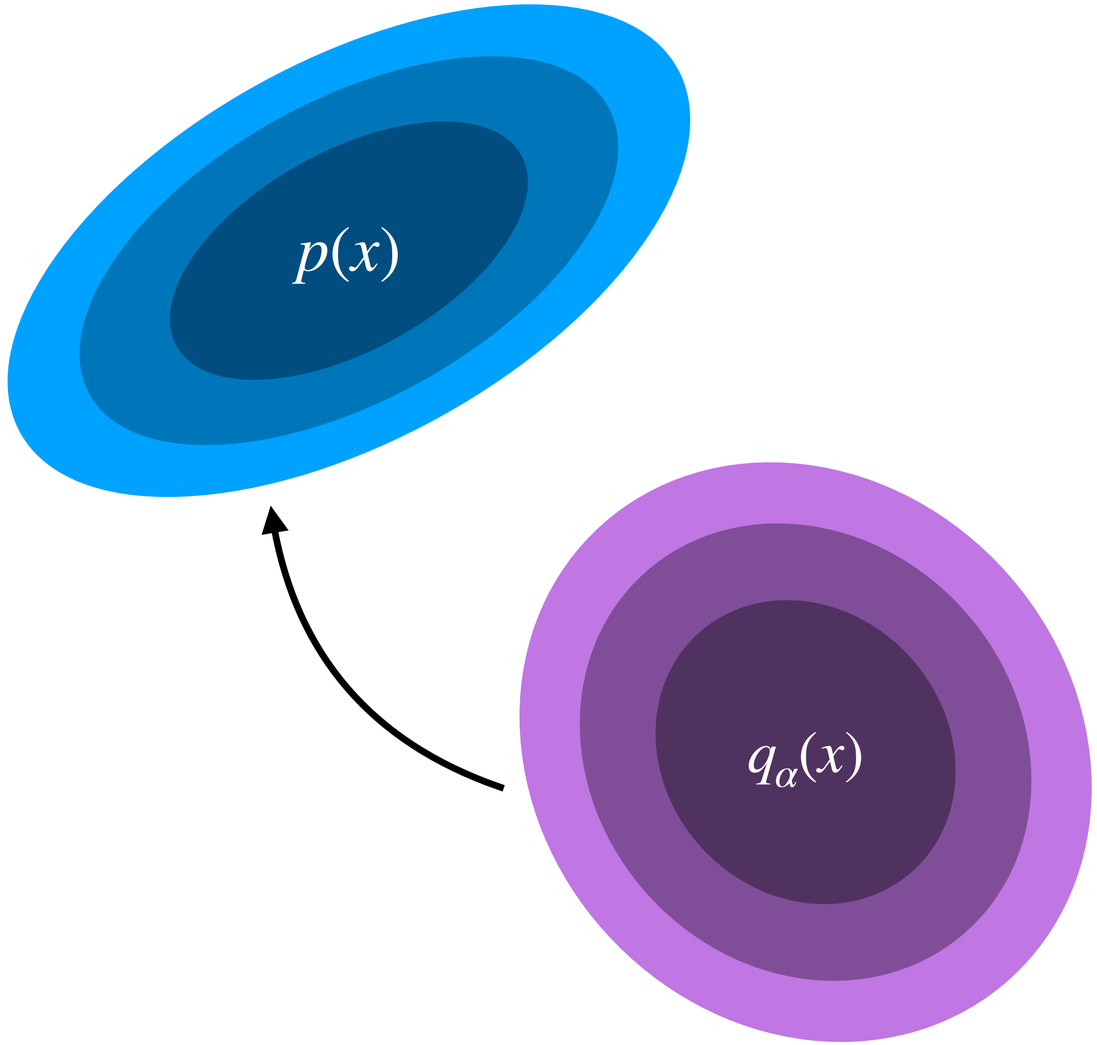
Variational approximation

Approximate posterior

$$p(\omega\,|\,x) \simeq q_\alpha(\omega\,|\,x) \approx q_\alpha(\omega)$$

Simplify (see later)

Parameters α to adjust



Kullback Leibler Divergence

$$\mathrm{KL}(q_\alpha(x), p(x)) = \int \mathrm{d}x\, q_\alpha(x)\, \log\left(\frac{q_\alpha(x)}{p(x)}\right)$$

- Measures information overlap
- Always positive → zero iff $p(x) = q_\alpha(x)$
- Non symmetric

BNN Loss function

Kullback Leibler Divergence

$$\mathrm{KL}(q_\alpha(\omega), p(\omega\,|\,x)) = \int \mathrm{d}x\, q_\alpha(\omega)\, \log\left(\frac{q_\alpha(\omega)}{p(\omega\,|\,x)}\right)$$

$$= \int \mathrm{d}x\, q_\alpha(\omega)\, \log\left(\frac{q_\alpha(\omega)\, p(x)}{p(\omega)\, p(x\,|\,\omega)}\right)$$

$$= \mathrm{KL}(q_\alpha(\omega), p(\omega)) - \int \mathrm{d}\omega\, q_\alpha(\omega)\, \log p(x\,|\,\omega) + \log p(x) \int \mathrm{d}\omega\, q_\alpha(\omega)$$

Intractable norm

$= 1$

BNN loss function

Problem specific

$$\mathscr{L}_{\mathrm{BNN}} = \mathrm{KL}(q_\alpha(\omega), p(\omega)) - \int \mathrm{d}\omega\, q_\alpha(\omega)\, \log p(x\,|\,\omega)$$

2 1

1. Neg log-likelihood averaged over q_α
2. q_α shoud not deviate too much from prior!

How to choose q_α and the prior?

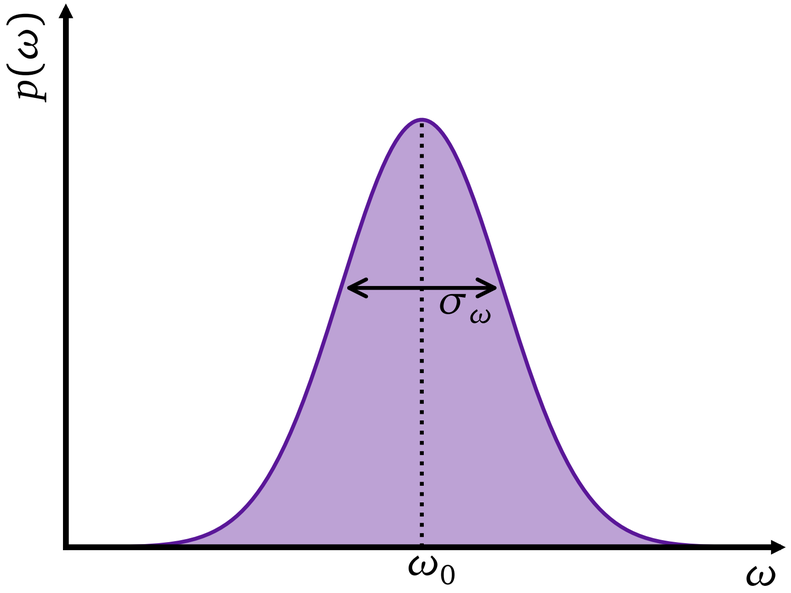
Standard choice

In practice: p, q_α are chosen to be Gaussian

$$q_\alpha(\omega) \equiv q_{\mu,\sigma}(\omega) = \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left[-\frac{(\omega-\mu_q)^2}{2\sigma_q^2}\right]$$

$$p(\omega) = \frac{1}{\sqrt{2\pi\sigma_p^2}} \exp\left[-\frac{(\omega-\mu_p)^2}{2\sigma_p^2}\right]$$

Usually $\mu_p = 0$



KL divergence

$$\mathrm{KL}(q_\alpha(\omega), p(\omega)) = \frac{\sigma_q^2 - \sigma_p^2}{2\sigma_p^2} + \frac{\mu_q^2}{2\sigma_p^2} + \log\frac{\sigma_p}{\sigma_q}$$

[2]

[2] serves as regularization!

Deterministic limit

Gaussian probability

$$q_\alpha(\omega) = \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left[-\frac{(\omega-\mu_q)^2}{2\sigma_q^2}\right]$$

$\sigma_q \to 0$

$\mu_q \to \omega_0$

Delta distribution = fixed weight

$$q_\alpha(\omega) = \delta(\omega - \omega_0)$$

Loss function

$$\mathcal{L}_{\text{BNN}} \to \mathcal{L}_{\text{NN}} = -\log p(x\,|\,\omega_0) + \frac{\omega_0^2}{2\sigma_p} + \text{const}$$

L2 norm: $\lambda\omega_0^2$

→ Usually λ is a free parameter

Mean-field approximation

For K network parameters
we do factorization ansatz:

$$q_\alpha(\omega) = \prod_{i=1}^{K} \mathcal{N}(\omega_i \,|\, \mu_i, \sigma_i)$$

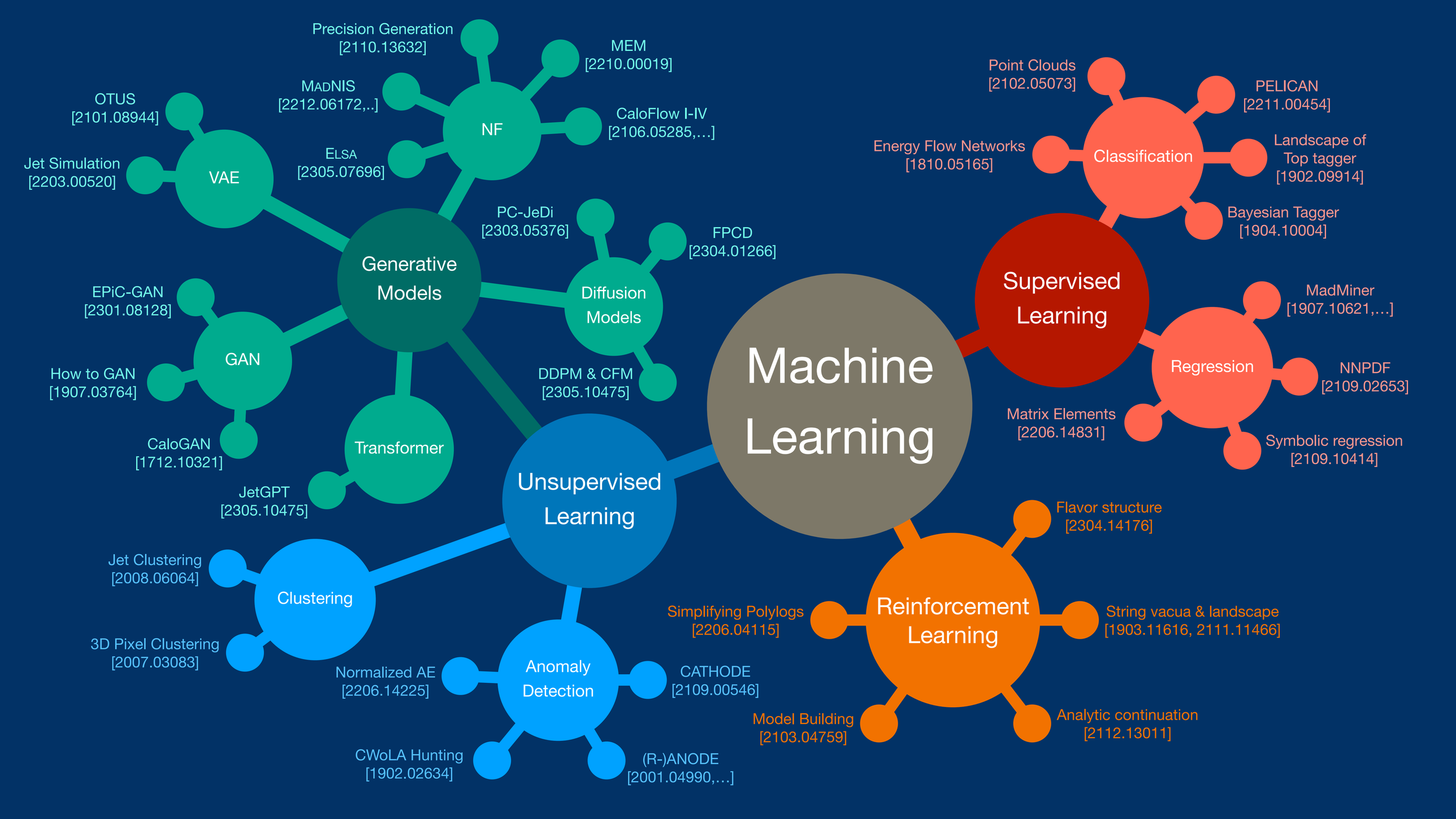
→ BNN has $2K$ parameters → **Mean-field approximation** [1505.05424, 1601.00670]

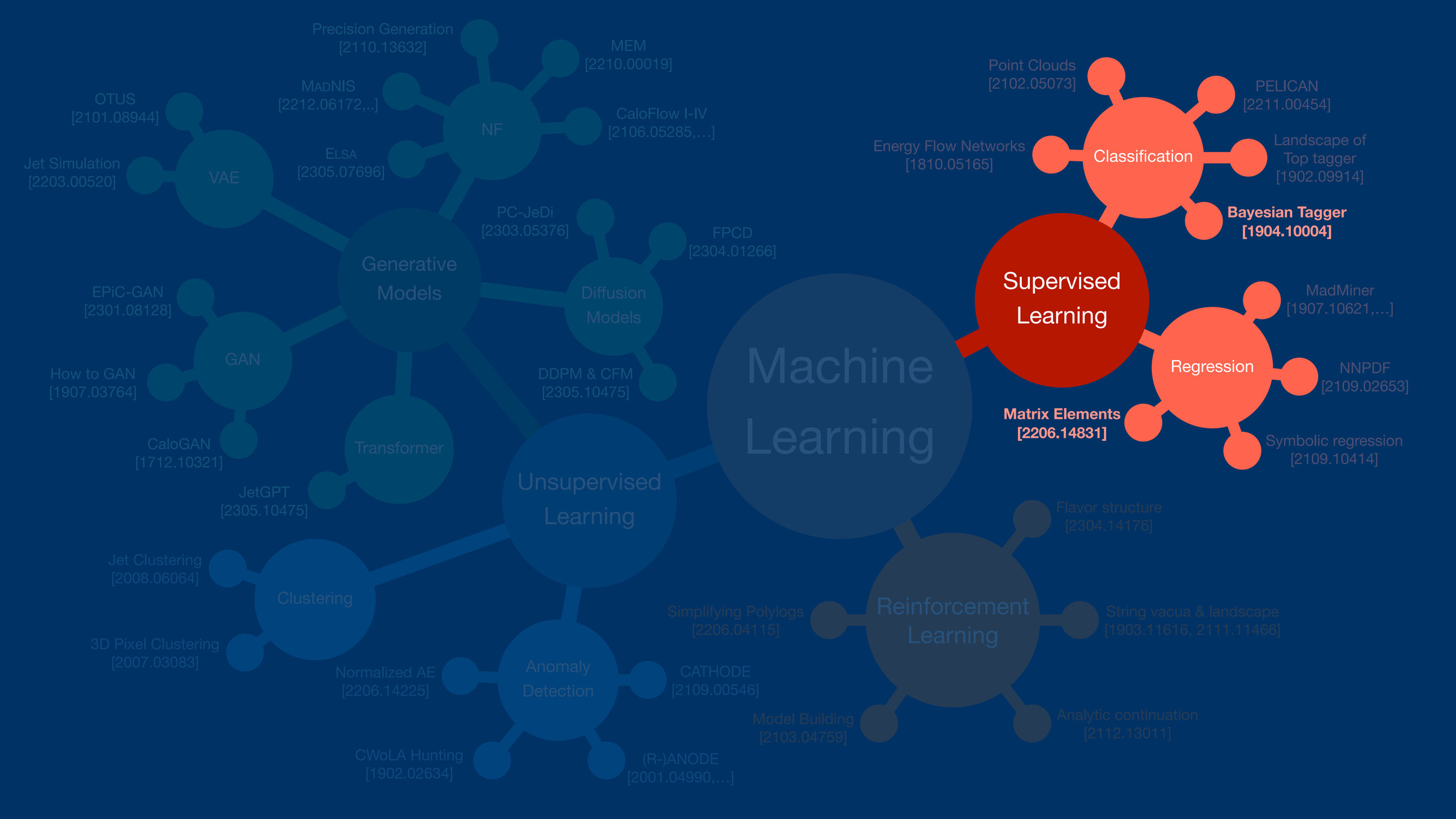
If correlated Gaussians → BNN $\sim K^2$ parameters → bad scaling ☹

→ Deepness of NN compensates approximation?

→ **Open question in ML community!**

Questions?


Machine Learning
Generative Models
VAE
OTUS [2101.08944]
Jet Simulation [2203.00520]
NF
Precision Generation [2110.13632]
MADNIS [2212.06172,..]
ELSA [2305.07696]
MEM [2210.00019]
CaloFlow I-IV [2106.05285,…]
GAN
EPiC-GAN [2301.08128]
How to GAN [1907.03764]
CaloGAN [1712.10321]
Transformer
JetGPT [2305.10475]
Diffusion Models
PC-JeDi [2303.05376]
FPCD [2304.01266]
DDPM & CFM [2305.10475]
Unsupervised Learning
Clustering
Jet Clustering [2008.06064]
3D Pixel Clustering [2007.03083]
Anomaly Detection
Normalized AE [2206.14225]
CWoLA Hunting [1902.02634]
CATHODE [2109.00546]
(R-)ANODE [2001.04990,…]
Supervised Learning
Classification
Point Clouds [2102.05073]
Energy Flow Networks [1810.05165]
PELICAN [2211.00454]
Landscape of Top tagger [1902.09914]
Bayesian Tagger [1904.10004]
Regression
Matrix Elements [2206.14831]
MadMiner [1907.10621,…]
NNPDF [2109.02653]
Symbolic regression [2109.10414]
Reinforcement Learning
Simplifying Polylogs [2206.04115]
Model Building [2103.04759]
Flavor structure [2304.14176]
String vacua & landscape [1903.11616, 2111.11466]
Analytic continuation [2112.13011]

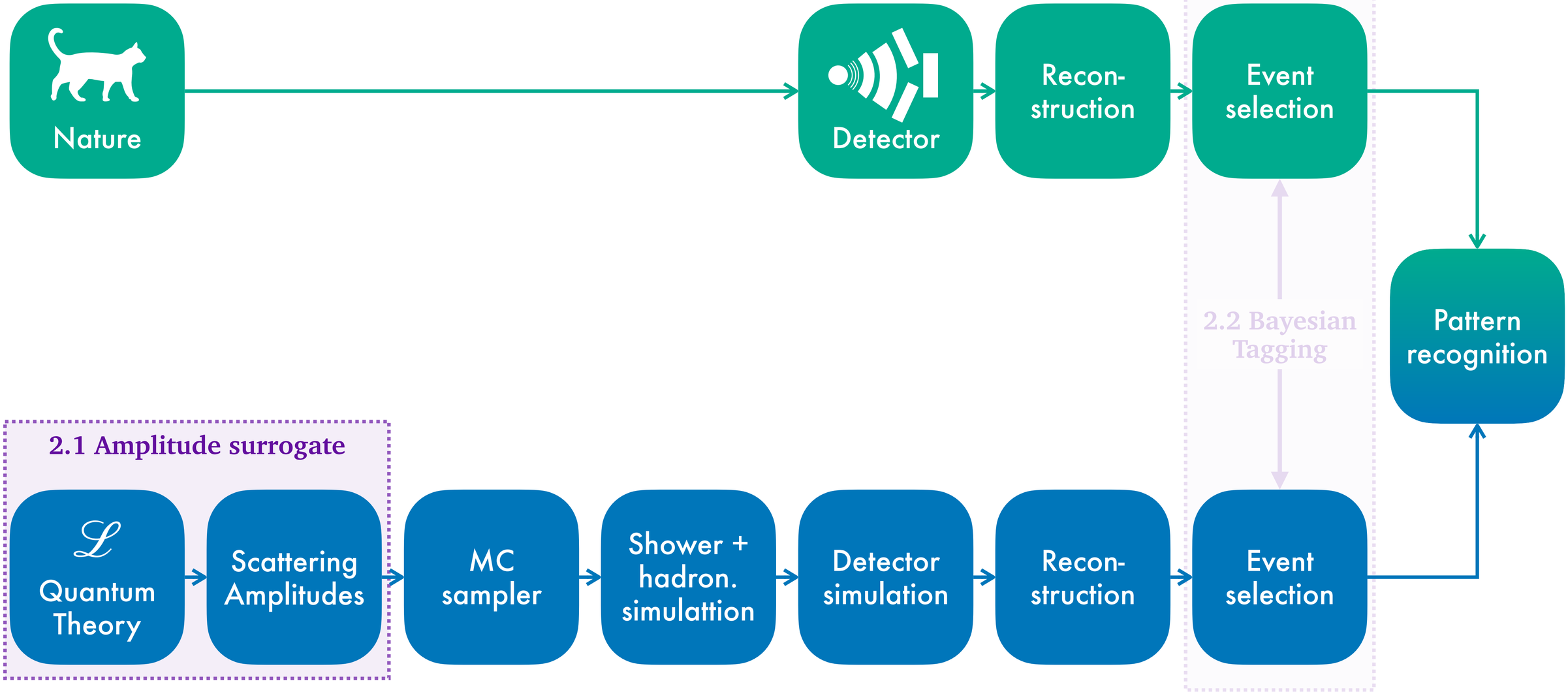

Machine Learning
Generative Models
NF
MEM [2210.00019]
Precision Generation [2110.13632]
MadNIS [2212.06172,..]
CaloFlow I-IV [2106.05285,…]
VAE
OTUS [2101.08944]
Jet Simulation [2203.00520]
Elsa [2305.07696]
Diffusion Models
PC-JeDi [2303.05376]
FPCD [2304.01266]
DDPM & CFM [2305.10475]
GAN
EPiC-GAN [2301.08128]
How to GAN [1907.03764]
CaloGAN [1712.10321]
Transformer
JetGPT [2305.10475]
Unsupervised Learning
Clustering
Jet Clustering [2008.06064]
3D Pixel Clustering [2007.03083]
Anomaly Detection
Normalized AE [2206.14225]
CATHODE [2109.00546]
CWoLA Hunting [1902.02634]
(R-)ANODE [2001.04990,…]
Supervised Learning
Classification
Point Clouds [2102.05073]
PELICAN [2211.00454]
Energy Flow Networks [1810.05165]
Landscape of Top tagger [1902.09914]
Bayesian Tagger [1904.10004]
Regression
MadMiner [1907.10621,…]
NNPDF [2109.02653]
Matrix Elements [2206.14831]
Symbolic regression [2109.10414]
Reinforcement Learning
Flavor structure [2304.14176]
Simplifying Polylogs [2206.04115]
String vacua & landscape [1903.11616, 2111.11466]
Model Building [2103.04759]
Analytic continuation [2112.13011]

Example I

Bayesian Amplitude Regression

Badger, Butter, Luchmann, Pitz, Plehn [2206.14831]

LHC analysis + ML



Bayesian Regression

We want to fit a set of amplitudes $A(x)$

Training data: $\{x, A(x)\}$

We can define: $p(A \mid x)$ ~ prob of A @ position x (→ $p(y \mid x)$ before)

Prediction in regression

$$A(x) \equiv \langle A \rangle = \int \mathrm{d}A \, A \, p(A \mid x)$$

with

$$p(A \mid x) = \int \mathrm{d}\omega \, p(A \mid \omega, x) \, p(\omega \mid x)$$

Variational approximation

$$p(\omega \mid x) \simeq q_\alpha(\omega)$$

BNN

We want to describe mean and stddev of prediction

BNN mean prediction

Mean prediction

$$
\begin{aligned}
A(x) \equiv \langle A \rangle &= \int \mathrm{d}A\, A\, p(A \,|\, x) \qquad (*) \\
&= \int \mathrm{d}A \int \mathrm{d}\omega\, A\, p(A \,|\, \omega, x)\, p(\omega \,|\, x) \\
&= \int \mathrm{d}A \int \mathrm{d}\omega\, A\, p(A \,|\, \omega, x)\, q_\alpha(\omega) \\
&= \int \mathrm{d}\omega\, q_\alpha(\omega) \int \mathrm{d}A\, A\, p(A \,|\, \omega, x) \\
&= \int \mathrm{d}\omega\, q_\alpha(\omega)\, \overline{A}(x, \omega)
\end{aligned}
$$

Variational approximation

network output

What about the uncertainty?

BNN variance of prediction

Variance of prediction

$$\sigma^2_{\text{tot}}(x) \equiv \langle (A - \langle A \rangle)^2 \rangle = \int \mathrm{d}A \, \big(A - \langle A \rangle\big)^2 \, p(A \,|\, x)$$

$$= \int \mathrm{d}A \int \mathrm{d}\omega \, \big(A - \langle A \rangle\big)^2 \, p(A \,|\, \omega, x) \, q_\alpha(\omega)$$

$$= \int \mathrm{d}\omega \, q_\alpha(\omega) \left[\int \mathrm{d}A \, A^2 \, p(A \,|\, \omega, x) - 2\langle A \rangle \int \mathrm{d}A \, A \, p(A \,|\, \omega, x) + \langle A \rangle^2 \int \mathrm{d}A \, p(A \,|\, \omega, x) \right]$$

$$= \underbrace{\int \mathrm{d}\omega \, q_\alpha(\omega) \Big(\overline{A^2}(x, \omega) - \overline{A}(x, \omega)^2 \Big)}_{1 \,=\, \sigma^2_{\text{model}}(x)} + \underbrace{\int \mathrm{d}\omega \, q_\alpha(\omega) \big(\overline{A}(x, \omega) - \langle A \rangle \big)^2}_{2 \,=\, \sigma^2_{\text{pred}}(x)}$$

Types of uncertainties

1 Predictive uncertainty

$$\sigma^2_{\text{pred}}(x) = \int \mathrm{d}\omega \, q_\alpha(\omega) \left(\overline{A}(x, \omega) - \langle A \rangle \right)^2$$

→ following (✻) this **vanishes** for $q(\omega) \to \delta(\omega - \omega_0)$

→ with **precise** training data

→ requires more and better training → **decreases** with more training data

→ represents **statistical uncertainty** (*epistemic uncertainty*)

for physicists

$\sigma_{\text{pred}} \to \sigma_{\text{stat}}$

Types of uncertainties

2 Likelihood related uncertainty

$$\sigma^2_{\text{like}}(x) = \int \mathrm{d}\omega\, q_\alpha(\omega)\left(\overline{A^2}(x,\omega) - \overline{A}(x,\omega)^2\right) = \int \mathrm{d}\omega\, q_\alpha(\omega)\, \sigma^2_{\text{like}}(\omega, x) \equiv \langle \sigma^2_{\text{like}}(\omega) \rangle$$

network constructs uncertainty

→ already occurs without sampling → does **not vanish** for $q(\omega) \to \delta(\omega - \omega_0)$

→ However, **vanishes** if amplitude is **perfectly known** → $p(A \,|\, \omega, x) \to \delta(A - A_0)$

$$\overline{A}(x,\omega) = \int \mathrm{d}A\, A\, p(A \,|\, \omega, x) \to A_0 \qquad \overline{A^2}(x,\omega) = \int \mathrm{d}A\, A^2\, p(A \,|\, \omega, x) \to A_0^2$$

Also works if training data has no uncertainty

→ Corresponds to the **Gaussian uncertainty** in the **heteroscedastic loss**

→ reaches constant for perfect training

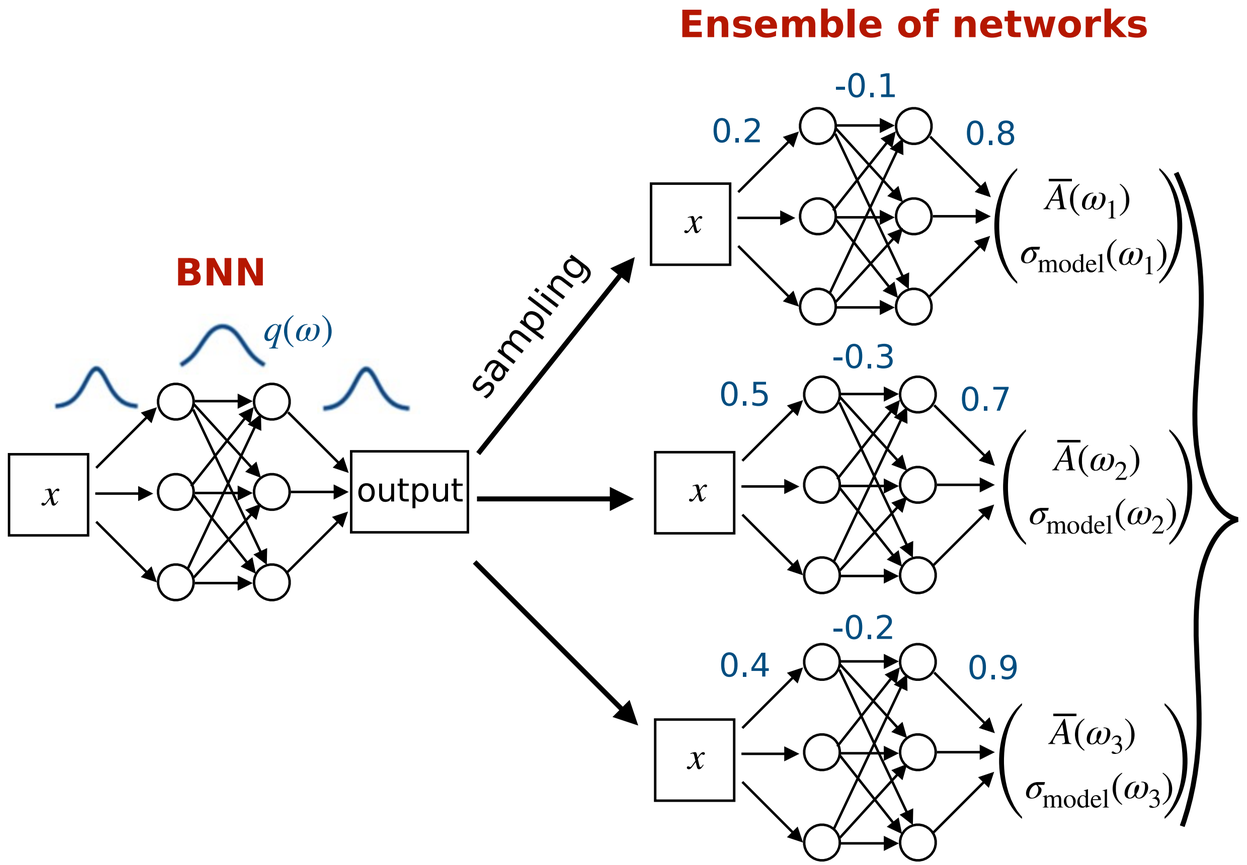
→ reflects stochastic training data, bad hyperparameters, model expressivity,..

→ represents **systematic uncertainty** (*aleatoric uncertainty*)

for physicists

$\sigma_{\text{like}} \to \sigma_{\text{syst}}$

Bayesian Regression



Output

$$\langle A \rangle = \frac{1}{N}\sum_i^N \overline{A}(\omega_i)$$

$$\sigma^2_{\text{model}} = \frac{1}{N}\sum_i^N \sigma^2_{\text{model}}(\omega_i)$$

$$\sigma^2_{\text{pred}} = \frac{1}{N}\sum_i^N (\langle A \rangle - \overline{A}(\omega_i))^2$$

$$\mathscr{L}_{\text{BNN}} = \text{KL}(q_\alpha(\omega), p(\omega)) + \int \text{d}\omega\, q_\alpha(\omega) \sum_j \left[\frac{(\overline{A}(x_j, \omega) - A_j^{\text{truth}})^2}{2\sigma_{\text{syst}}(x_j, \omega)} - \log \sigma_{\text{syst}}(x_j, \omega) \right]$$

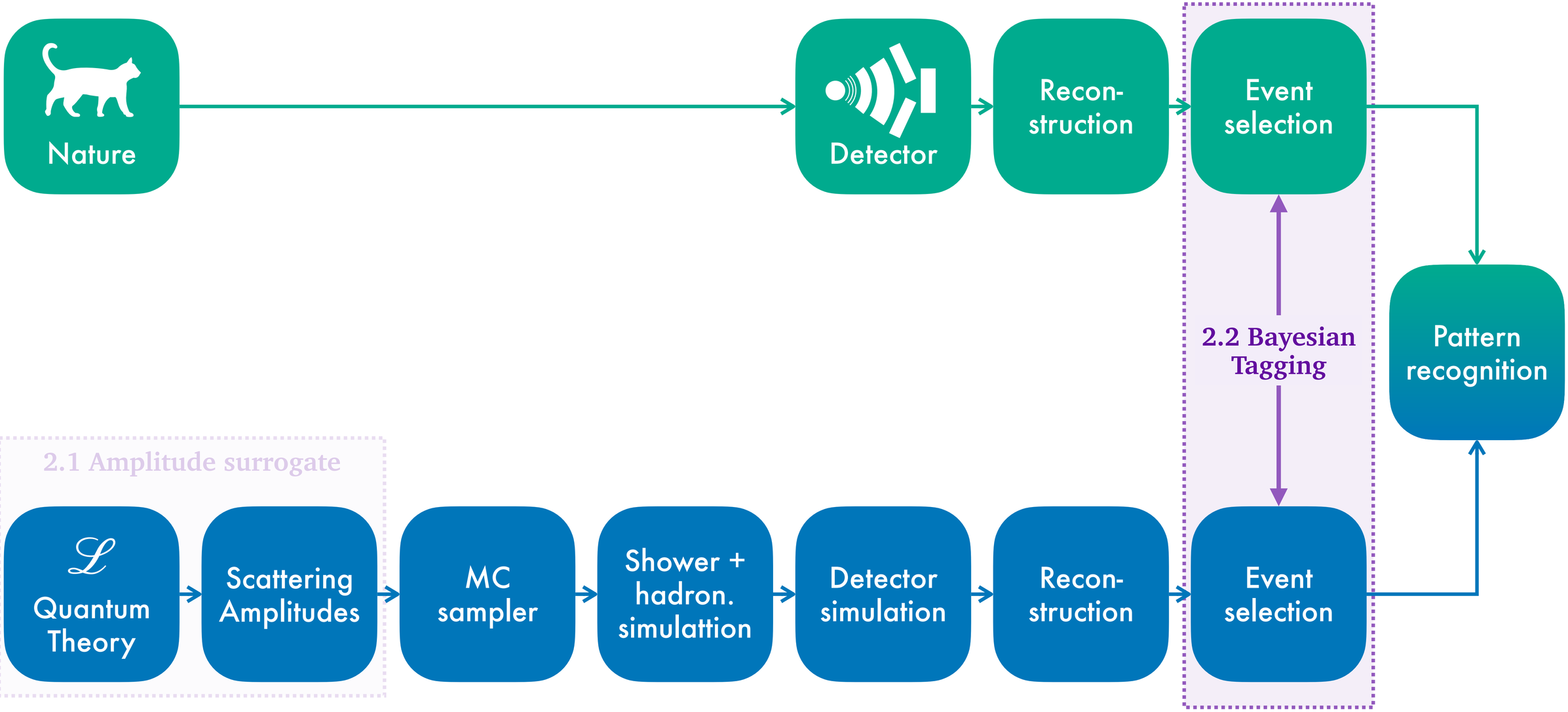
[2206.14831]

Example II

Bayesian Tagger

Bollweg, Haussmann, Kasieczka, Luchmann, Plehn, Thompson [1904.10004]

LHC analysis + ML



Bayesian Jet Tagger

Prediction of Bayesian classifier

$$\mu_{\text{pred}} \equiv p(c\,|\,x) = \int \mathrm{d}\omega\, q_\alpha(\omega)\, p(c\,|\,\omega, x)$$

BNN classifier output

Class prediction is only output

only statistic uncertainty

Statistical uncertainty

$$\sigma^2_{\text{pred}} \equiv \sigma^2_{\text{stat}} = \int \mathrm{d}\omega\, q_\alpha(\omega)\, (p(c\,|\,x, \omega) - \mu_{\text{pred}})^2$$

no additional parameter for syst. uncertainty

BCE loss

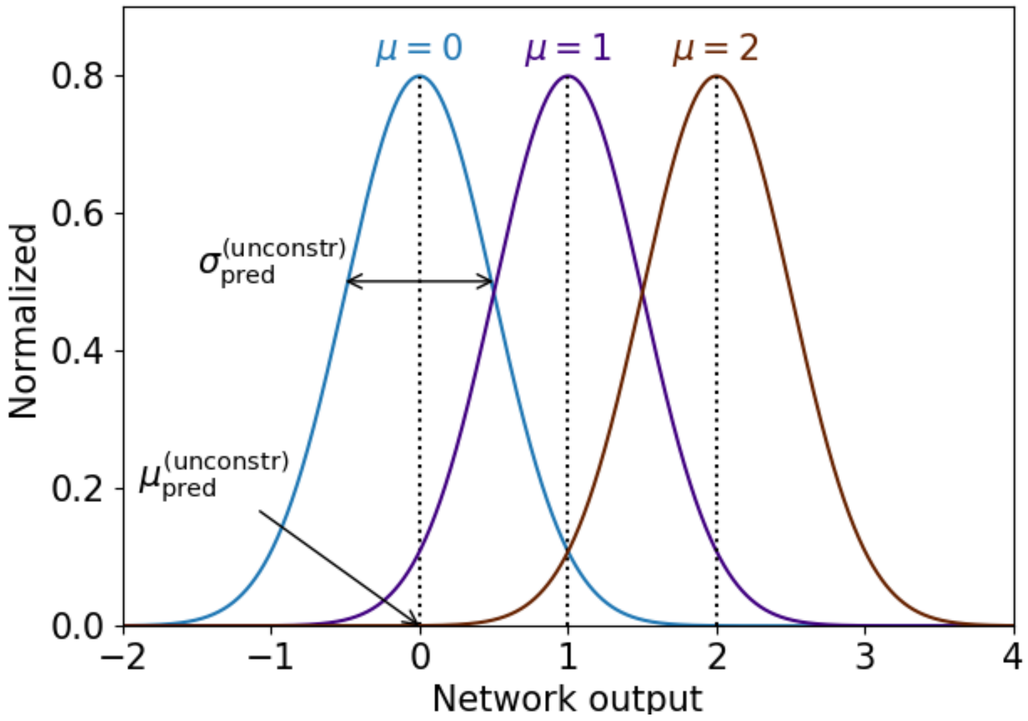
$$\mathscr{L}_{\text{BNN}} = \mathrm{KL}(q_\alpha(\omega), p(\omega)) + \int \mathrm{d}\omega\, q_\alpha(\omega) \left[\sum_j y_j^{\text{truth}} \log p(c\,|\,x_j, \omega) + (1 - y_j^{\text{truth}})\, \log(1 - p(c\,|\,x_j, \omega)) \right]$$

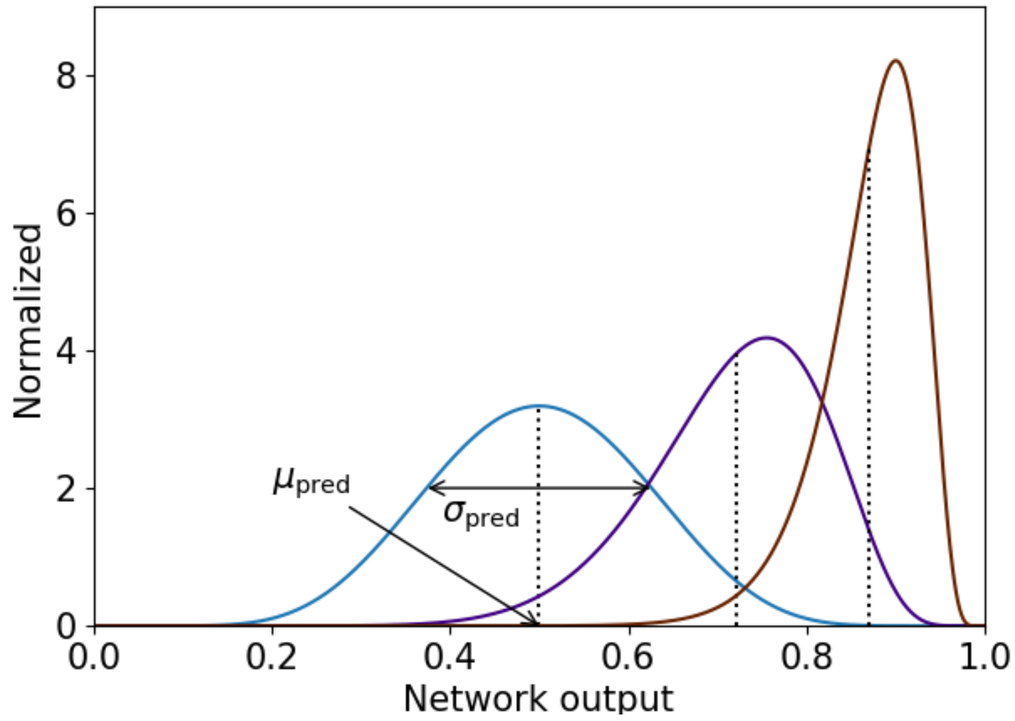
We have to be careful though…

Bayesian Jet Tagger

Classifiers have a non-linearity (sigmoid) in last layer to get $p(c \mid x) \in [0,1]$

→ **Leads to distortion of Gaussian shapes in general** ⚠️





Have to check if still Gaussian after sigmoid!!

Bayesian Jet Tagger