Friedrich-Alexander-Universität
Erlangen-Nürnberg

Improving KM3NeT Event Reconstructions and Simulations using Generative Neural Networks

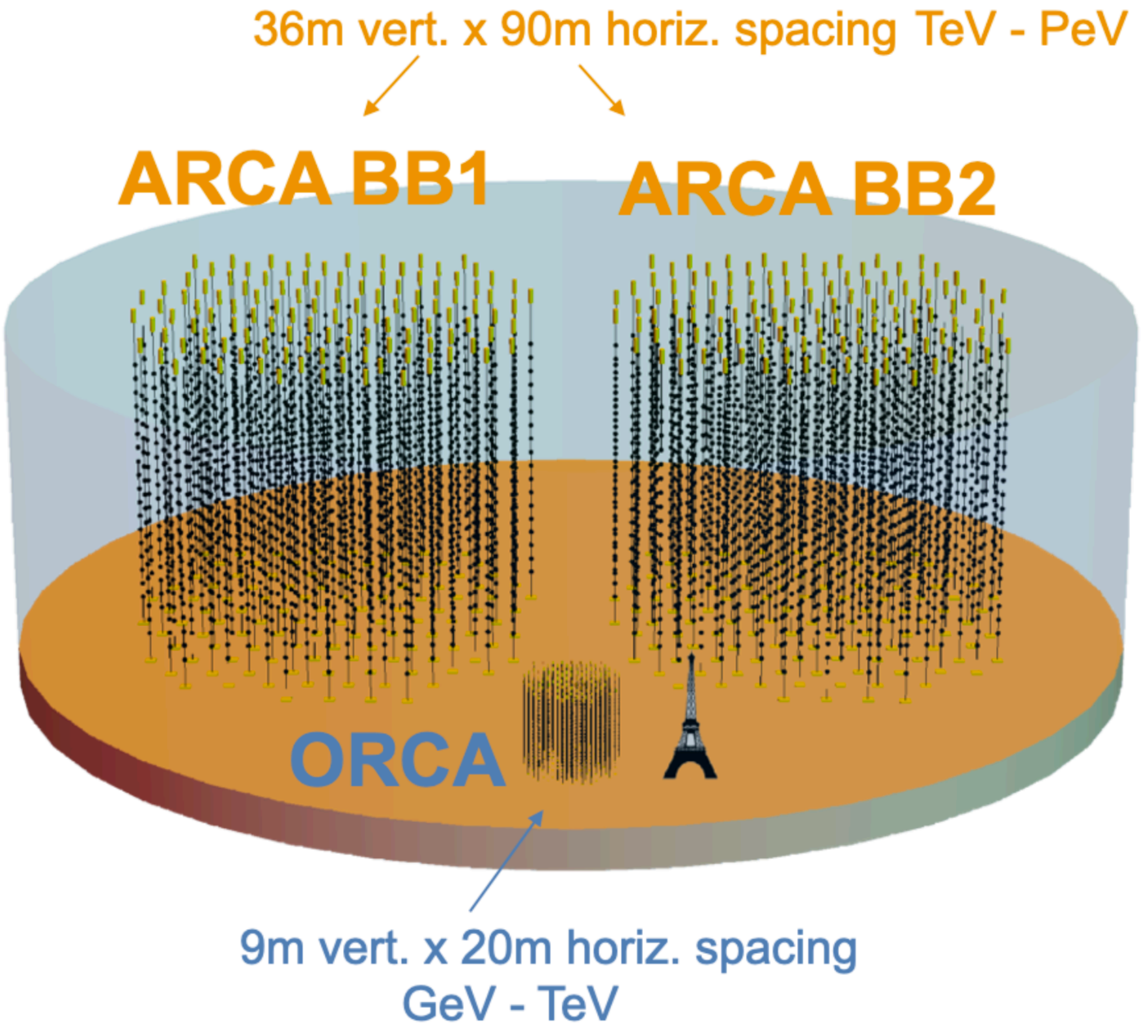
Lukas Hennig

Neutrino Physics and Machine Learning 2025

28.10.2025

The KM3NeT neutrino telescopes

- **KM3NeT is building two neutrino telescopes in the Mediterranean Sea**
  - ORCA for GeV-TeV neutrinos to resolve neutrino mass hierarchy
  - ARCA for TeV-PeV neutrinos for astronomy

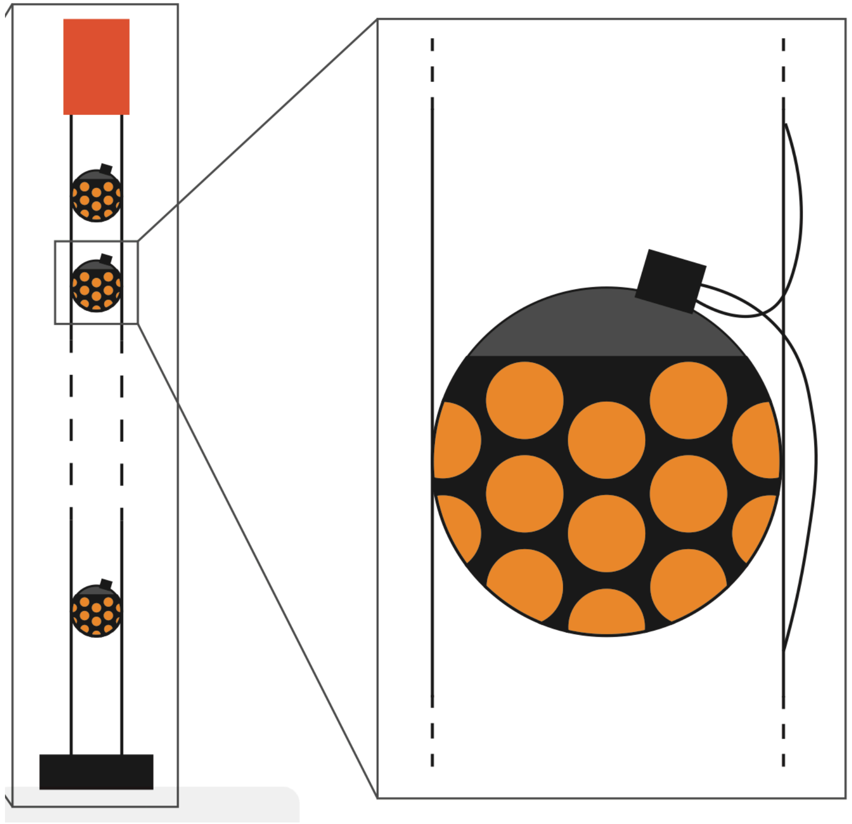


- **Detectors are already data-taking**
  - **ORCA**: 33/115 detection units; target volume $7 \cdot 10^{-3}\,\mathrm{km}^3$
  - **ARCA**: 51/230 detection units; target volume $1\,\mathrm{km}^3$

Signal detection

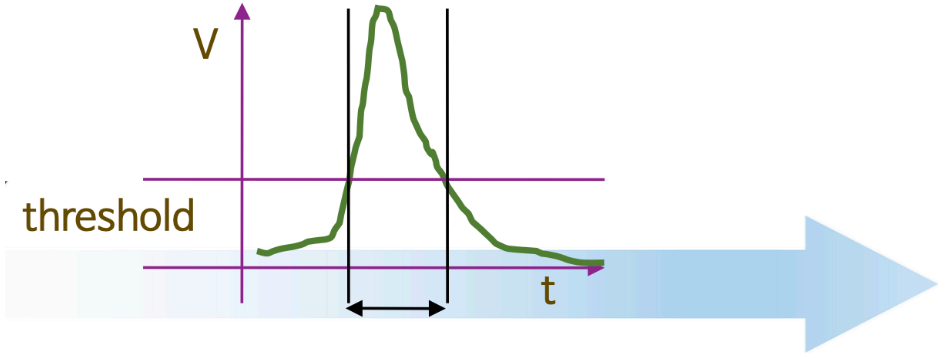
Detection unit holding 18 **digital optical modules (DOMs)**

DOM containing 31 **photomultiplier tubes** (PMTs)



- **PMT pulses surpassing a set threshold define a hit**
  - Hit defined by position, direction, and time:

$$h = (x, y, z, \theta, \phi, t)$$



Through-going muons

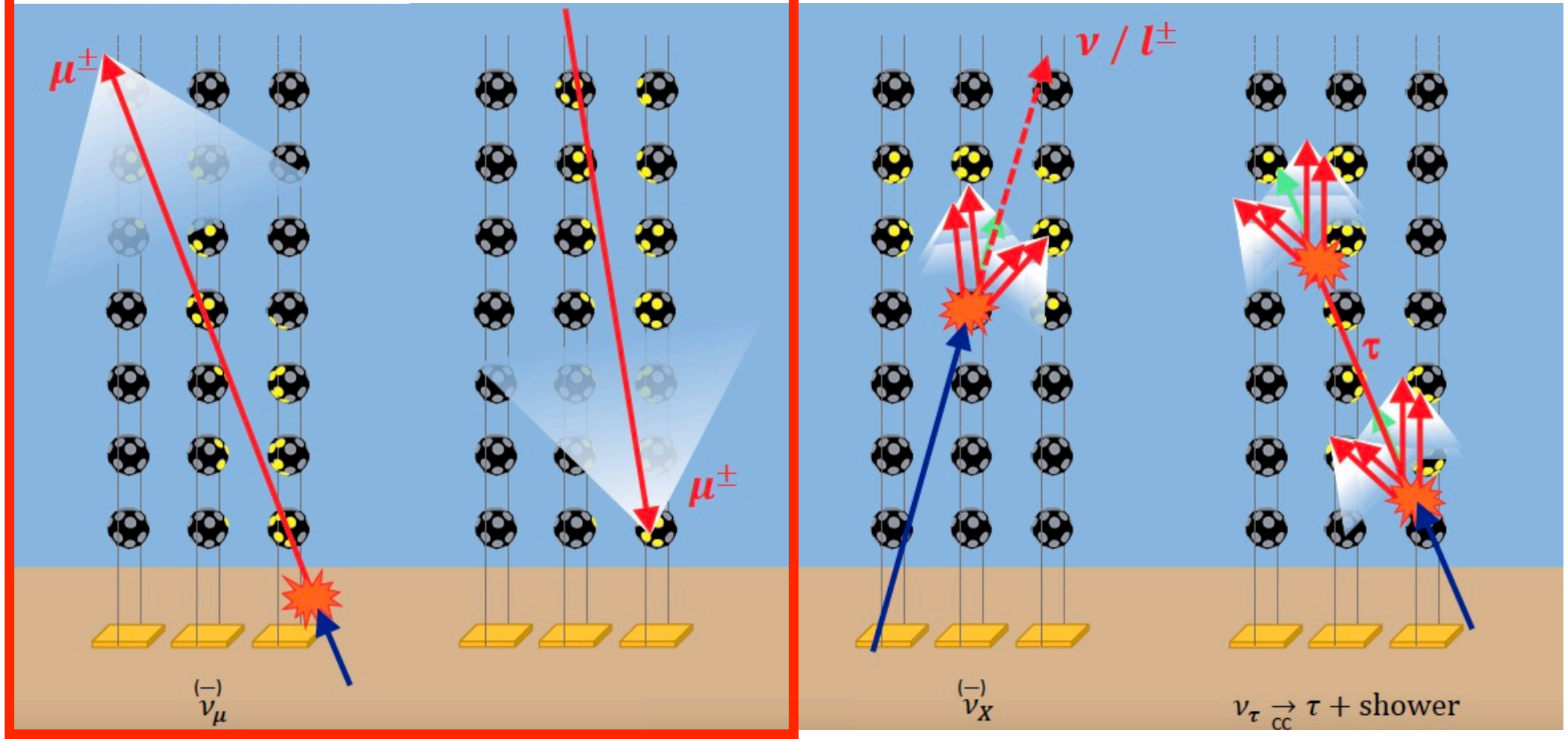
- **Events** are a set of hits that "look interesting"
  - Causal relationship between hits
- **Focus on through-going muons**
- **Aim to reconstruct:**
  - Reference point and time
  - Energy E_μ
  - Direction θ_μ, ϕ_μ

- ν_μ CC
- Track-like
- Good pointing

- Atmospheric μ
- Background for ν studies
- Signal for Cosmic Ray studies

- ν_e CC or any ν NC
- Shower-like
- Good energy resolution

- ν_τ CC
- "Double bang"

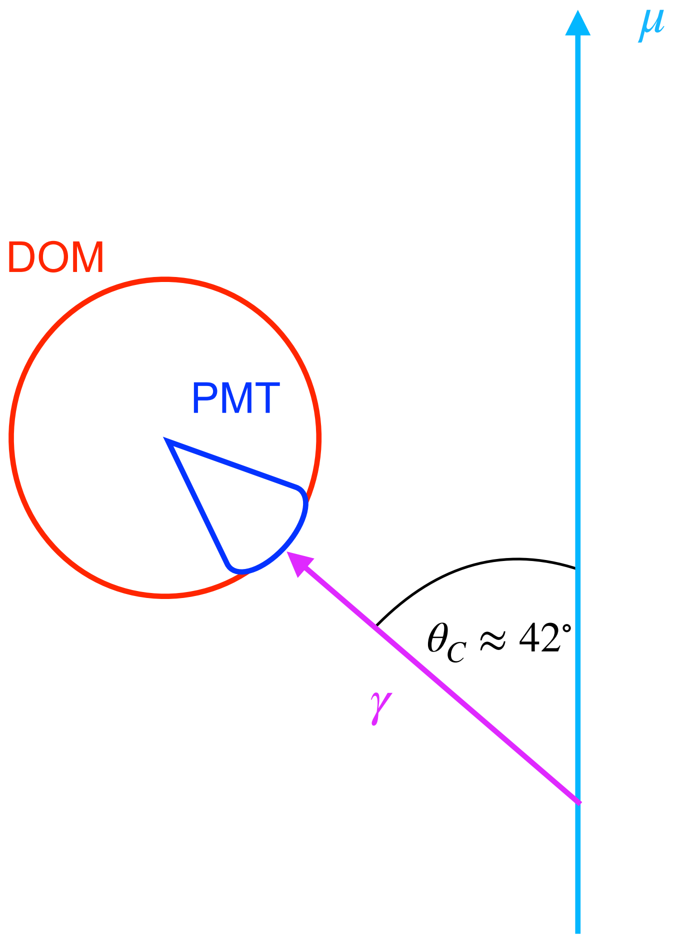


Project idea

- **Start of a new project two months ago**
  - **Goal:** Modeling hit arrival time probability density functions (PDFs)
  - **Method:** Generative Neural Networks

- **What are hit arrival time PDFs?**
  - Describe **when** we expect hits at each PMT
  - Model **how many hits** we expect

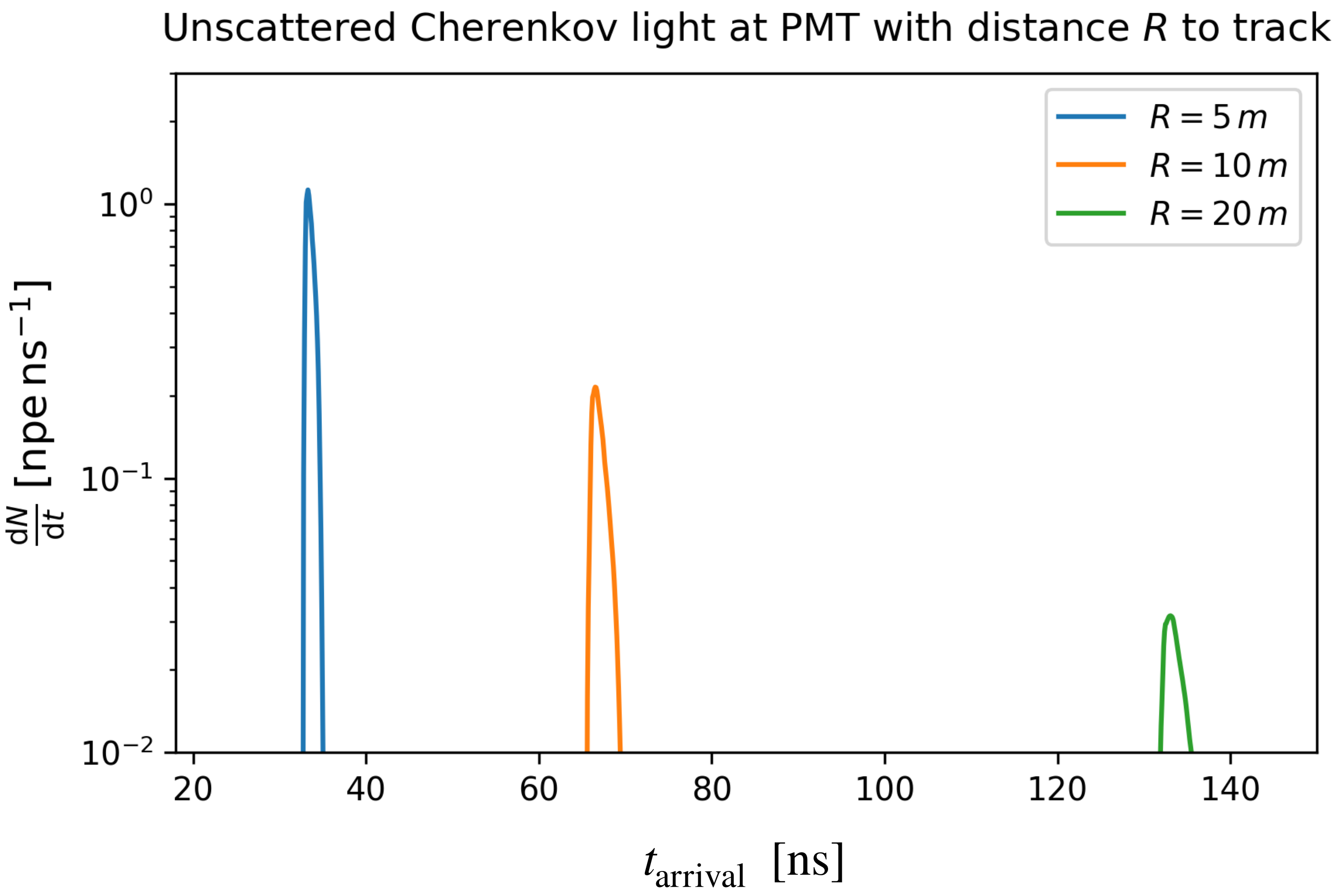
- **Arrival time PDFs are used in many applications in KM3NeT**
  - Reconstructions (likelihood computation)
  - Simulations at high energies
  - Detector calibrations (repeated maximum-likelihood fitting)



Hit arrival time distributions

- **Currently, the PDFs are stored in lookup tables**
  - Computed from semi-analytical formulas
  - Allows for fast access

- **Targets of this presentation:**
  - **Production of TeV-PeV Monte-Carlo (MC) simulations** with GPU-accelerated PhotonPropagation.jl package
  - **Cross-checks with lookup tables** for through-going muons
  - **Generative models** will enable us to use a very complex reconstruction phase space

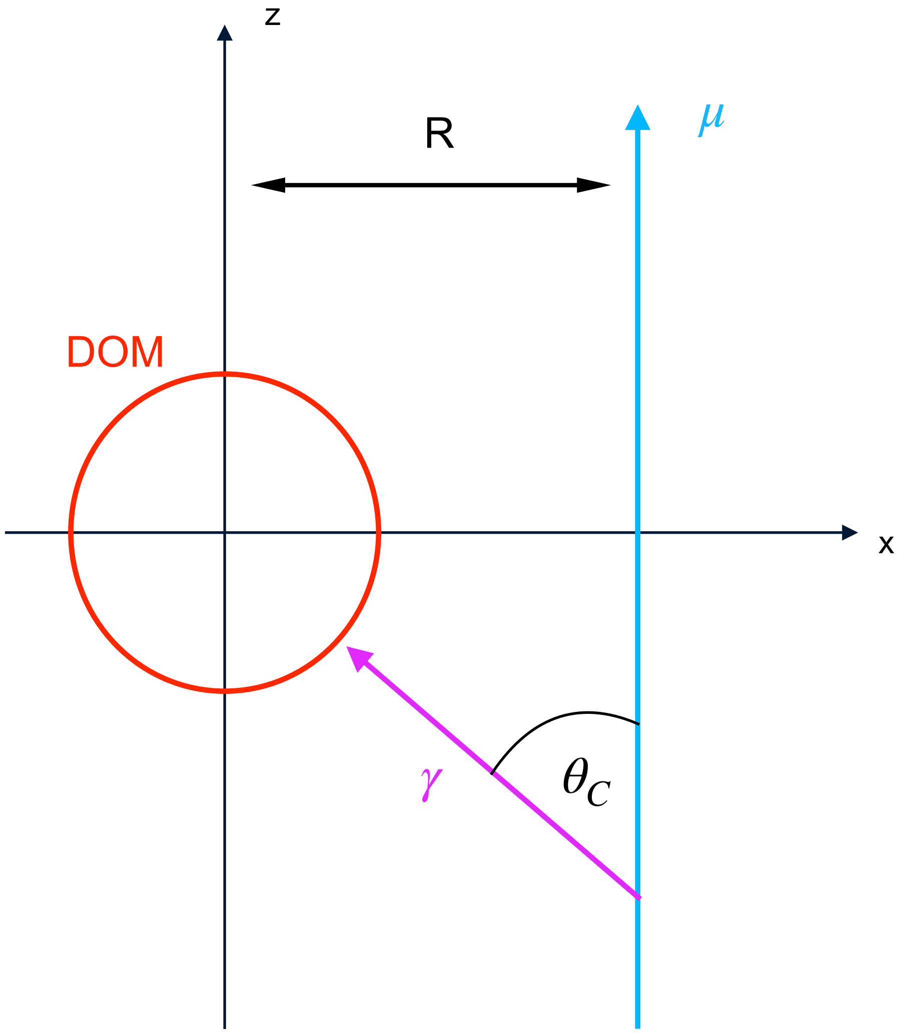


Simulation setup

- **Training of generative models requires simulating a lot of hits**
  - **Propagation of photons is slow** in the TeV-PeV energy range for KM3NeT's standard MC method
  - **Idea:** use GPU-accelerated [PhotonPropagation.jl] package

- **First simulations: upgoing muon**
  - DOM (modeled by sphere) at origin
  - Muon at distance **R = 10m** propagating in z-direction
  - Energy of muon: **10 TeV**

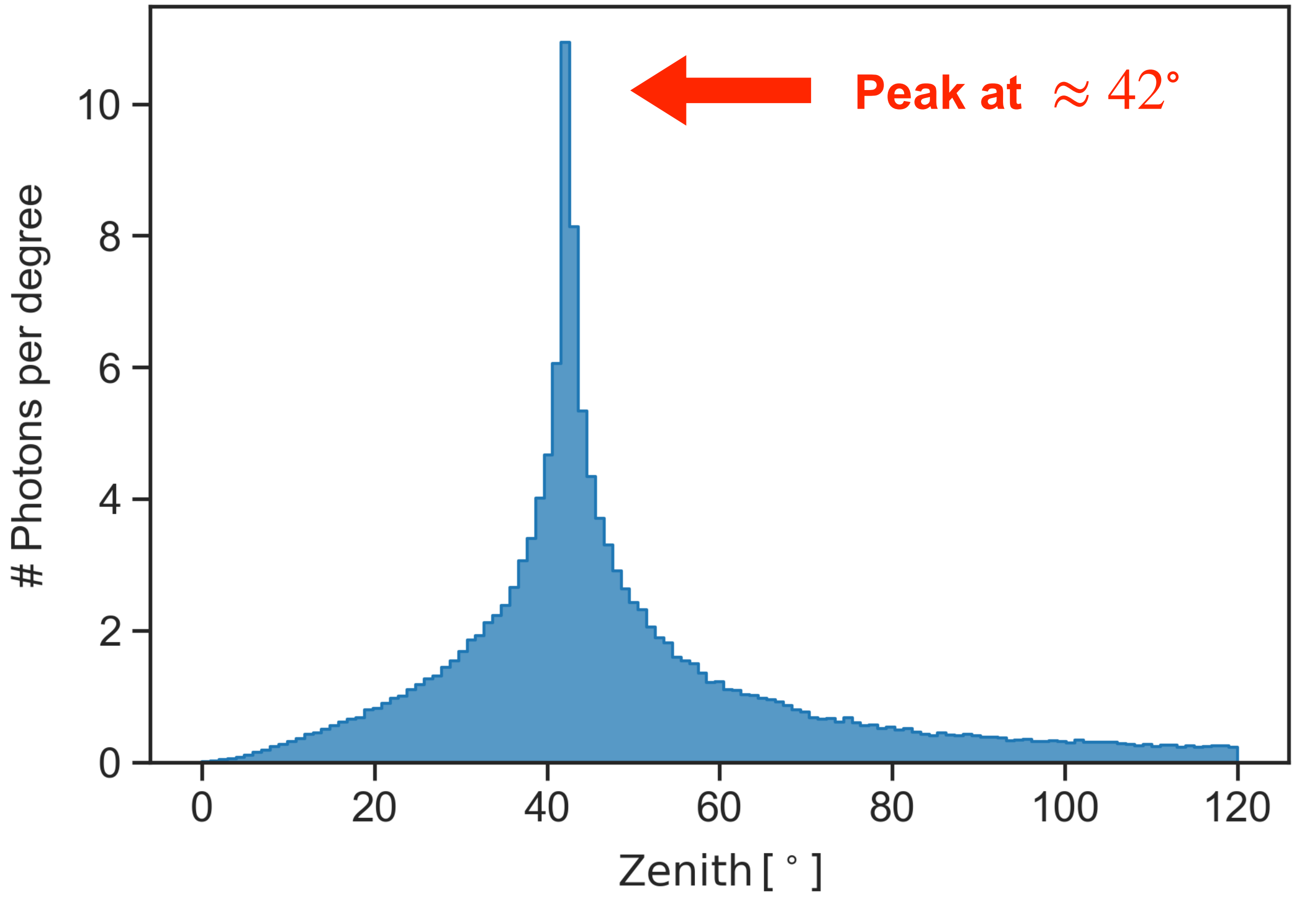
- **Expectation:** many photons will be emitted under zenith angle equal to Cherenkov angle $\theta_C \approx 42°$



Results from photon propagation

- **Simulated 1400 muons using this setup**
  - Zenith angle distribution of photons hitting the sphere has peak at Cherenkov angle ✅
- **High-energy muons emit energy by stochastic losses at discrete points along the track**
  - Photon emission from stochastic losses currently averaged over the muon track
  - Analogous to the lookup tables
- **Nominal water properties** same as in other KM3NeT simulations
- **Hits** are computed by applying **photon acceptance probability**

Expected number of photons hitting the sphere per muon

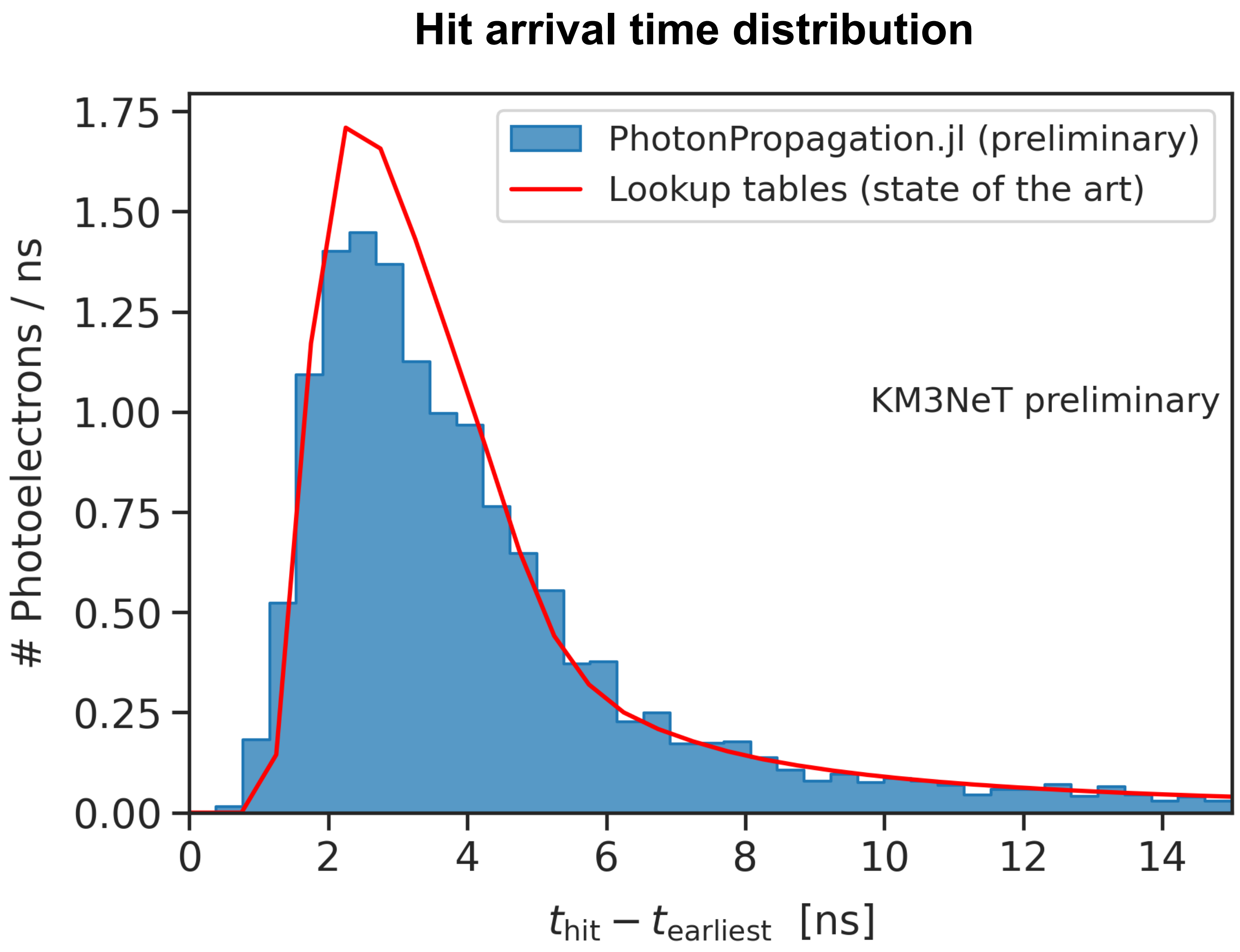


Hit arrival time distribution

- **Reference time chosen as earliest possible arrival of Cherenkov photons**

- **First PhotonPropagation simulations predict less hits than lookup tables**
  - **PhotonPropagation:** $\approx 5.8\,\mathrm{PE}$
  - **Lookup tables:** $\approx 6.3\,\mathrm{PE}$
  - **Ratio:** $\approx 92\,\%$

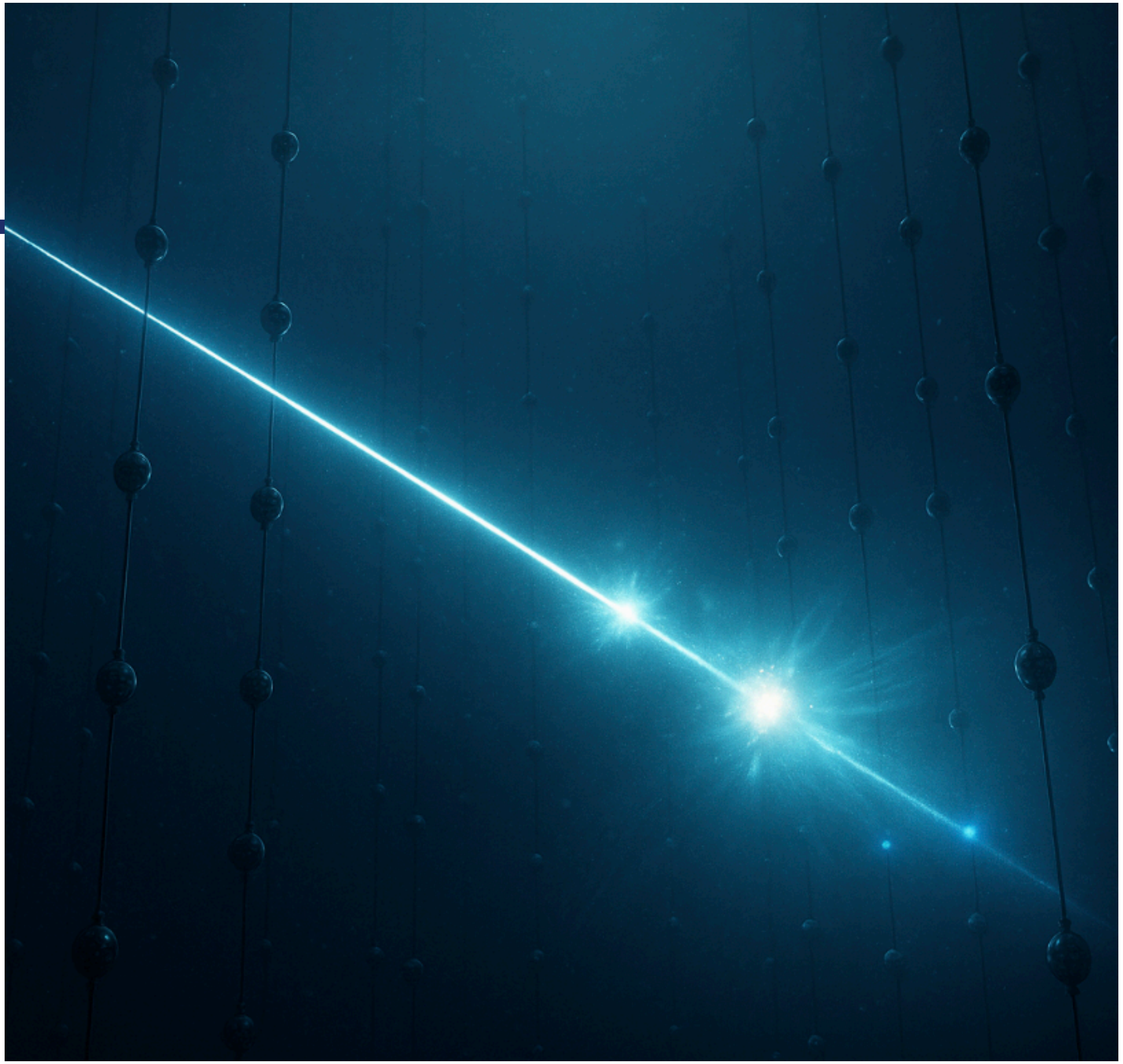
- **The lookup tables are well cross-checked with other simulation tools**
  - Comparisons of the used models ongoing
  - Most likely mismodeling in my code



Generative models

- **Generative models are machine-learning algorithms**
  - They interpolate the underlying **probability density function** during training
  - Can be used to **generate new samples**

- **Their strength:** ability to infer and encode very complex PDFs into compact model
  - No performance improvement over lookup tables expected in current reconstruction scenarios!
  - **Expectation: Models will allow us to fit very complex event hypotheses very fast!**
  - Suitable for GPU-accelerated reconstruction



ChatGPT's impression of the ultra-high energy event detected by KM3NeT

Observation of an ultra-high-energy cosmic neutrino with KM3NeT. [Nature 638, 376–382 (2025)]

Normalizing Flows

- **Tool of choice: Normalizing Flows (NFs)**
  - Bijective function f learning a coordinate transformation between random variables $x \leftrightarrow t_{\mathrm{Hit}}$
  - Base space x is distributed Gaussian, t_{Hit} is the target variable

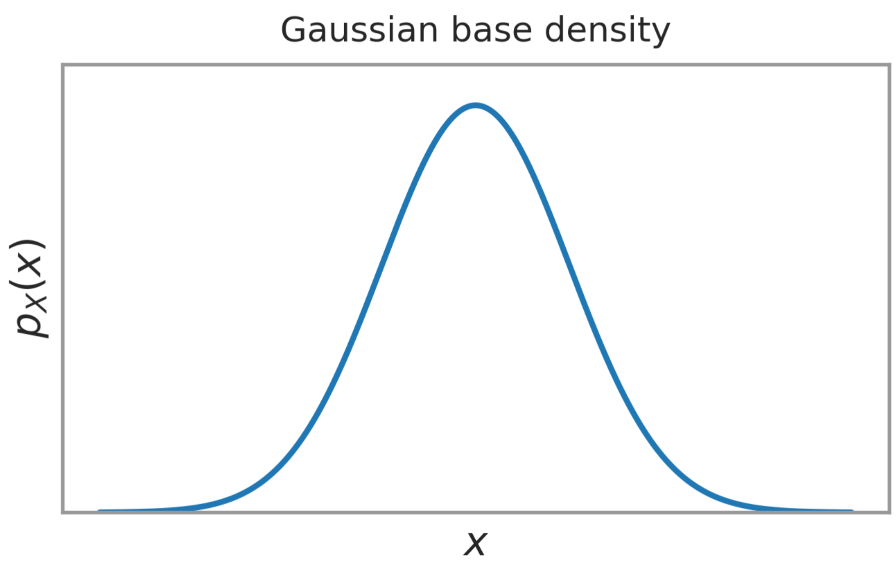
- Normalizing flow f is implemented by an invertible, differentiable neural network

Generating new hits:

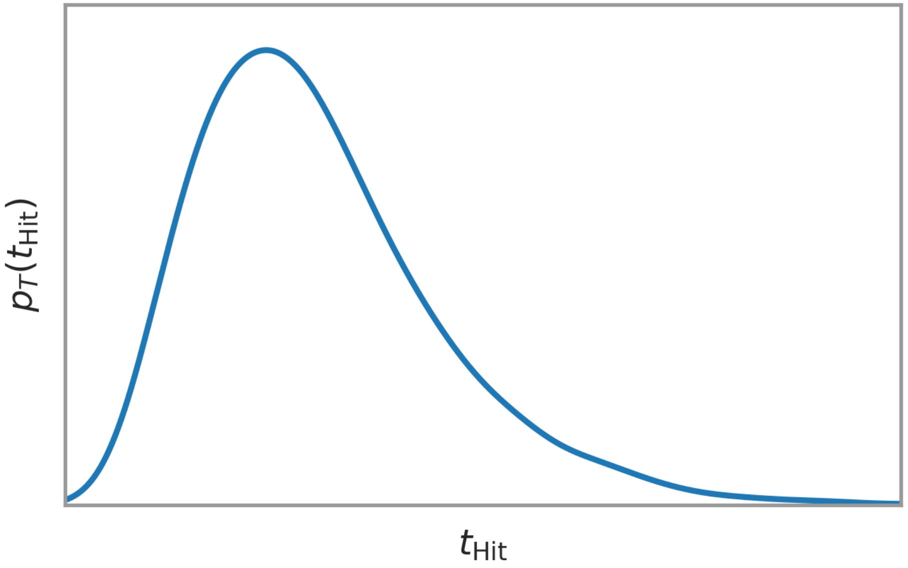
$$\hat{x} \sim p_X(x)$$
$$\hat{t}_{\mathrm{Hit}} = f(\hat{x})$$

Computing likelihoods:

$$\hat{x} = f^{-1}(\hat{t}_{\mathrm{Hit}})$$
$$p_T(\hat{t}_{\mathrm{Hit}}) = p_X(\hat{x})\,|\det J_f(\hat{x})\,|^{-1}$$

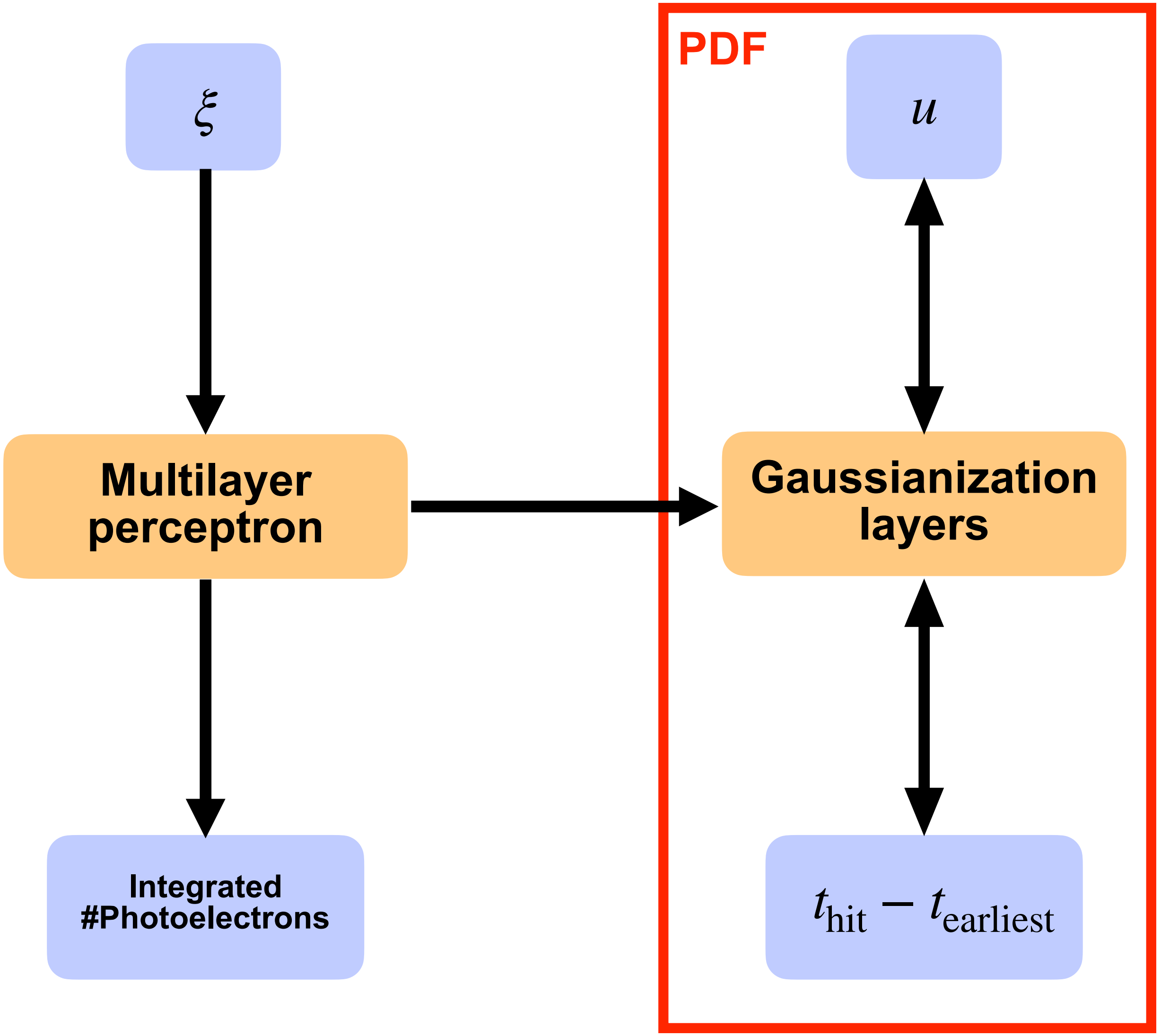


$t_{\mathrm{Hit}} = f(x)$ $x = f^{-1}(t_{\mathrm{Hit}})$



Model architecture overview

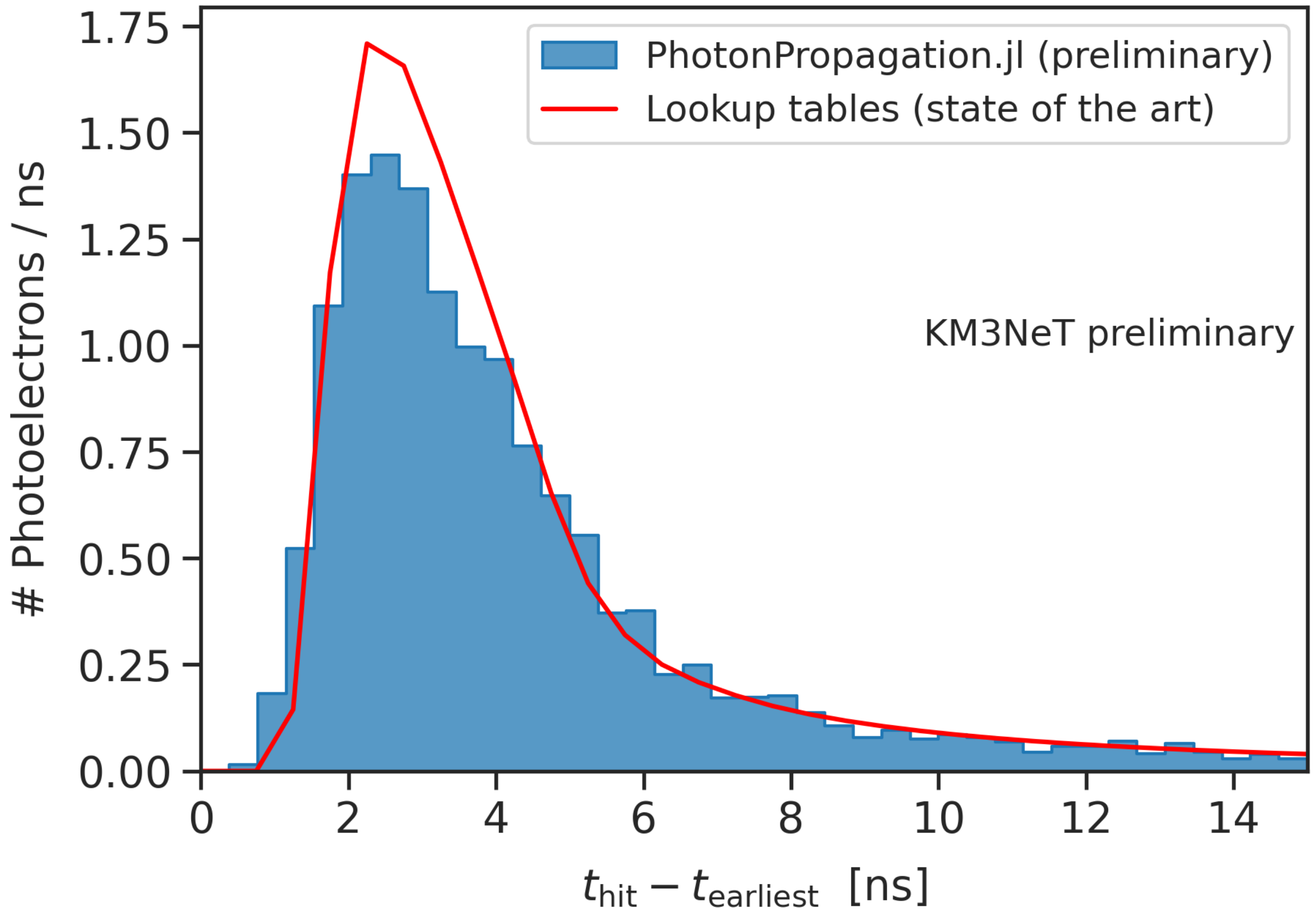
- **Arrival time PDFs conditioned on event hypothesis** $\xi = (E_\mu, \vec{r}_0, t_0, \theta_\mu, \phi_\mu)$
- **Conditional normalizing flows** from [jammy_flows] package are suitable
- **Invertible Gaussianization layers** connect base space and target space
- **Multilayer perceptron** encodes the condition on ξ
  - **Maps ξ to parameters** of the Gaussianization layers
  - **Predicts the normalization** of the probability density function (integrated number of photoelectrons)
- **First trainings will begin in the next weeks!**



Conclusion

- **GPU-accelerated Monte-Carlo simulations for high-energy muons with [PhotonPropagation.jl]**
- **Normalizing flows** will be used to model the PDFs
- **Project will follow an iterative approach:**
  - **Generate Monte-Carlo simulations**
  - **Cross-check** the distributions with lookup tables
  - Train a **normalizing flow**
  - **Add new fit parameters** and repeat
- **Stay tuned for first results!**

Hit arrival time distribution



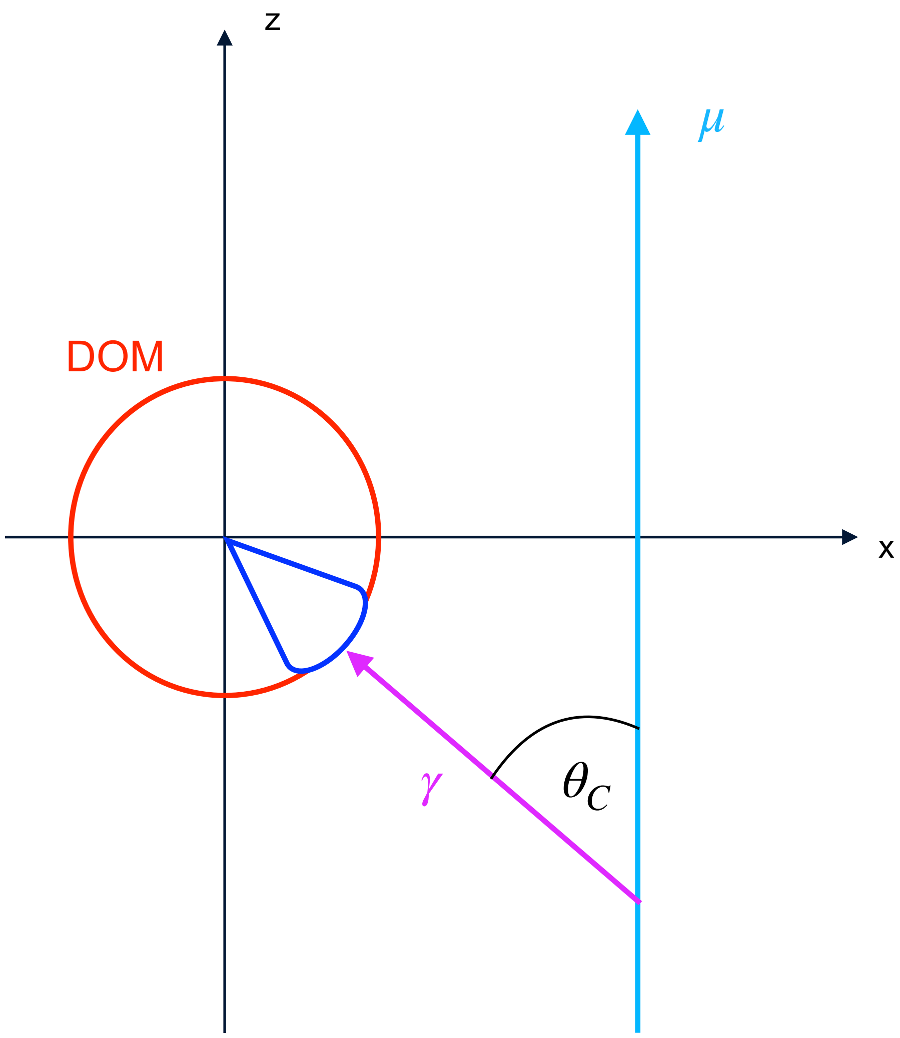
Backup

Photon acceptance probability

- **Compute time distribution of accepted photons**
  - **Place a PMT** directly facing the Cherenkov photons
  - Compute **effective area** of PMT for each photon
  - Divide by area of **DOM cross section**

- **Result:** photon acceptance probability

- Effective PMT area computed using **angular acceptance** and **quantum efficiency**



GPU photon propagation code

- https://github.com/PLEnuM-group/PhotonPropagation.jl
- Forward ray-tracing of individual photons
- Pure julia implementation, CUDA accelerated photon propagation
- Customizable medium properties (absorption length, scattering length, scattering function, refractive index, dispersion), however only completely homogenous media supported
- Customizable emitters / receivers
- Uses IceCube parametrizations for Cherenkov light yields

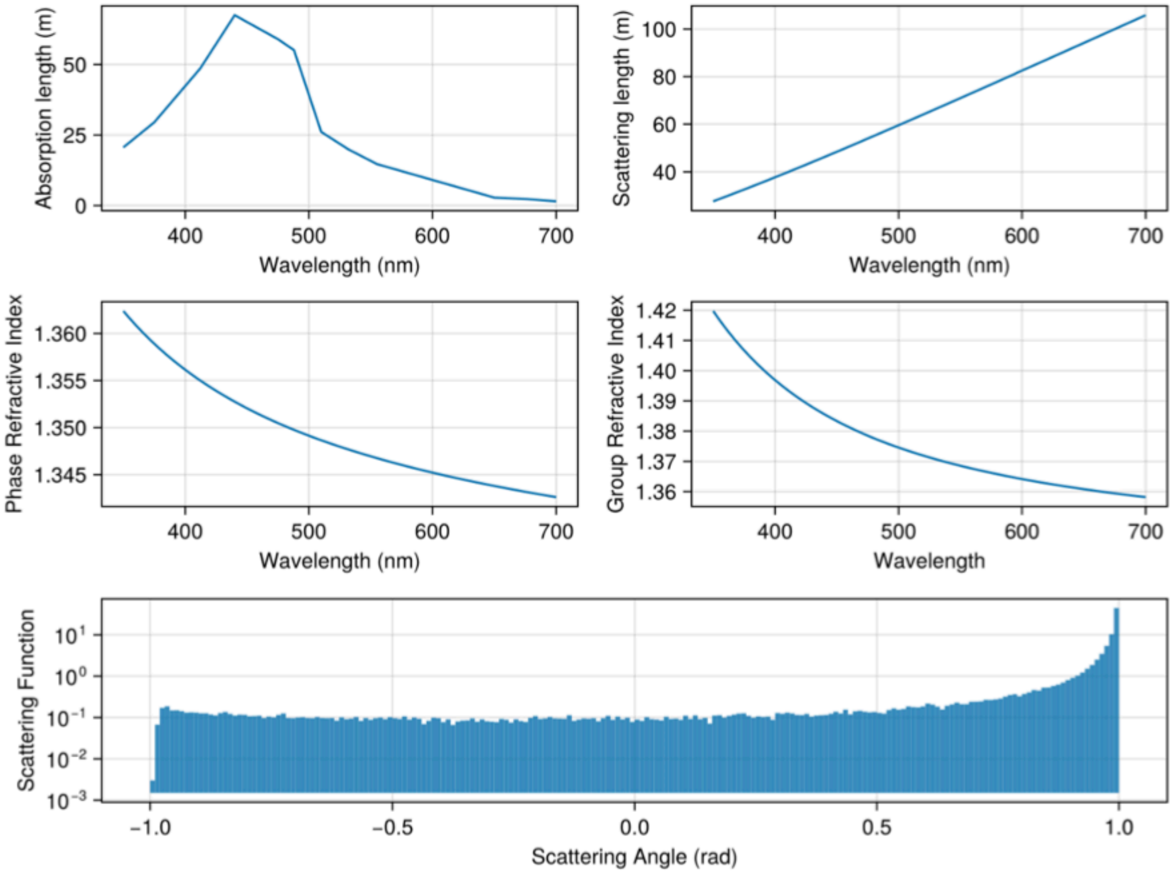
Slide by Christian Haack

ARCA medium properties

Base package (public, will be in Julia general registry soon):
https://github.com/JuliaHEP/CherenkovMediumBase.jl

Implementes properties defined in the Simulation Description Document:

- Quan & Fry Dispersion model
- Kopelevich scattering model
- Mixed Henyey-Greenstein & Einstein-Smoluchowsky (pure water) scattering function



Slide by Christian Haack

Algorithm

1. Sample photon properties (position, direction, wavelength)
2. Repeat for N steps: Sample step length (distance to next scattering site)
   I. Check for intersection with detectors
      - Intersection: Save impact point and go back to 1)
      - No Intersection: Continue with II)

   II. Step to scattering site. Sample new direction from scattering function. Continue with 2)

Computation for each photon is independent -> Great potential for parallelization on GPUs

Slide by Christian Haack

Normalizing flows

$$\mathbf{x} = T(\mathbf{u}) \quad \text{where} \quad \mathbf{u} \sim p_{\mathrm{u}}(\mathbf{u})$$

$$p_{\mathrm{x}}(\mathbf{x}) = p_{\mathrm{u}}(\mathbf{u})\ |\det J_T(\mathbf{u})|^{-1} \quad \text{where} \quad \mathbf{u} = T^{-1}(\mathbf{x})$$

$$J_T(\mathbf{u}) = \begin{bmatrix} \frac{\partial T_1}{\partial \mathrm{u}_1} & \cdots & \frac{\partial T_1}{\partial \mathrm{u}_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial T_D}{\partial \mathrm{u}_1} & \cdots & \frac{\partial T_D}{\partial \mathrm{u}_D} \end{bmatrix}$$

- **Training is performed by minimizing the Kullback-Leibler divergence between the target distribution $p_X^*(x)$ and the normalizing flow $p_X(x;\ \theta)$**

$$(T_2 \circ T_1)^{-1} = T_1^{-1} \circ T_2^{-1}$$
$$\det J_{T_2 \circ T_1}(\mathbf{u}) = \det J_{T_2}(T_1(\mathbf{u})) \cdot \det J_{T_1}(\mathbf{u})$$

$$\mathcal{L}(\boldsymbol{\theta}) = D_{\mathrm{KL}}\left[p_{\mathrm{x}}^*(\mathbf{x}) \,\|\, p_{\mathrm{x}}(\mathbf{x};\boldsymbol{\theta})\right]$$

$$\approx -\frac{1}{N}\sum_{n=1}^{N} \log p_{\mathrm{u}}(T^{-1}(\mathbf{x}_n;\boldsymbol{\phi});\boldsymbol{\psi}) + \log|\det J_{T^{-1}}(\mathbf{x}_n;\boldsymbol{\phi})|$$

Log-likelihood in base-space **Volume correction**

Journal of Machine Learning Research 22 (2021) 1-64