Advancing Detector Calibration and Event Reconstruction in Water Cherenkov Detectors through Differentiable Simulation

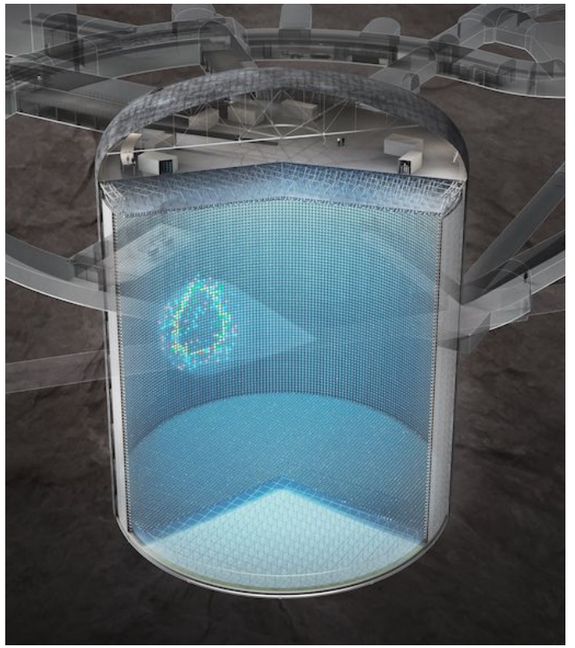
Omar Alterkait

Tufts University / IAIFI

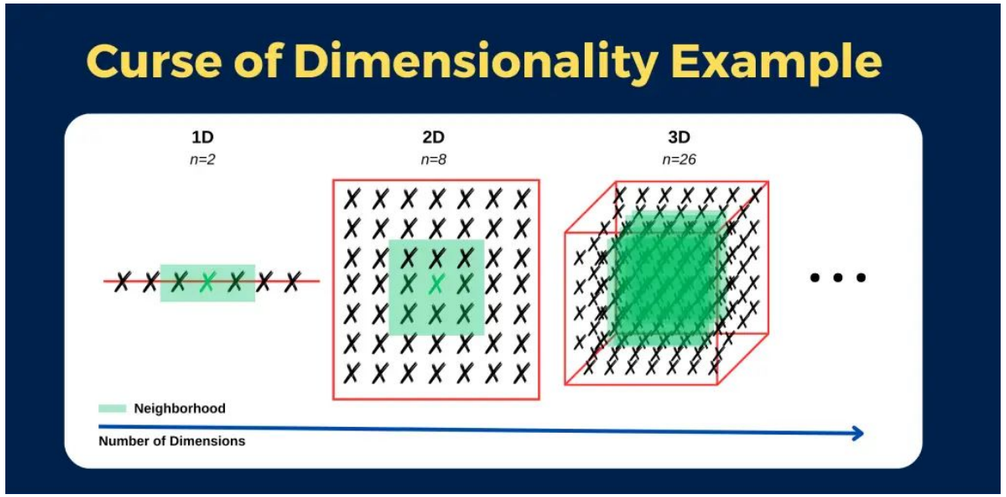
on behalf of CIDeR-ML collaboration (HEP US-Japan)

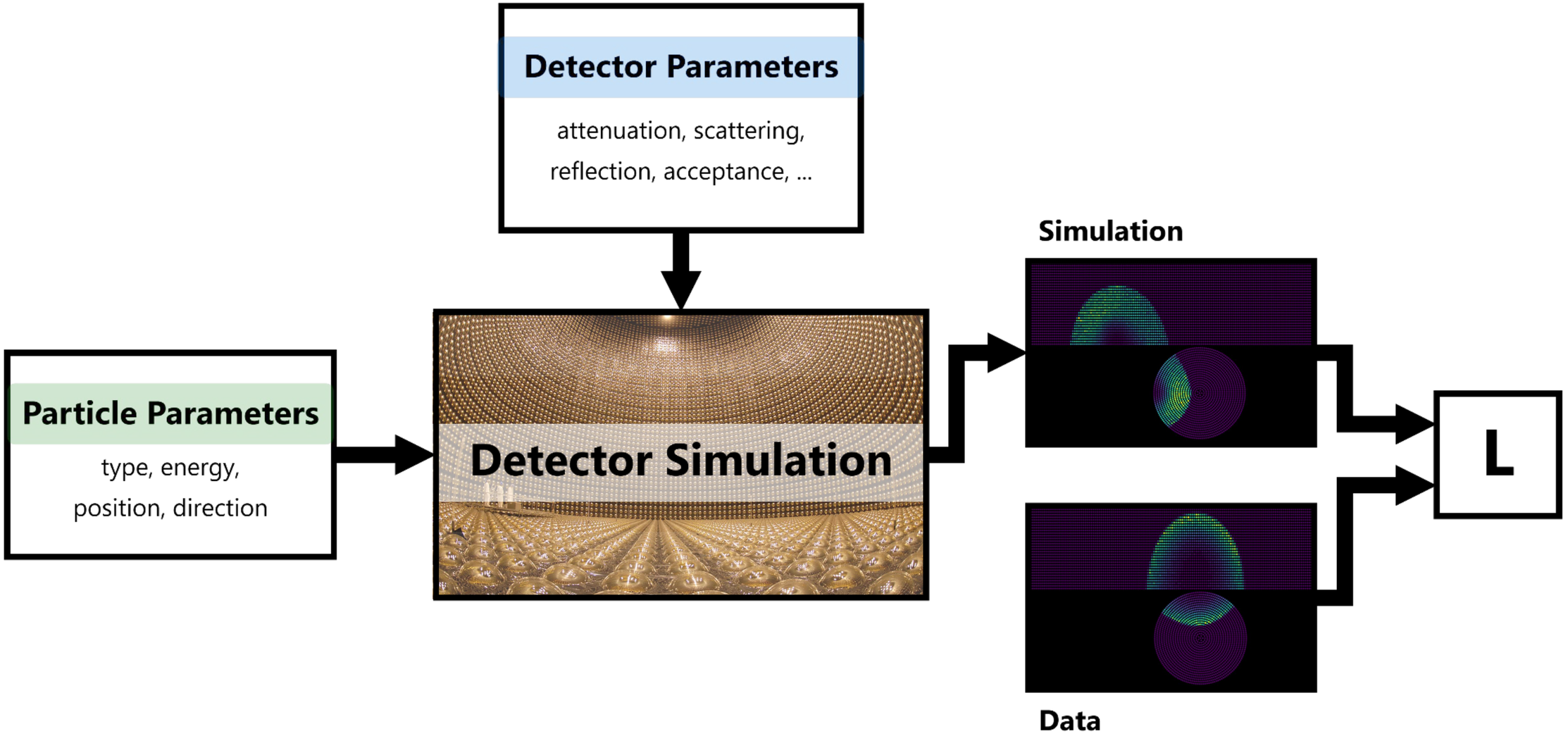
The Calibration Challenge: A Limiting Systematic

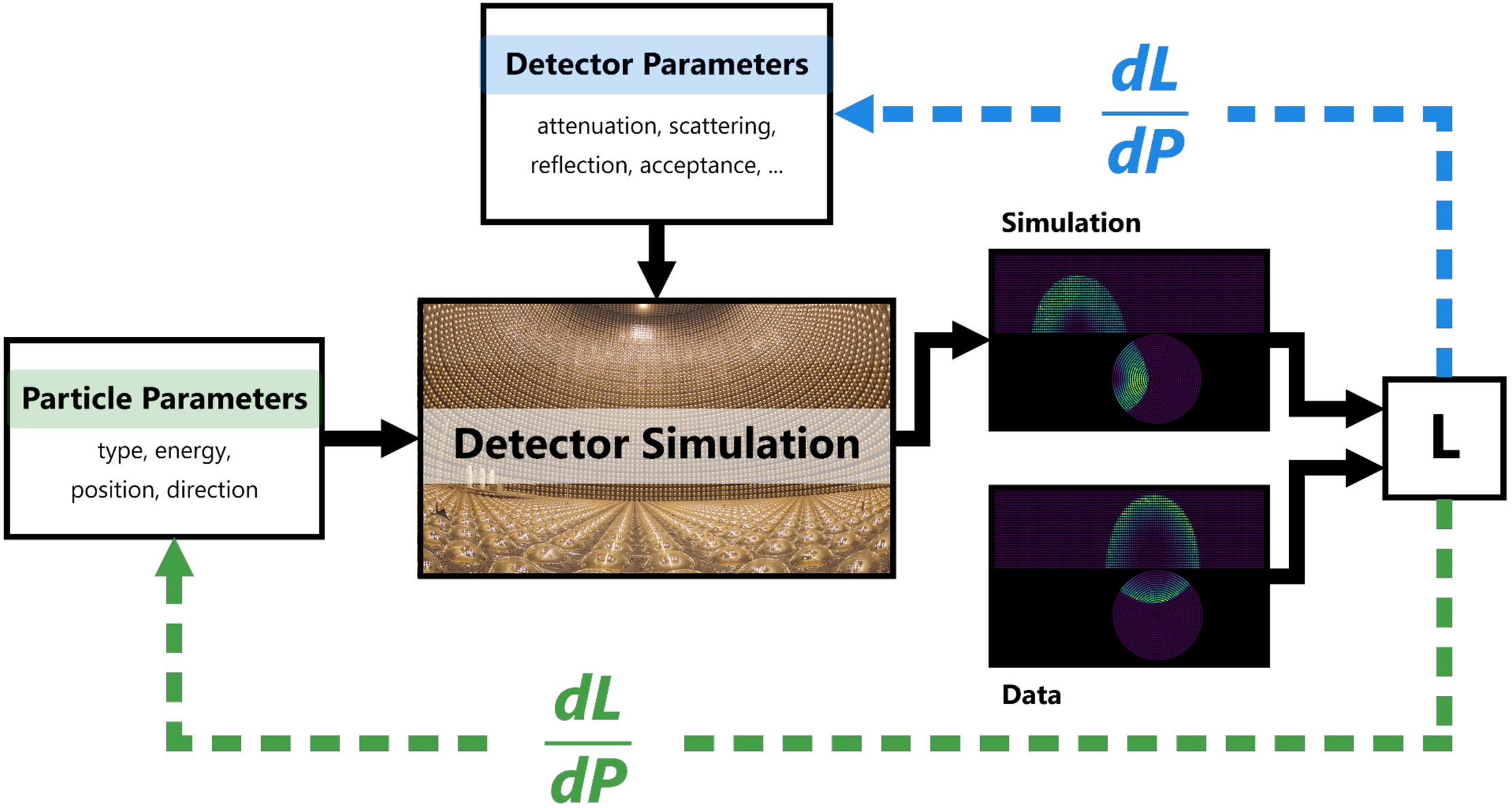
- CP violation discovery potential directly limited by calibration precision
- Detector response governed by complex interplay of multiple parameter types
- Traditional approaches face "curse of dimensionality":
  - Sequential parameter optimization does not explore critical interdependencies
  - Conventional optimization methods pronte to converge to suboptimal solutions
  - Need a fast and scalable framework that can optimize all detector parameters concurrently



Hyper Kamiokande




Detector Parameters
attenuation, scattering,
reflection, acceptance, ...
Particle Parameters
type, energy,
position, direction
Detector Simulation
Simulation
Data
L

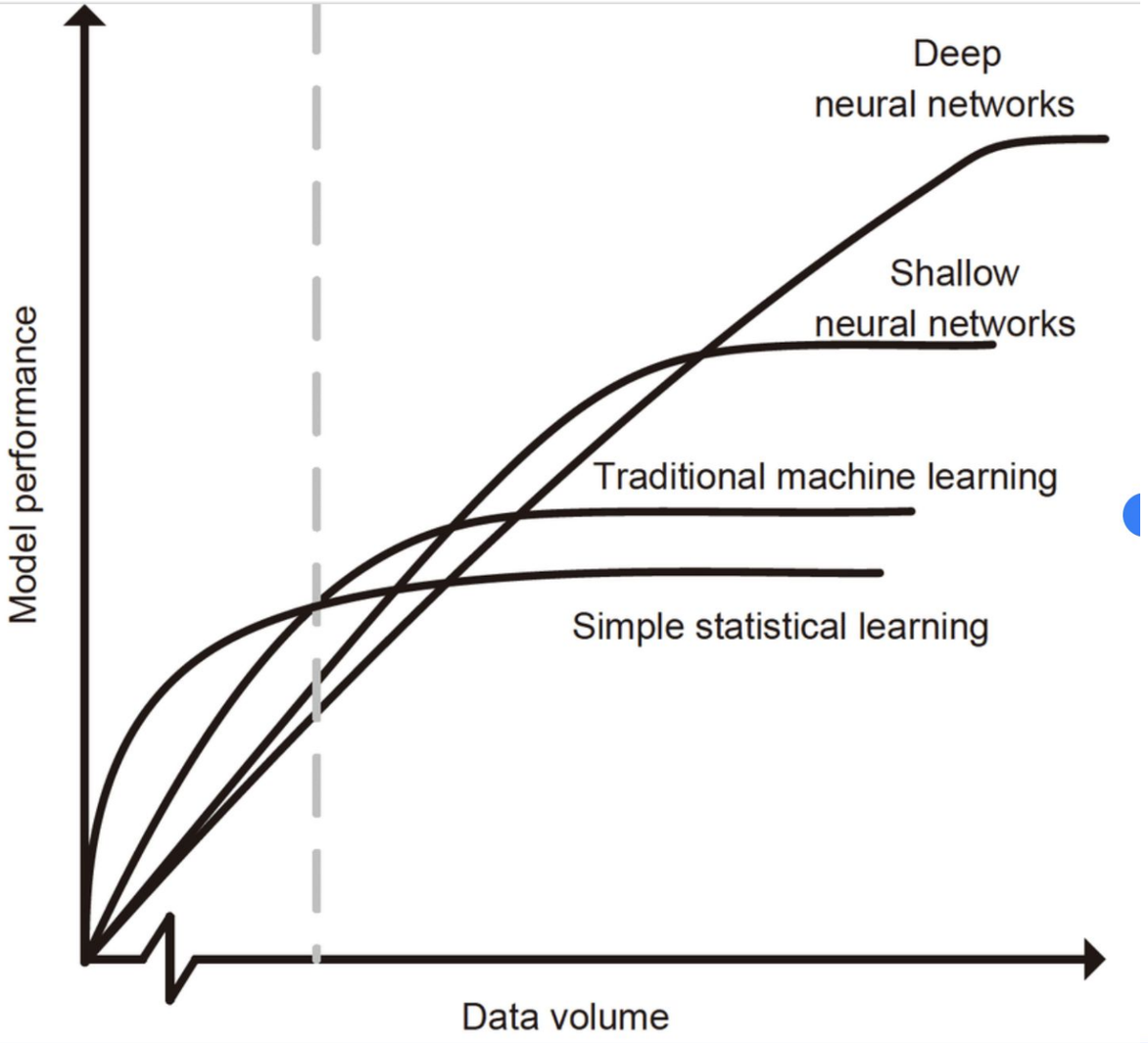

Detector Parameters
attenuation, scattering,
reflection, acceptance, ...
dL
dP
Simulation
Particle Parameters
type, energy,
position, direction
Detector Simulation
L
Data
dL
dP

Neural Networks as Function Approximators

We can try to learn this detector simulation using a neural network surrogate (next talk)

Neural networks offer:

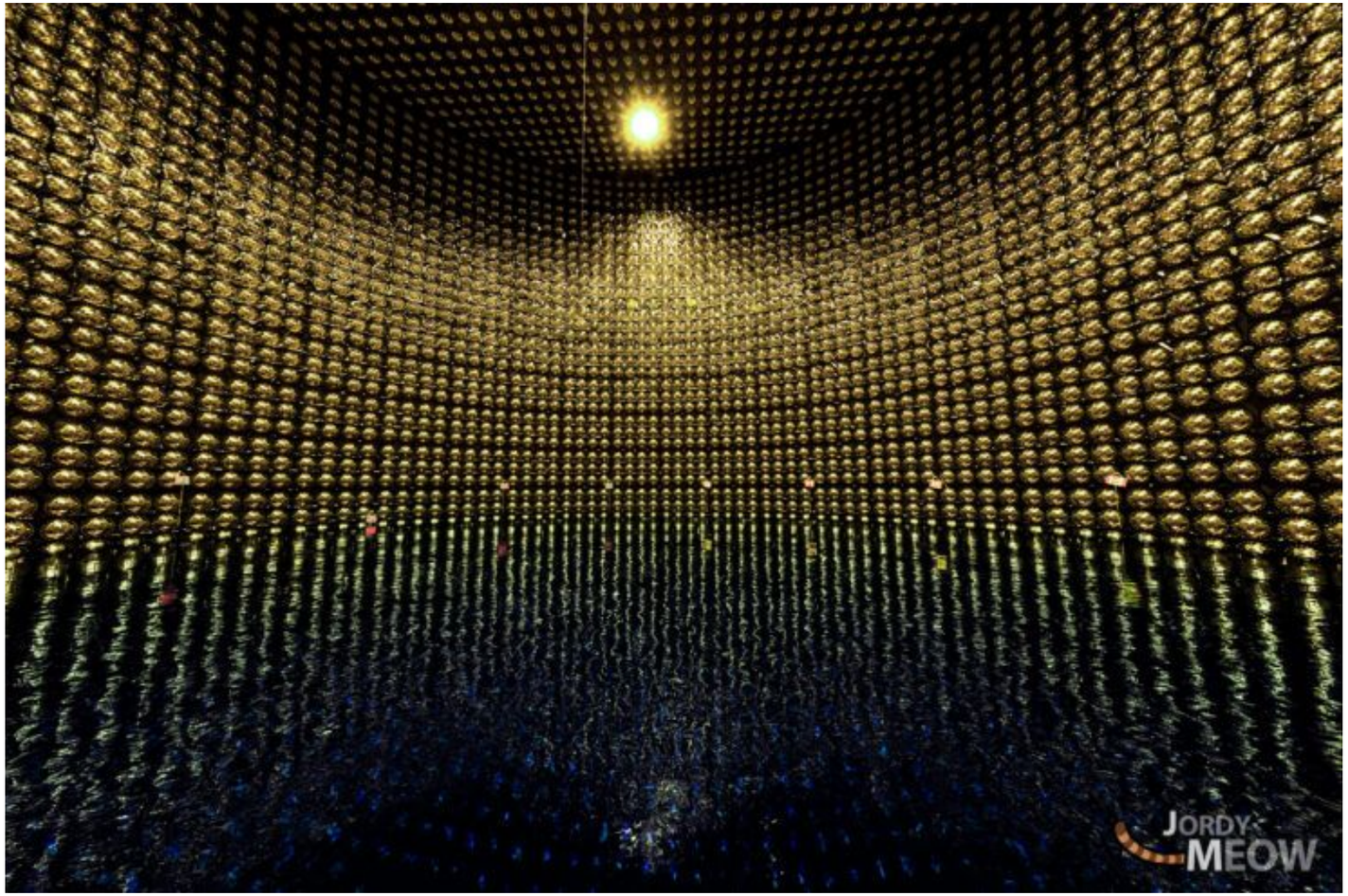
- Freedom from restrictive model assumptions
- Ability to learn complex patterns
- Automatic feature extraction
- Handling of high-dimensional problems



Breaking Free from Assumptions

Real detector responses are complex:

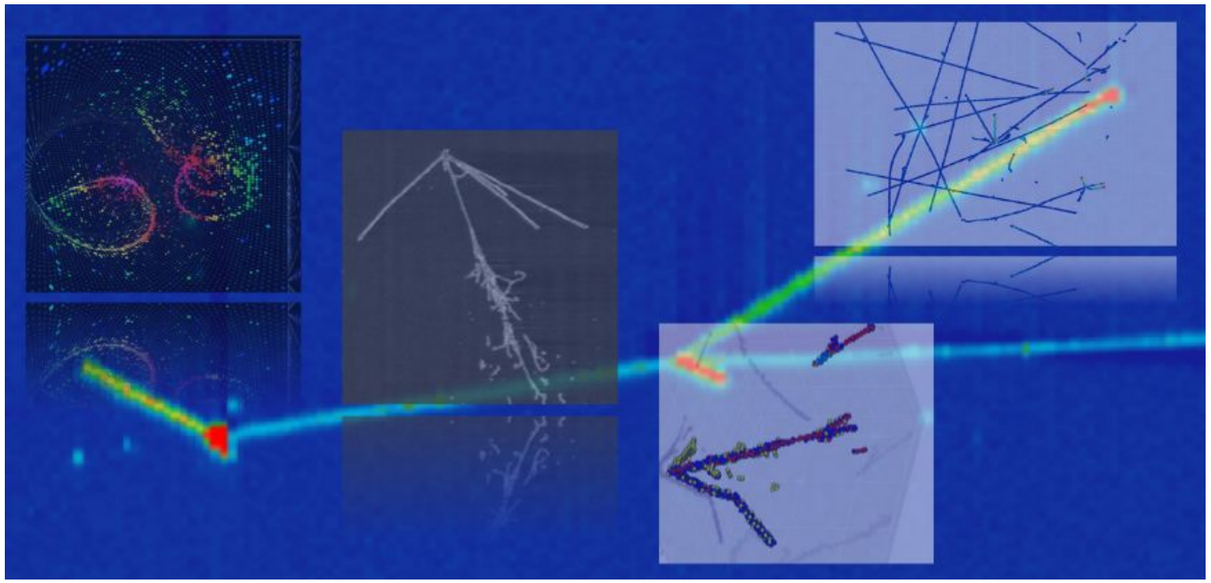
- Non-Gaussian uncertainties
- Position-dependent effects
- Time-varying calibrations
- Correlated parameters



The ML Tradeoff

Limitations of pure ML approaches:

- Black box nature
- Need large training samples
- Difficult to impose physics constraints
- Usually trained on simulation and then applied to data
- Retrain for any simulation changes



Traditional: Physics → Simulation → Data

ML: Data → Black Box → Physics

White Box Simulation

Hybrid approach combining physics knowledge with ML:

- Preserve explicit physics models where understood
- Apply neural networks only for complex, unknown components
- Modular design ensures interpretability through separation
- Physics constraints naturally incorporated where needed

Benefits:

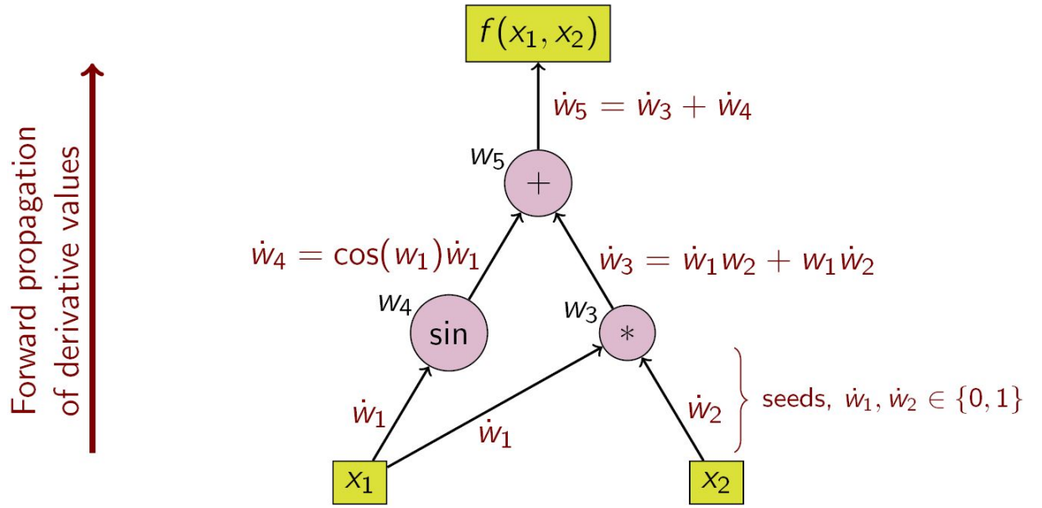
- Interpretable by design
- Requires less training data than pure ML
- More robust to simulation changes
- Directly connects to underlying physics

Challenge: How to optimize the entire pipeline?

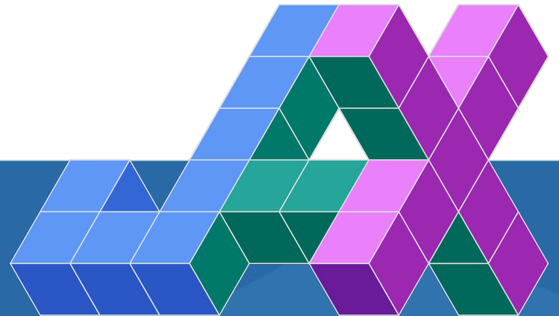
→ Need gradients through both physics and ML components

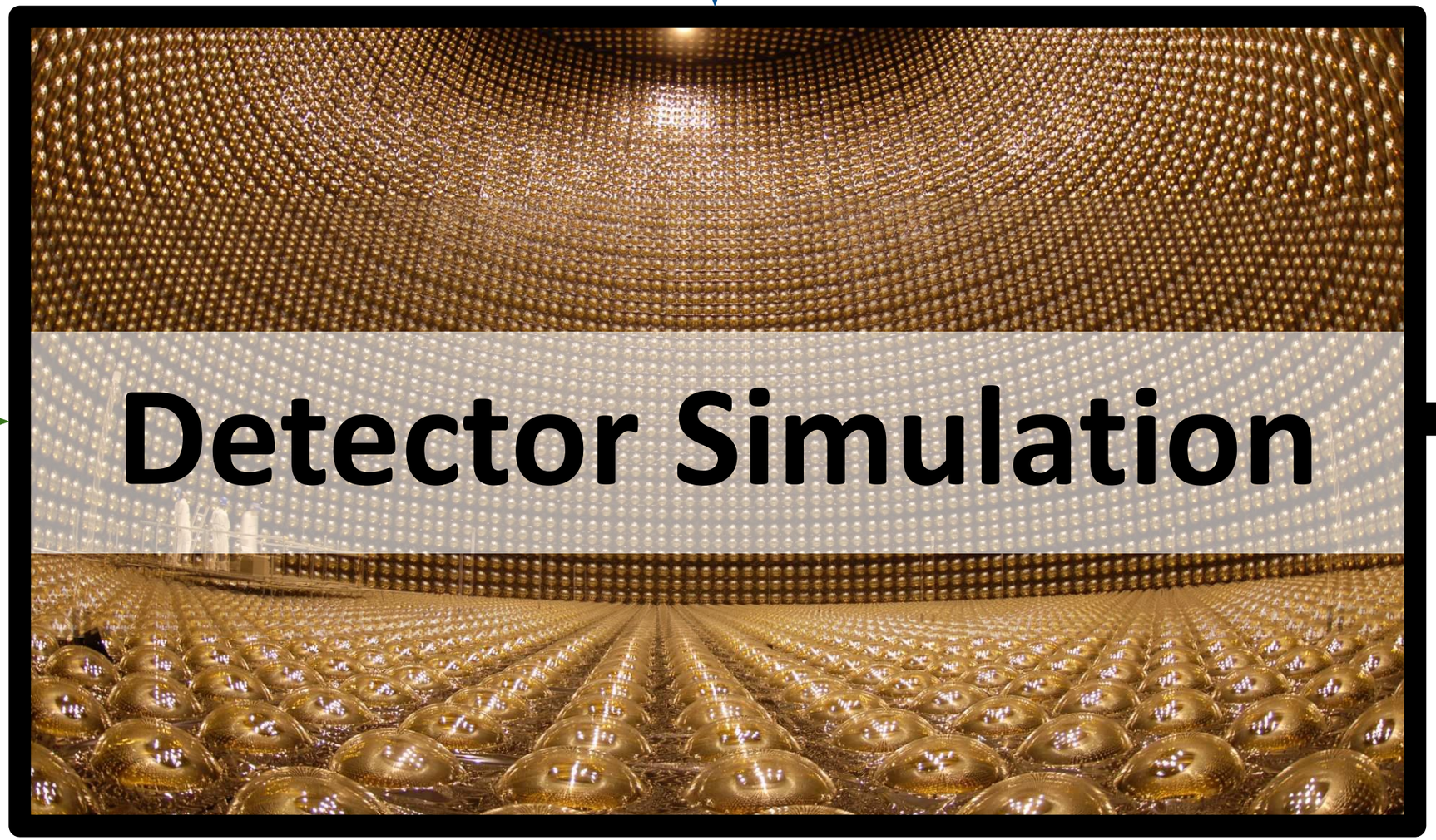
Automatic Differentiation to the Rescue

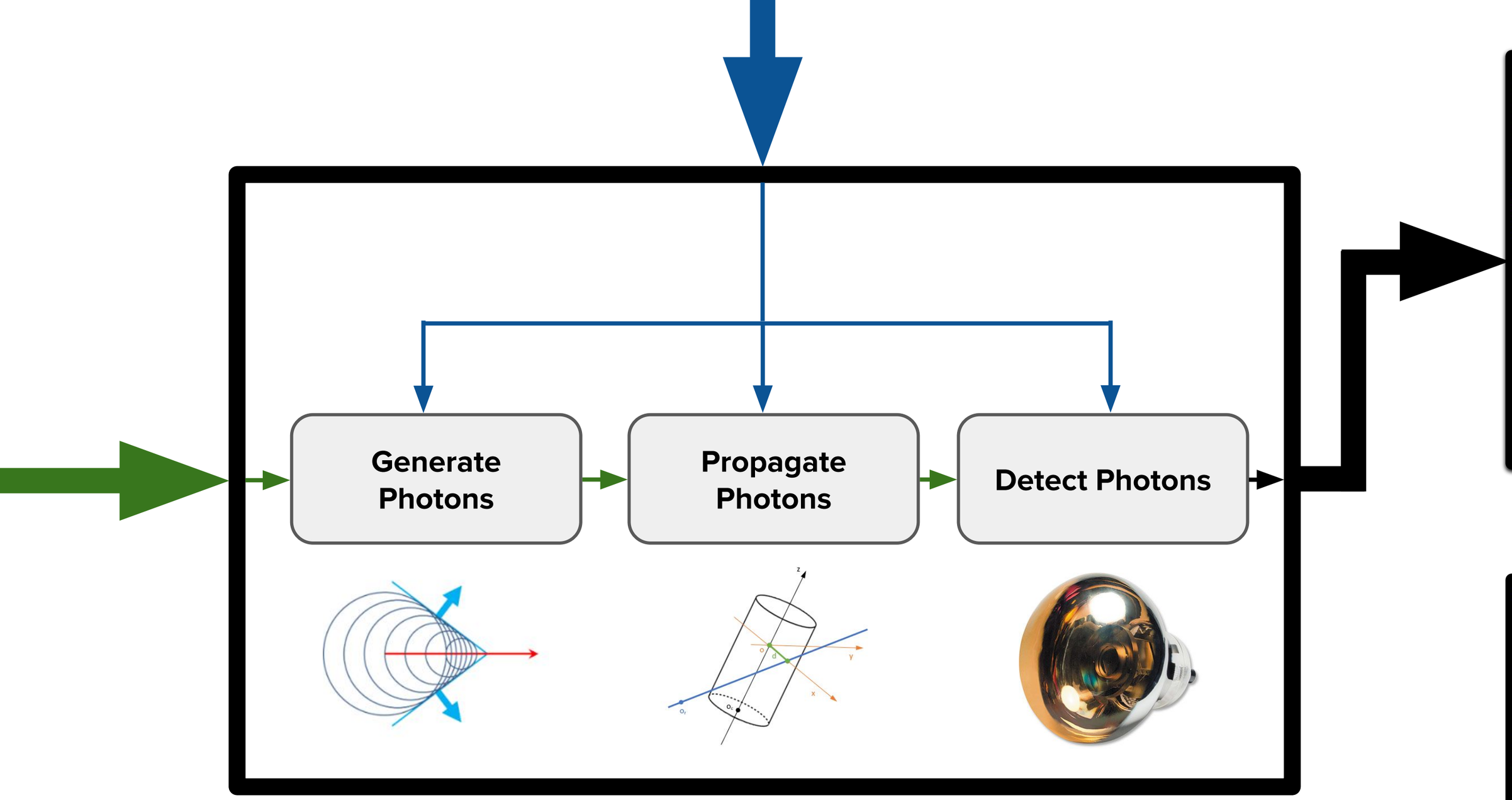
- Automatic Differentiation (AD): Precisely calculates derivatives by systematically applying the chain rule to computational operations
- Achieves machine-precision gradients, free from numerical approximation errors
- Forms the computational backbone of today's neural network architectures
- Implemented in JAX to harness GPU parallelism, vectorization, and compilation for AD calculations
- 100-1000× speed improvement on GPUs. O(10ms) full simulation



$$
\begin{aligned}
y &= f(x_1, x_2) \\
&= x_1 x_2 + \sin x_1 \\
&= w_1 w_2 + \sin w_1 \\
&= w_3 + w_4 \\
&= w_5
\end{aligned}
$$




Detector Simulation

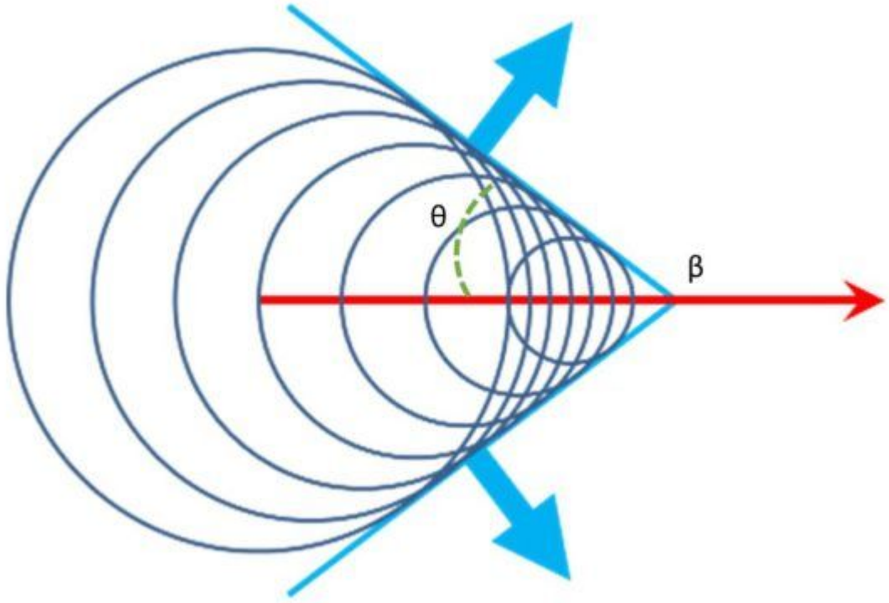

Generate Photons
Propagate Photons
Detect Photons

From Particles to Photons

Generate Photons | Propagate Photons | Detect Photons

From particles to photons:

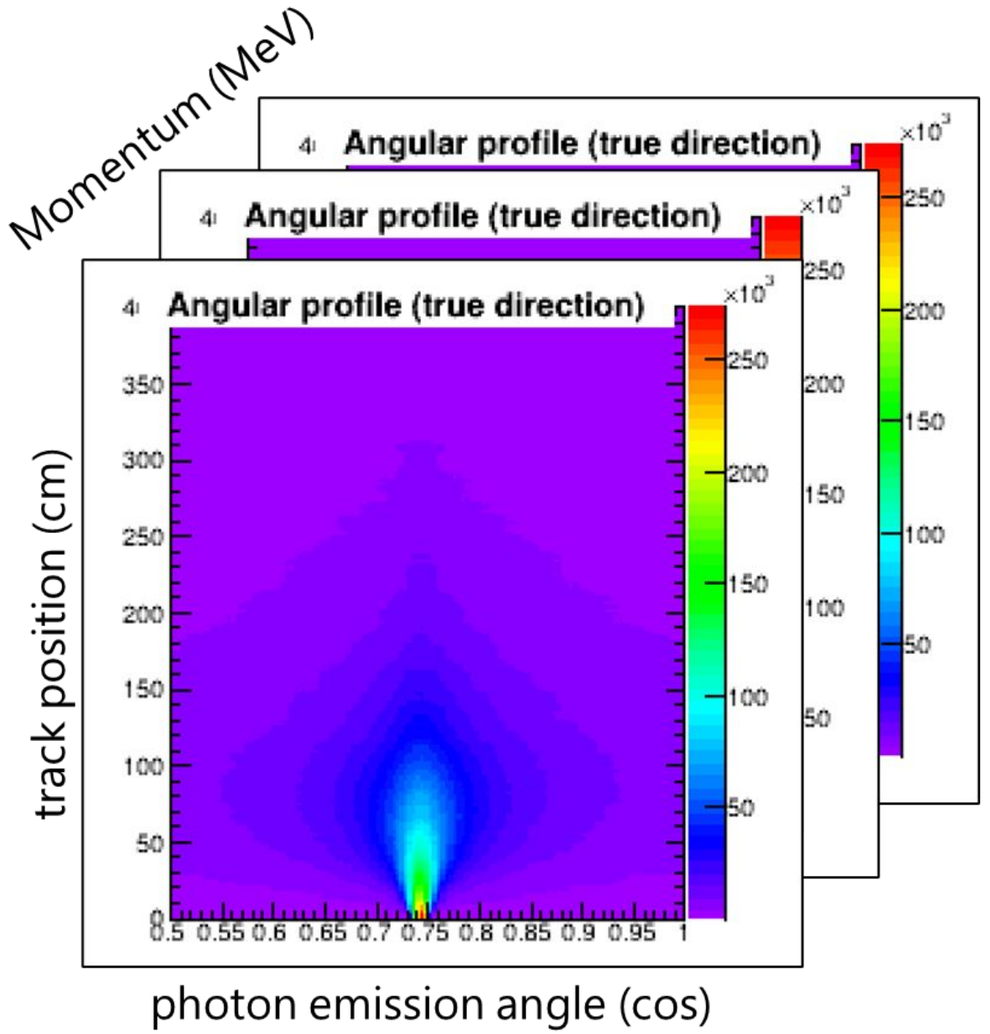
- Cherenkov radiation occurs when charged particles exceed the speed of light in water
- Particle information (position, momentum, energy) is translated into a sequence of emission points throughout the medium
- At each point, photons are generated following Cherenkov physics with appropriate cone opening angles and intensity
- Secondary physical corrections are incorporated through surrogate modeling based on simulation data



CherenkovSiren

Generate Photons | Propagate Photons | Detect Photons

- Cherenkov Profiles are 3D histograms capturing complete physics from momentum to photon generation, allowing sampling along particle paths.
- Colleagues developed a neural network version using SIREN (3-input neural network) to learn these profiles.

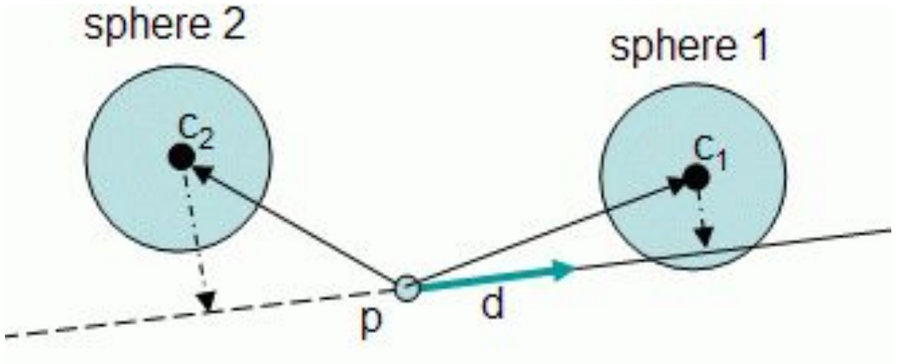


Ray Tracing

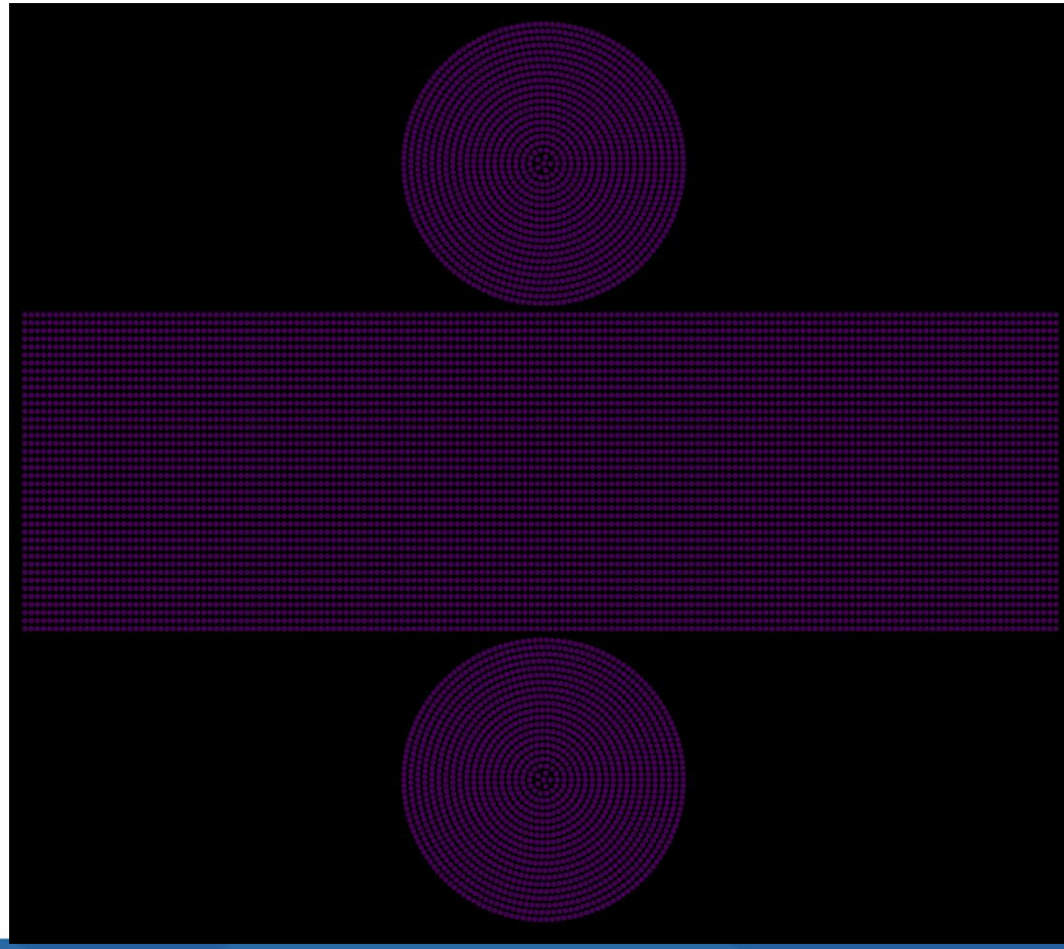
Generate Photons | Propagate Photons | Detect Photons

The biggest bottleneck by far in the detector simulation is the ray tracing.

The detectors are assumed to line up the walls and caps of a cylinder. We can use ray sphere intersection to find the final location each ray



r = 4m, h = 6m, pmt_size = 40cm, ~10k pmts

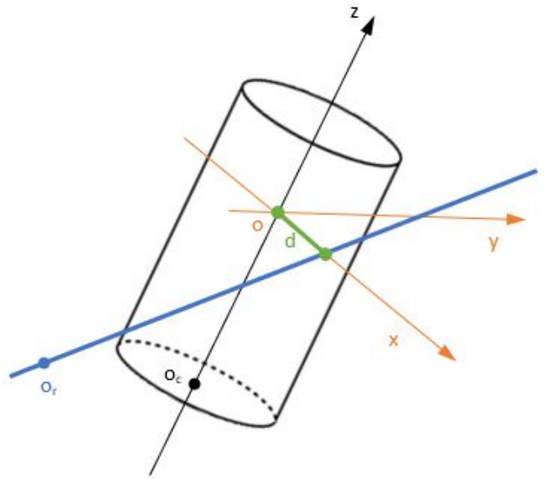


Filtered Intersection

Generate Photons | Propagate Photons | Detect Photons

Let's make our simulation faster:

- Checking every PMT for each photon is computationally wasteful
- Key insight: We can pre-filter likely intersections
- Solution: Cylindrical grid-based acceleration structure

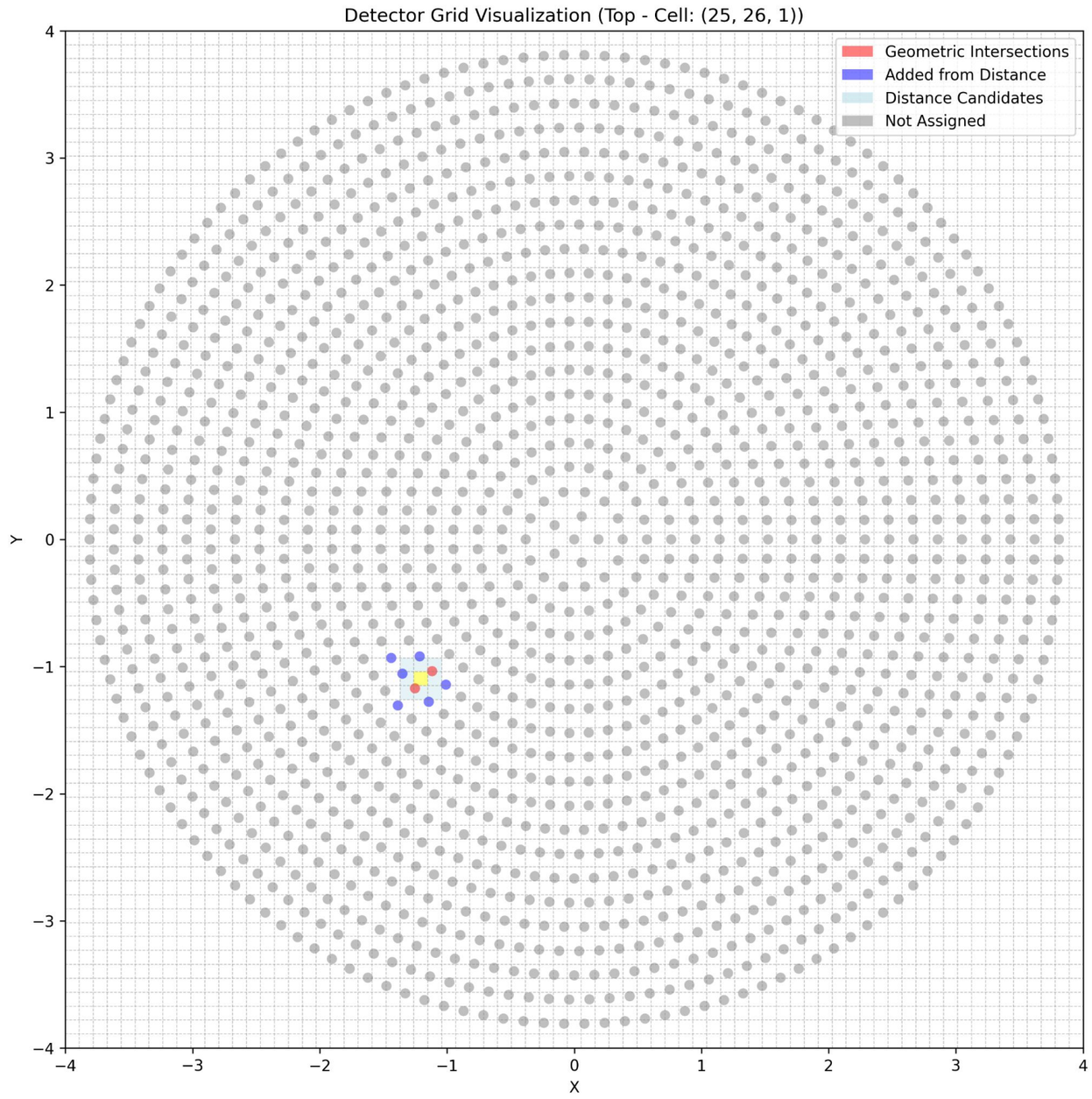


Our approach:

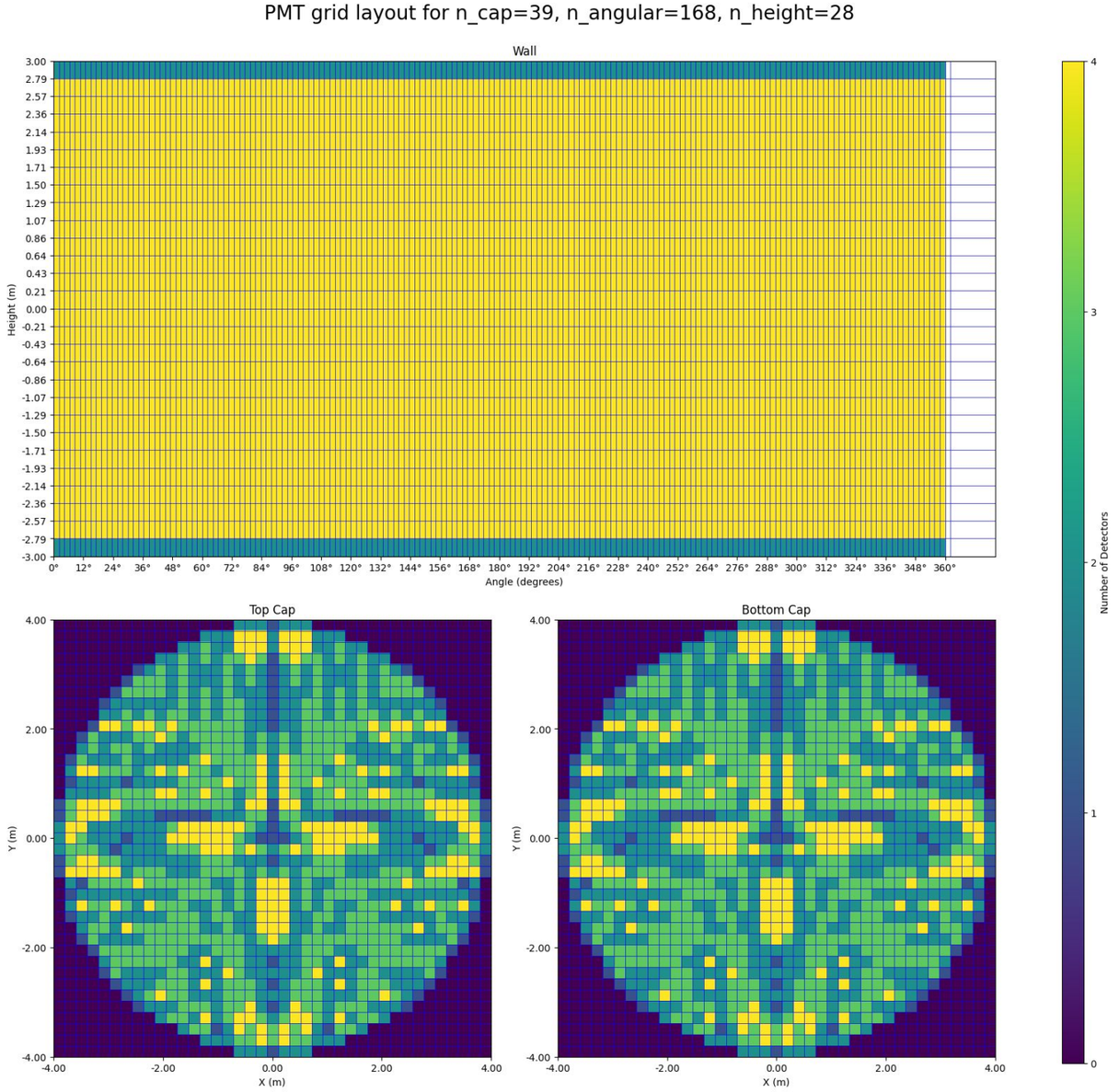
1. First pass: Quick check for cylinder wall or cap intersection
2. Second pass: Use grid system to identify relevant PMTs
3. Final pass: Detailed intersection calculations only for candidate PMTs

The grid system is extremely efficient leading to O(10ms) overall simulation runtime on a GPU

Detector Grids

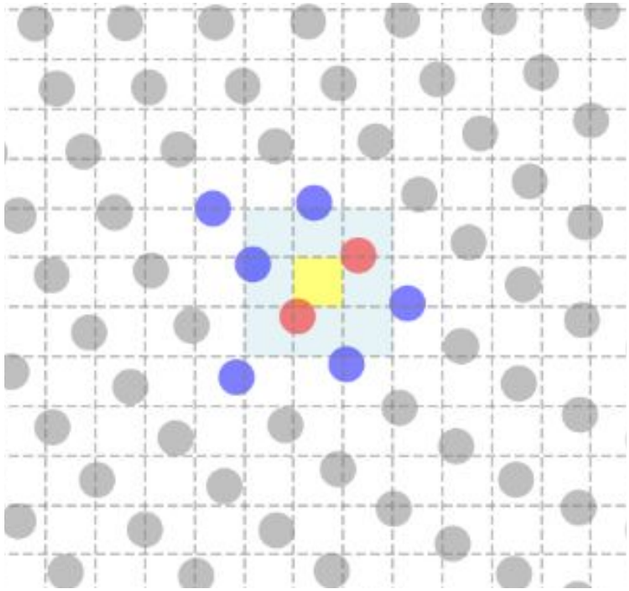


Number of intersected PMTs per grid



From Discrete to Continuous Detection

Generate Photons | Propagate Photons | Detect Photons

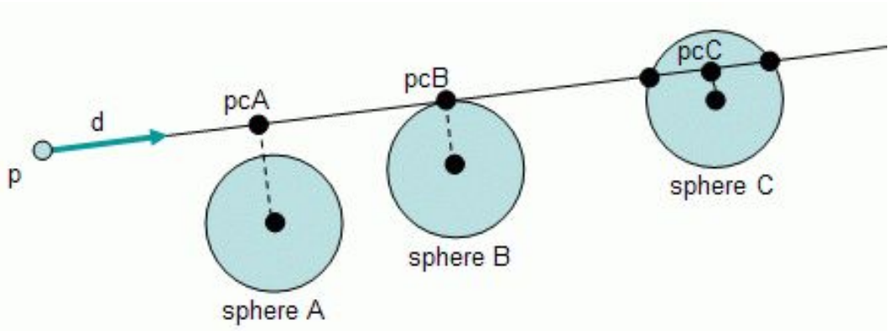


Traditional ray tracing problem: photons either hit or miss PMTs completely.

Our approach:

- Replace hard binary decisions with smooth probability weights. One Photon → Multiple PMTs
- Calculate weights based on proximity between photon path and PMT
- Adjust sensitivity through a tunable temperature parameter

This approach creates smoother gradients and significantly reduces noise during optimization



Photon Relaxation: Gaussians

Generate Photons | Propagate Photons | Detect Photons

Proposed Solution:

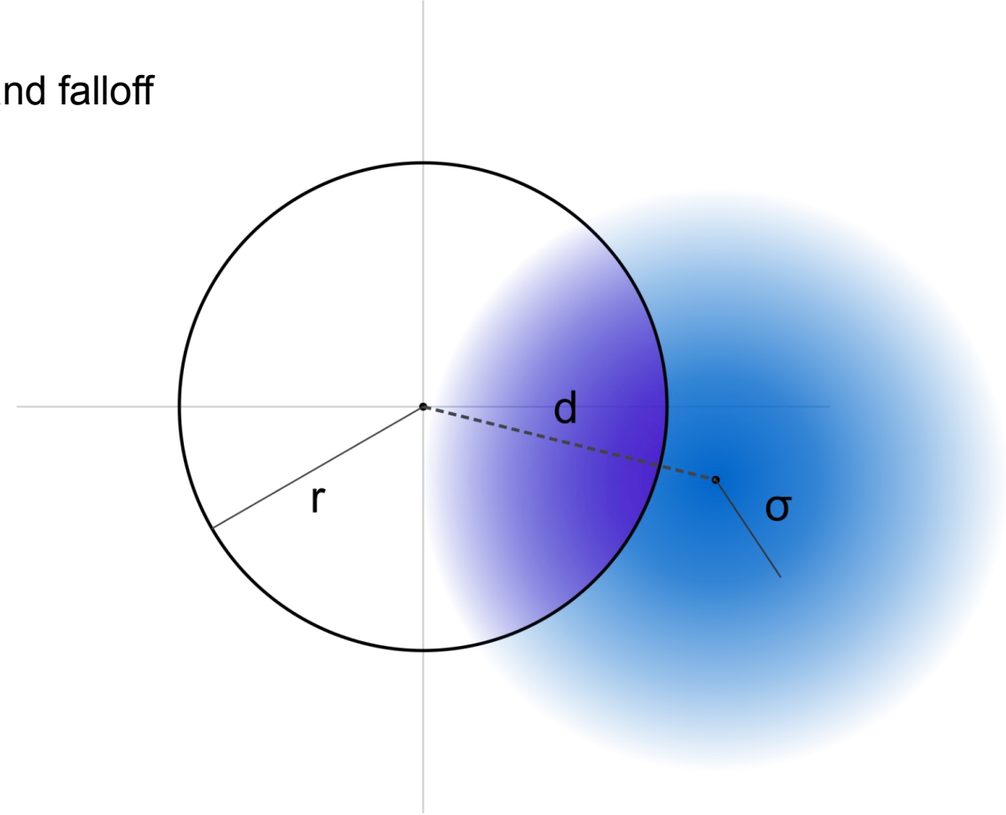
- Model the “contribution” to each PMT as the photon overlapping with a circular detector on the wall
- Represent photons as distributions with spatial falloff
- Requires precise calculation of 2D overlap integral between the circle and falloff

Integration Challenge:

- Full 2D overlap integral needed
- Parameters: PMT radius (r) and distribution width (sigma)

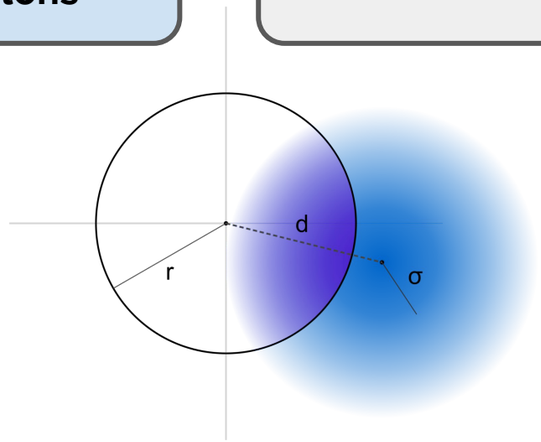
Optimization Strategy:

- Leverage problem symmetry
- Pre-calculate overlap as function of distance (d)
- Distance measured to closest point on trajectory
- Reduces computational overhead significantly



Overlap Calculation

Generate Photons | Propagate Photons | Detect Photons



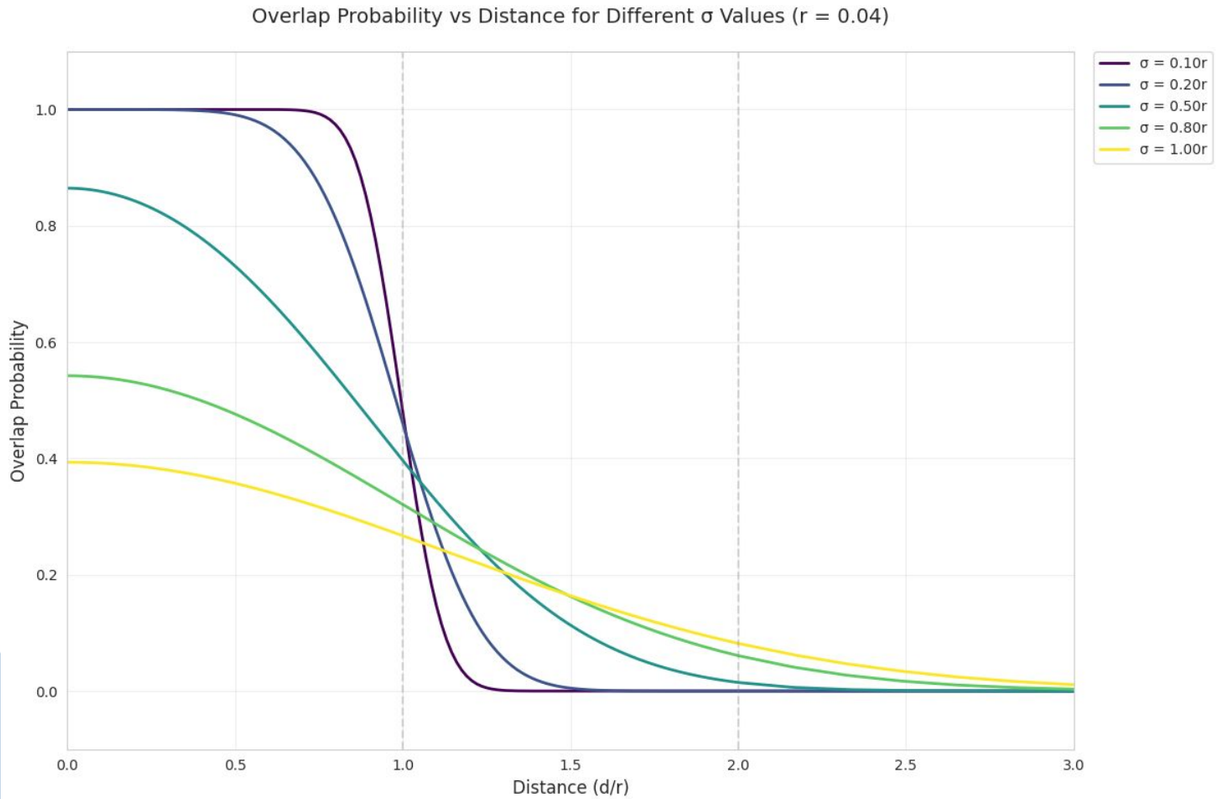
1D differentiable lookup table in JAX.

Save the overlap values and the gradient at that location and interpolates between them

Overlap Probability Values at Selected Distances:

Overlap Probability Values at Selected Distances:

σ	d = 0	d = 2r	d = 3r	d = 4r
0.10r	1	5.37e-24	0	0
0.20r	1	1.99e-07	1.07e-23	0
0.50r	0.865	0.0147	1.87e-05	6.18e-10
0.80r	0.542	0.0609	0.00322	4.28e-05
1.00r	0.393	0.0819	0.0109	0.000604



Realistic PMT Simulation

Generate Photons | Propagate Photons | Detect Photons

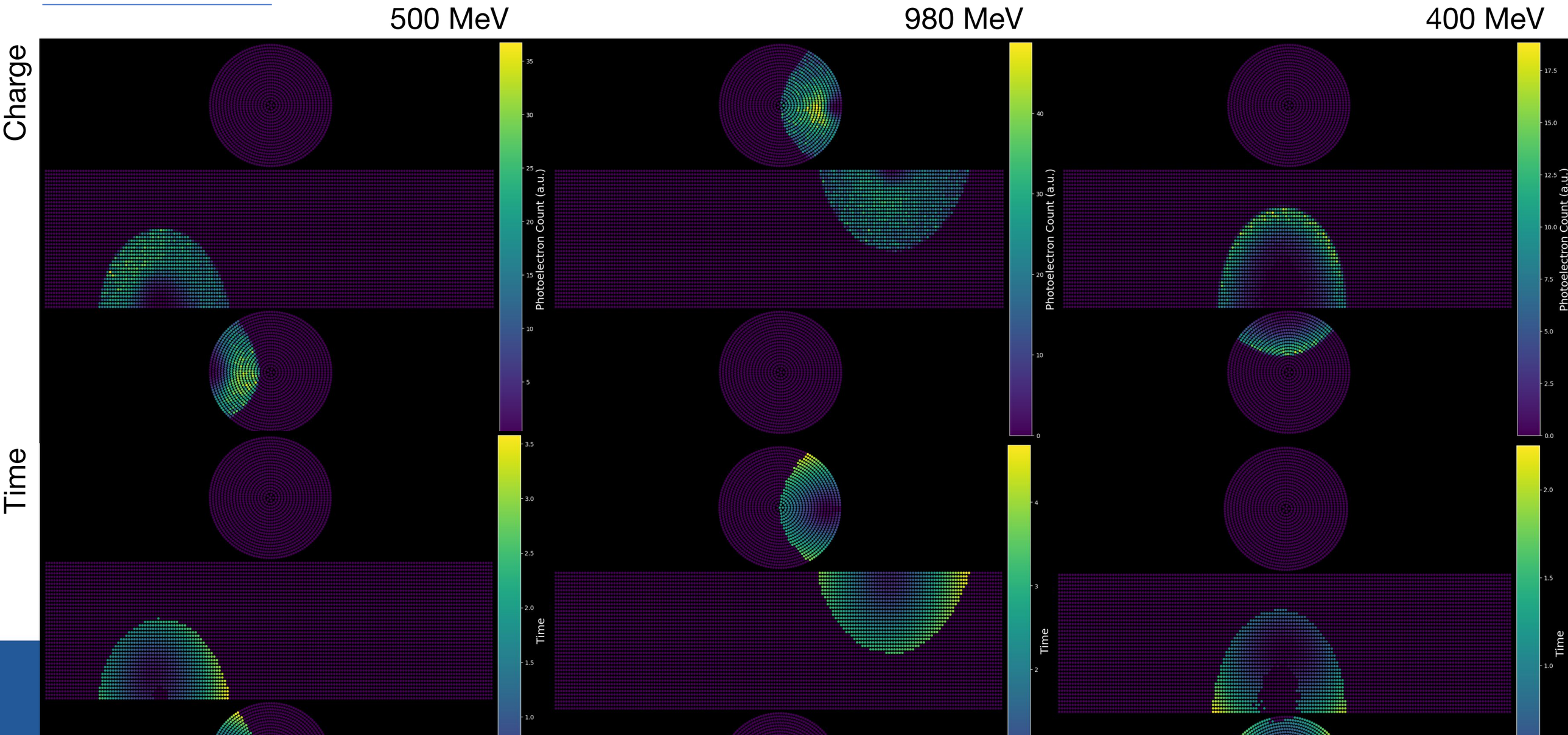
From photons to signals:

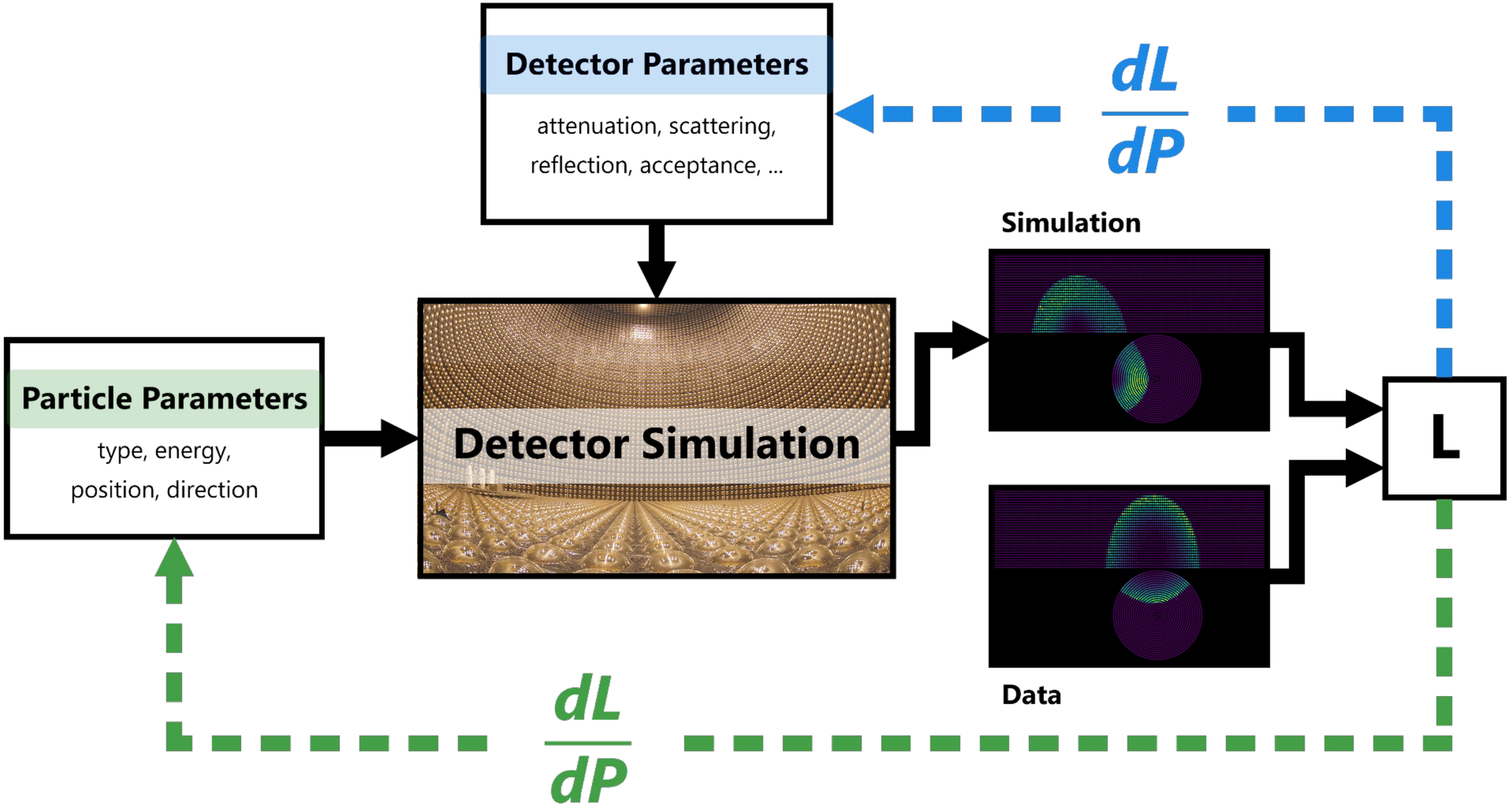
- Accumulate weighted contributions per PMT
- Calculate charge and arrival times to obtain hits
- Framework easily extendable for PMT response:
  - provides per photon per PMT surface hit location and normal
  - could also be modified easily to get waveforms (currently only hits)

Future capabilities:

- Train response models on individual PMT calibration data
- Fine-tune using real detector data (per PMT variation)

Event Displays




Detector Parameters
attenuation, scattering,
reflection, acceptance, ...
dL
dP
Simulation
Particle Parameters
type, energy,
position, direction
Detector Simulation
L
Data
dL
dP

Loss Construction

What is a Loss Function?

- Quantifies the mismatch between simulation predictions and actual detector data
- Smaller values indicate better agreement between simulation and reality

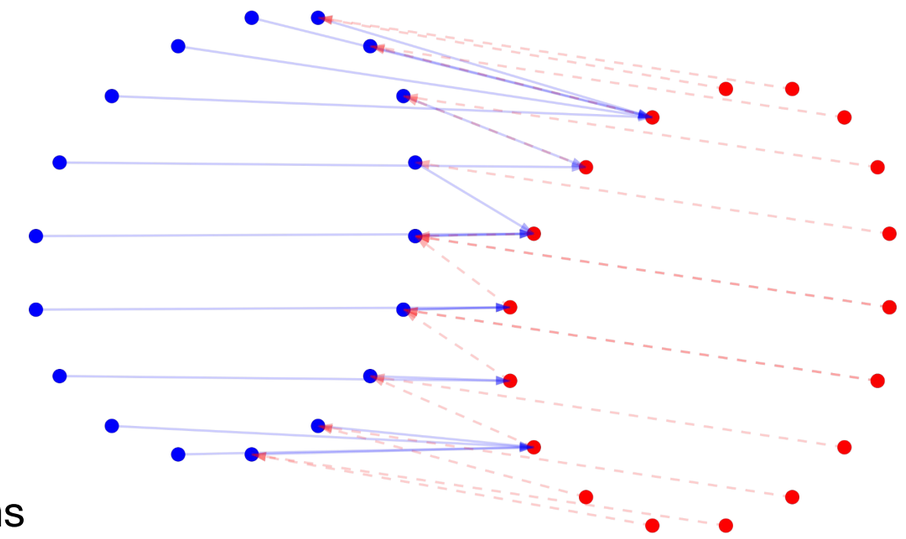
Our Approaches:

1. PMT-Level Matching Loss

- All-to-all PMT distance-based charge matching
- Uses softmin to find closest PMT pairs between simulation and data
- Combines both time and charge comparisons

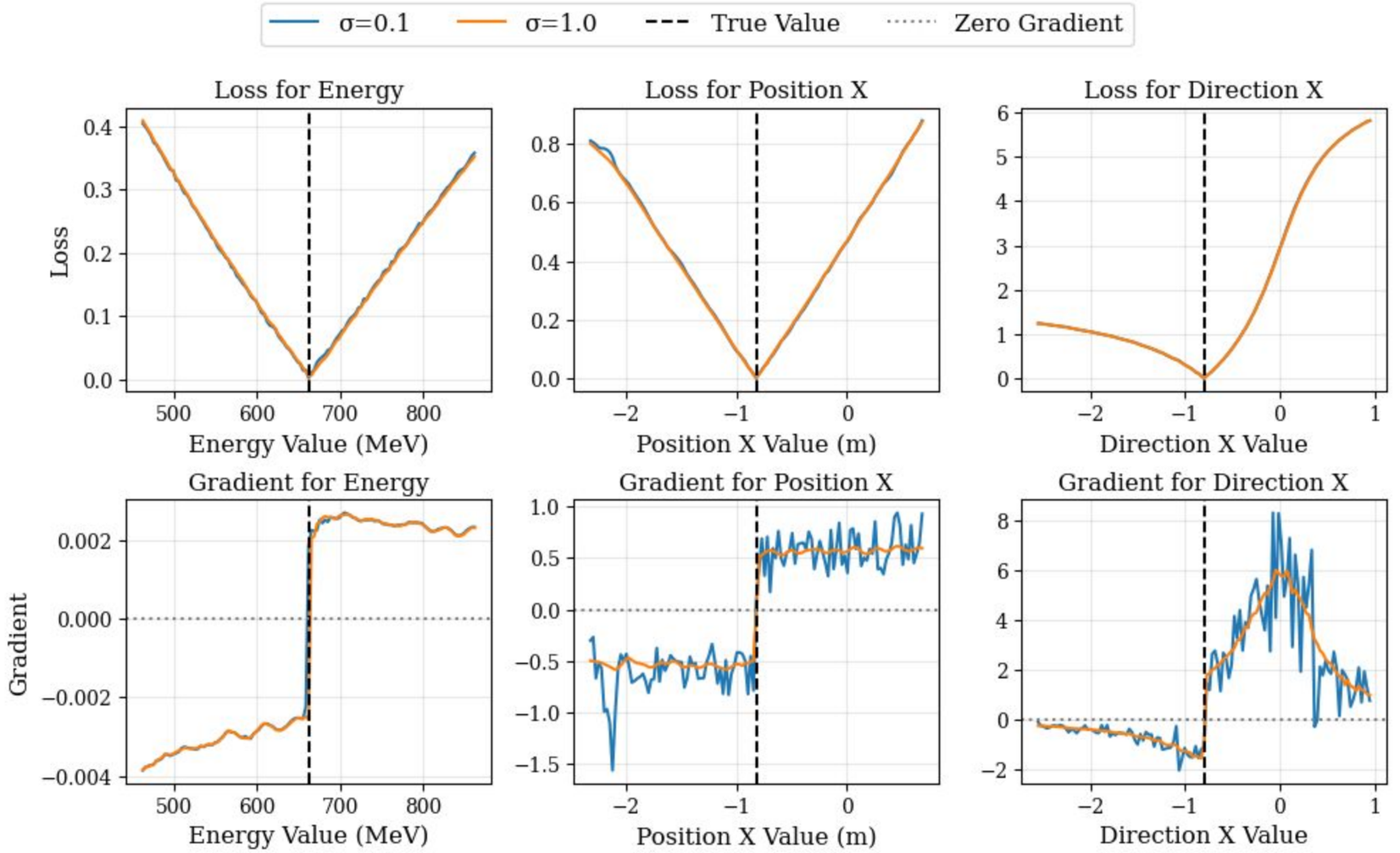
2. Distribution-Based Loss

- Centroid Loss: Measures spatial agreement using charge-weighted positions
- Intensity Loss: Evaluates overall energy scale matching
- Temporal Loss: Compares timing spread characteristics



One Parameter Gradients

- The first sanity check we can do, is freeze all parameters of the simulation, and alter one at a time.
- Reconstruction Parameters:
  - Energy
  - 3D Position
  - Direction (represented as a 3D vector)

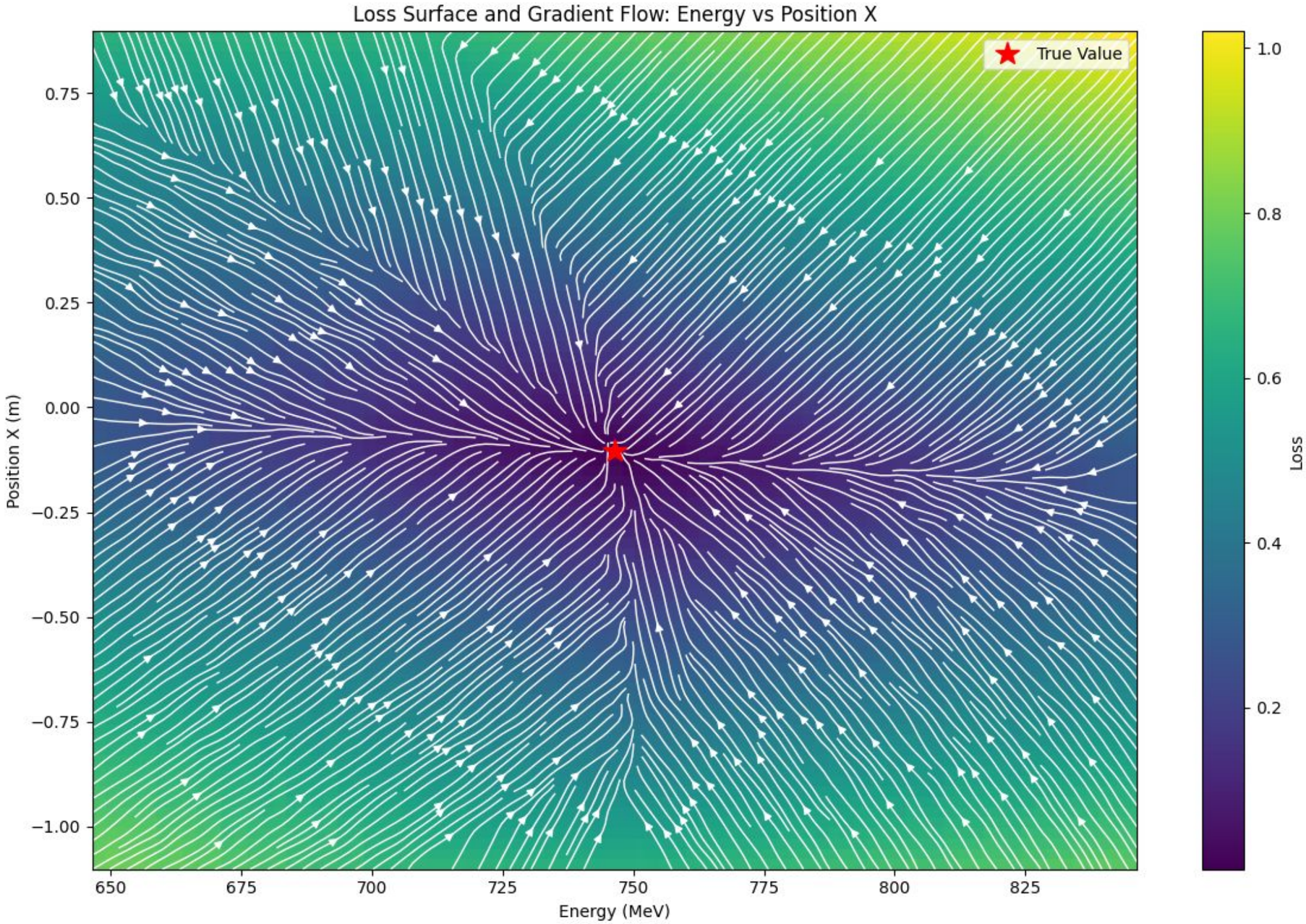


2 Parameter Variation

We can vary two parameters and look at the gradients extracted. The color is the loss.

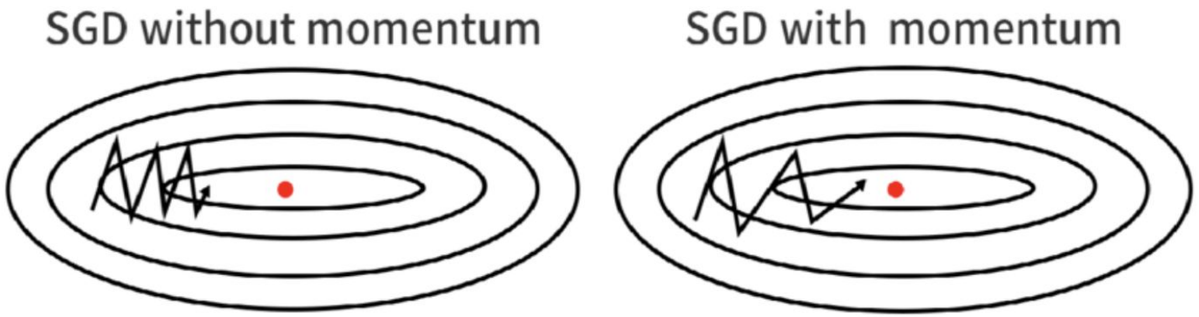
The streamlines are the values of gradients at those locations

Energy/Position

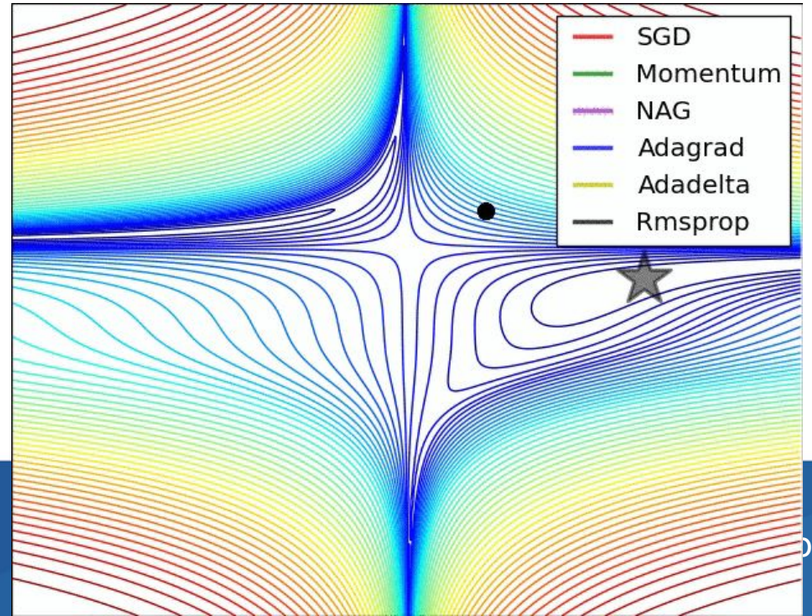


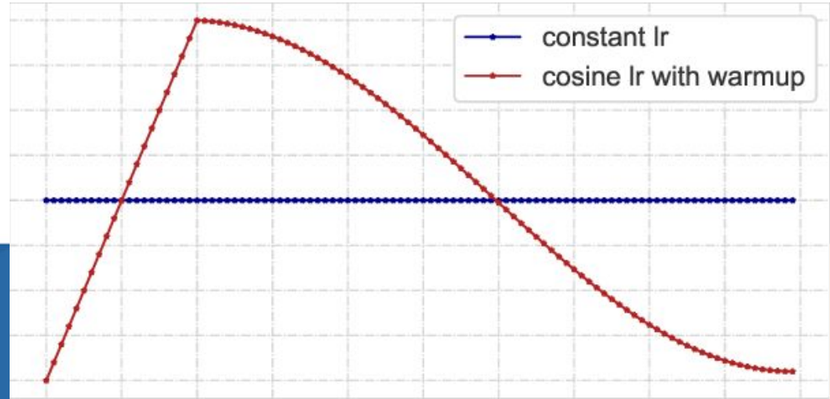
Training code implementation

- Training using Stochastic Gradient Descent with Momentum.
- Also used a learning rate schedule to get better optimization
- Example of different training optimizers



$$V_t = \beta V_{t-1} + \alpha \nabla_w L(W, X, y)$$
$$W = W - V_t$$





Reconstruction Results

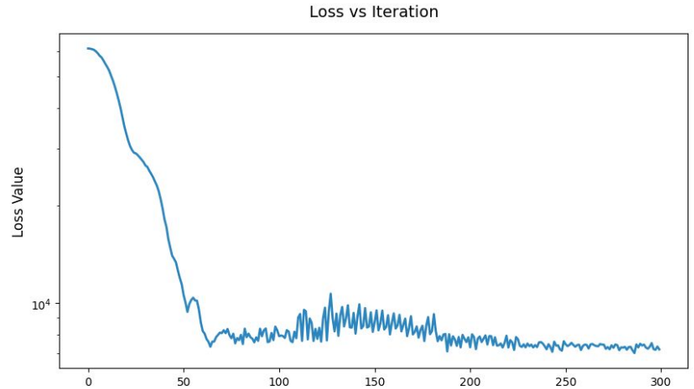
The simulation is successfully capable of reconstructing the true parameters from some initial guess

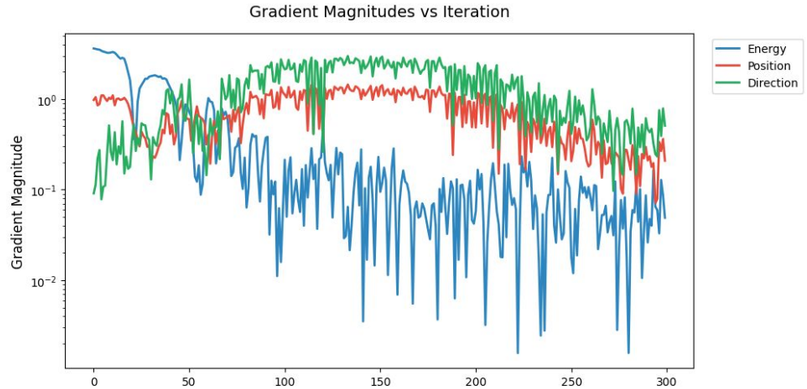
Difference in Parameters True vs Simulation:

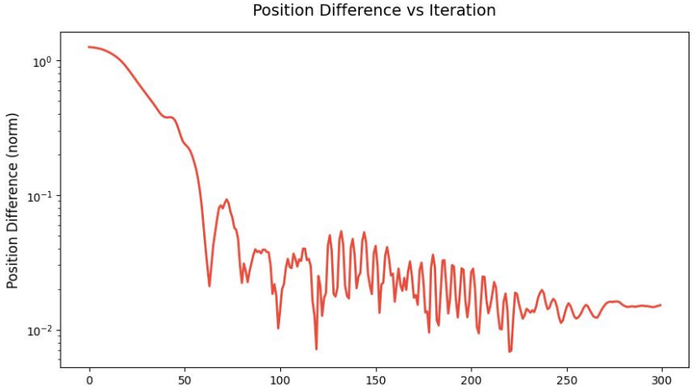
ΔEnergy: -4.14 MeV
Percent: 0.59 %

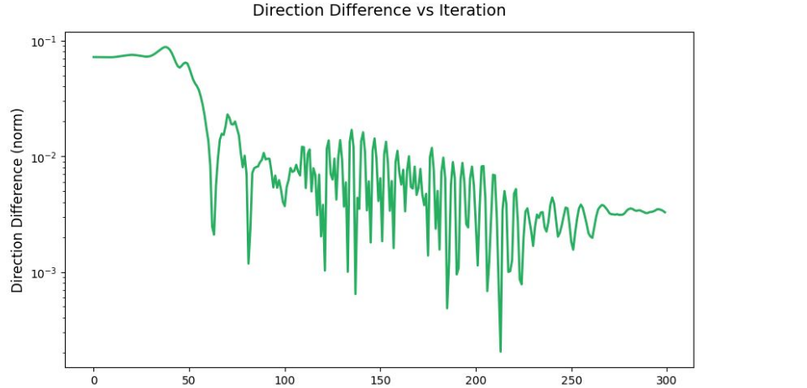
ΔPosition: 15 cm

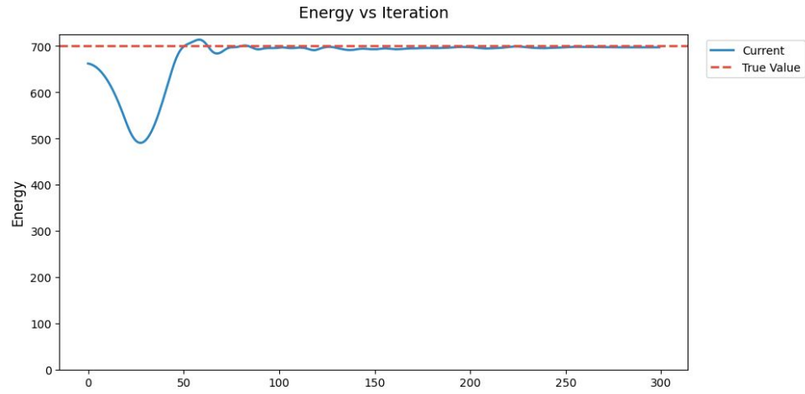
ΔDirection: 0.0032 ,
Angular: 0.186 degrees

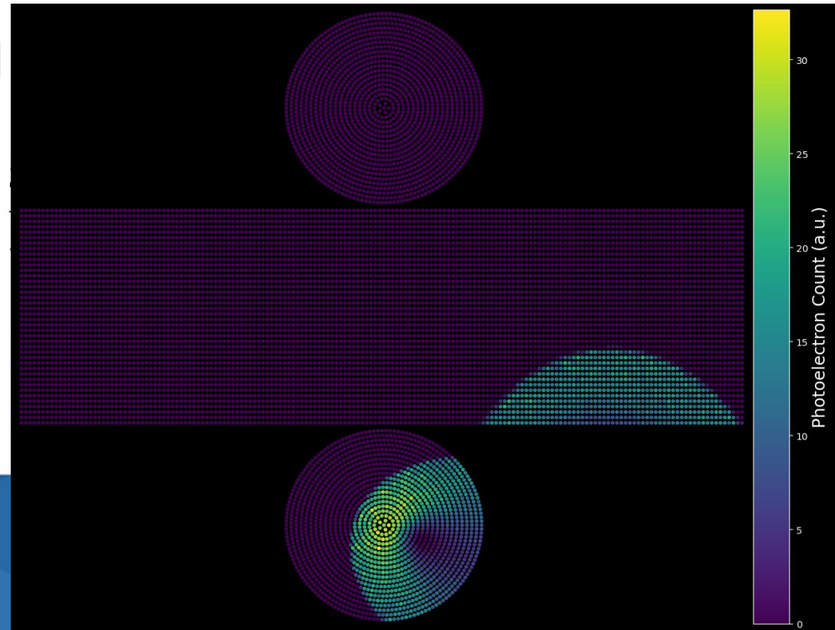






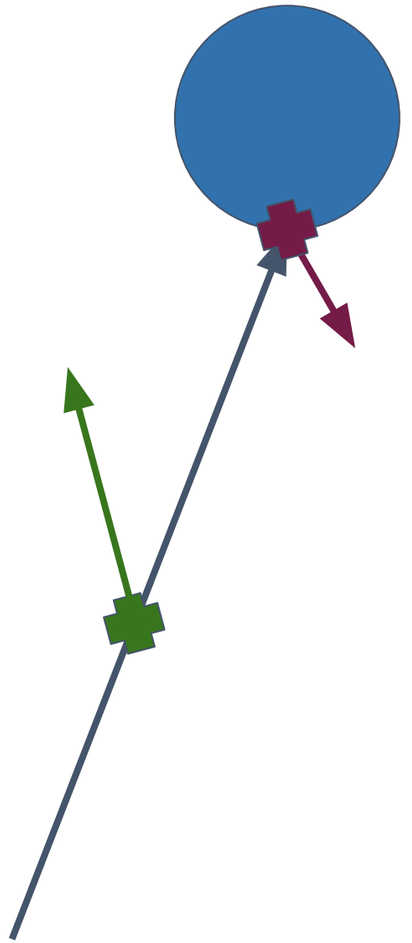






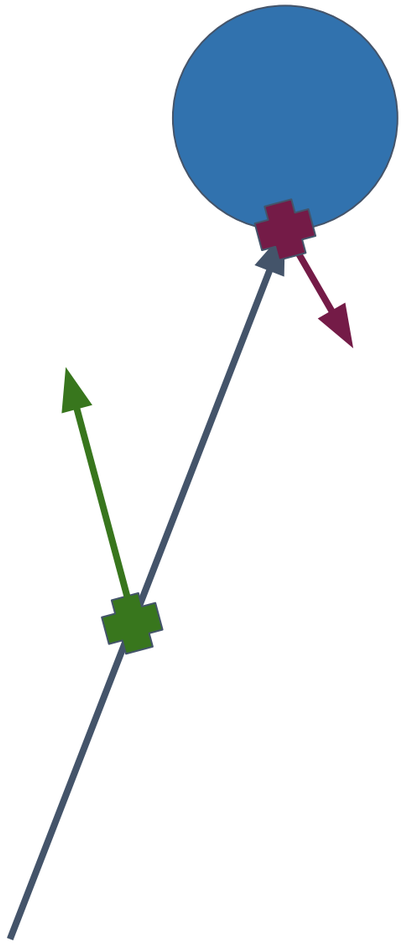
Reflections, Scattering and Attenuation

- Given the total travel distance, we add attenuation per photon
- To deal with reflections and scattering:
  - Given some reflection rate r, and effective scattering probability (based off of travel distance) s, we can use them both.
  - We always deposit some charge in each step and leave some behind. If the photon has some initial intensity I, then we have I x (1-A) deposited and I x A left. Here A is obtained from r and s.
  - Now given the I x A left, we need to choose a new starting position and direction for the new photon.



Reflections, Scattering and Attenuation

- To deal with reflections and scattering:
  - Now given the I x K left, we need to choose a new starting position and direction for the new photon.
  - for reflections:
    - new_position = hit_position
    - new_direction is flipping relative to the hit normal
  - for scattering:
    - new_position = sampled from some exponential relative to scattering distance using reparameterization trick (this makes the scattering length differentiable)
    - new_direction = sampled from rayleigh scattering
  - Given r and s, we do gumbel softmax to have smooth relaxation of discrete sampling for combining the two

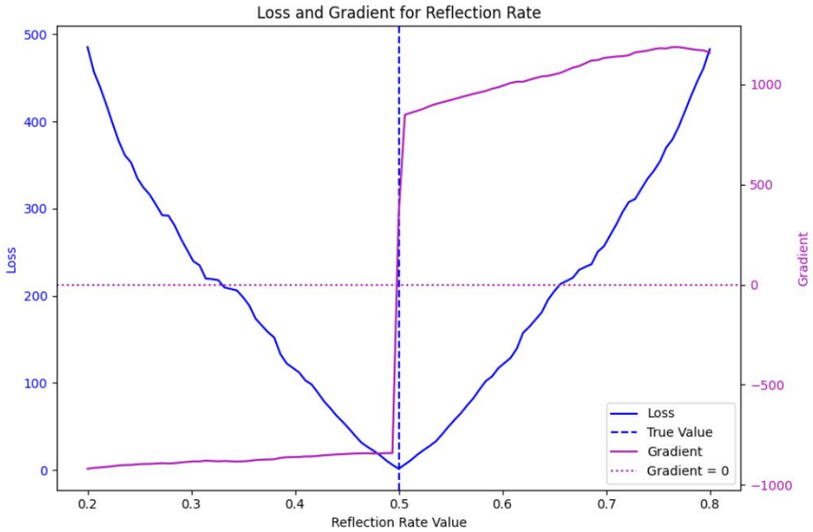


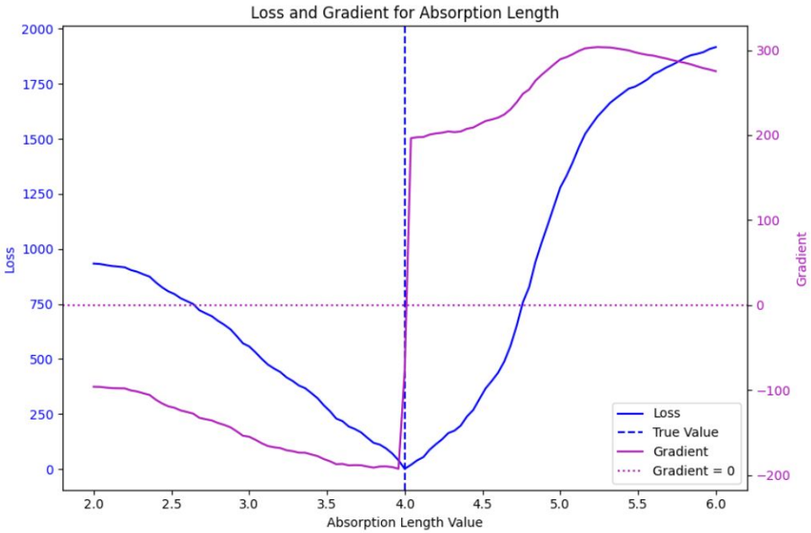
new_position = GS(s, , r , )

new_direction= GS(s,  , r , )

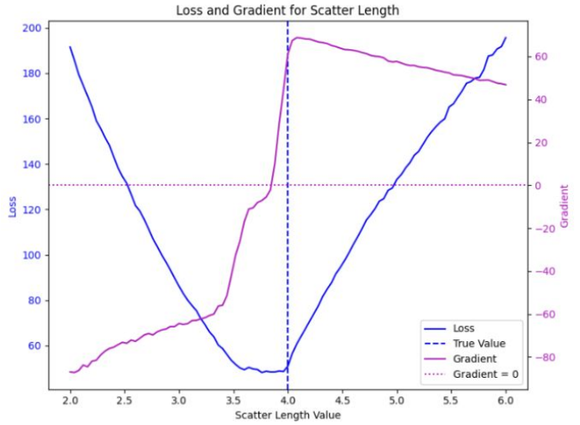
Preliminary Results

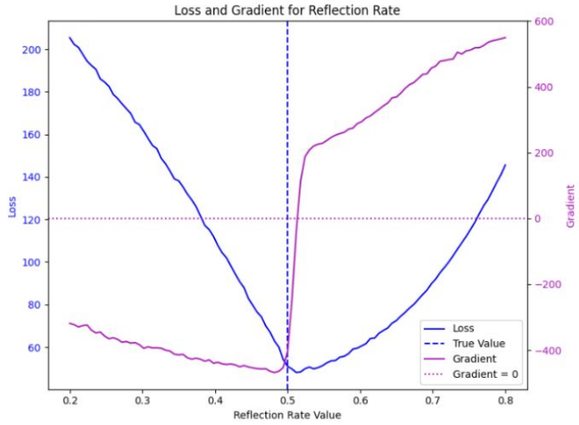
1D gradients for reflection + attenuation

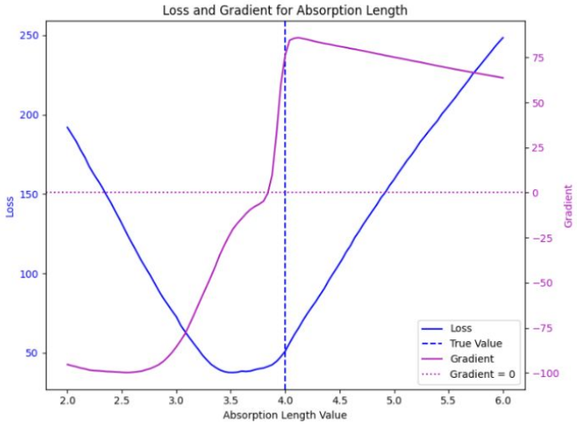




1D gradients for reflection + attenuation + scattering







Conclusion

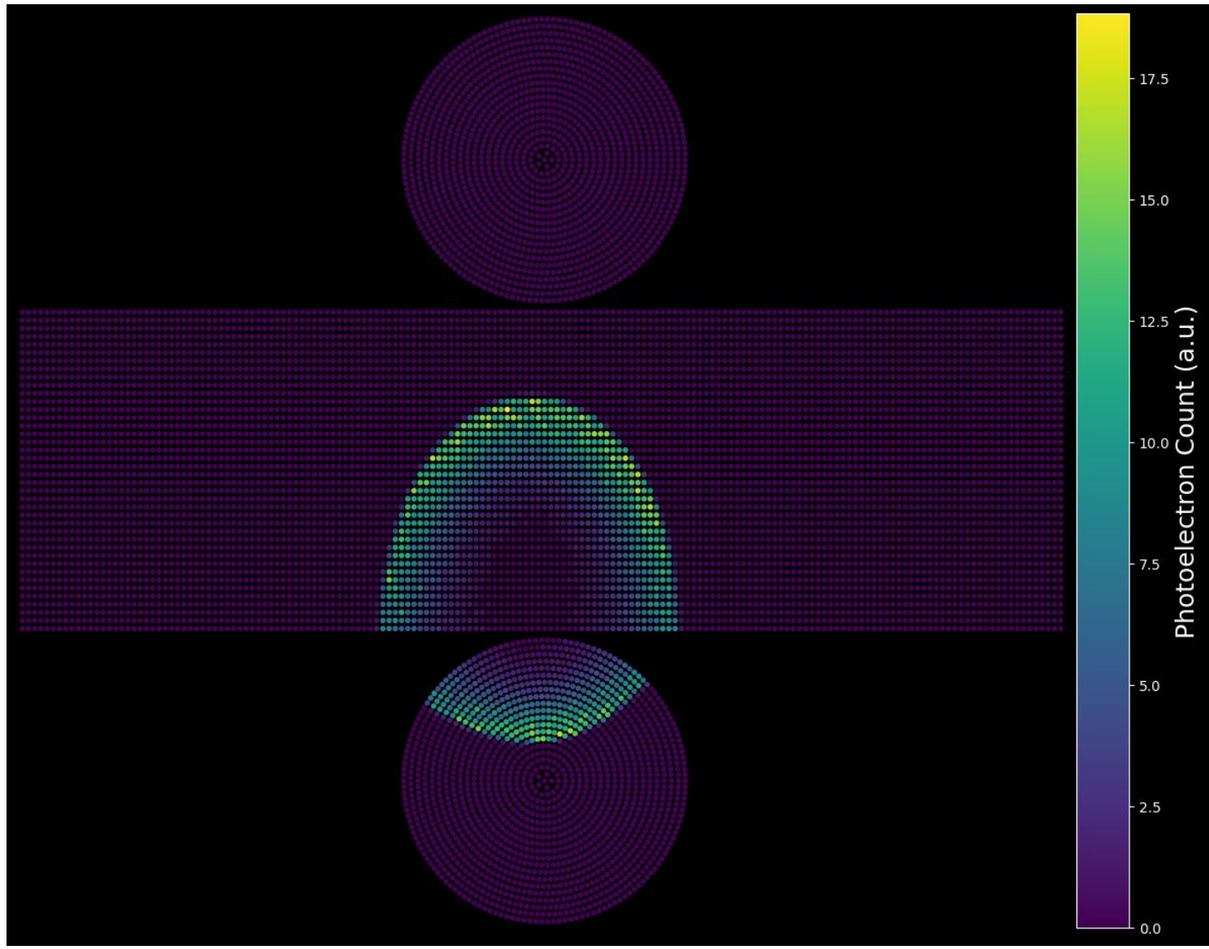
Differentiable simulation offers a new paradigm of detector physics modeling and beyond:

- Automate calibration and reconstruction
- Enable joint-optimization with other differentiable tools (NNs)

Next Steps:

- Include Stochasticity through reflections and scattering
- Employ a more realistic geometry

My research demonstrates the first critical step toward a differentiable photon signal transport in a generic Water Cherenkov detector.



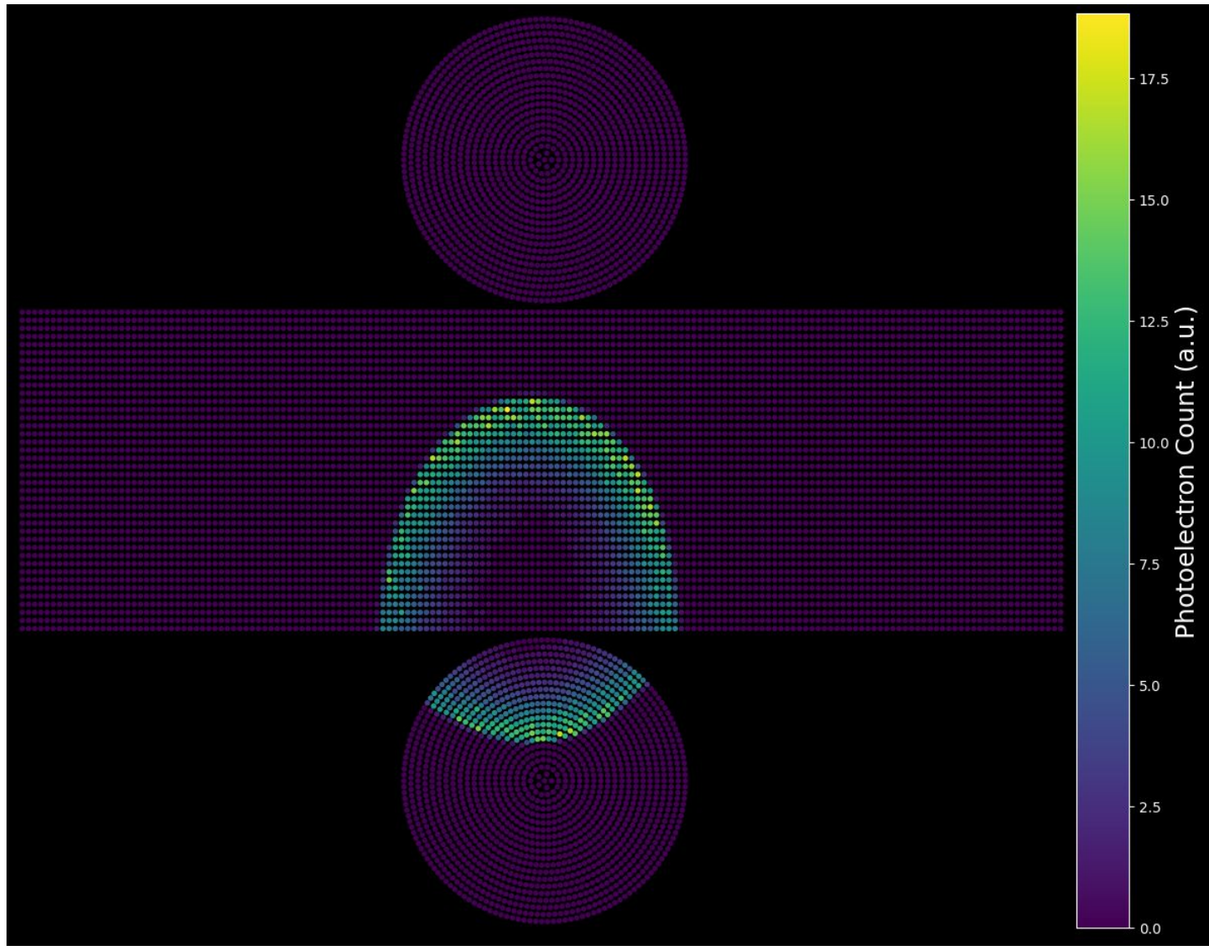
Funding Sources

Tufts UNIVERSITY

CIDeR-ML collaboration (HEP US-Japan)

Brookhaven National Laboratory

Ozaki Exchange Program



Backup

Bringing It All Together

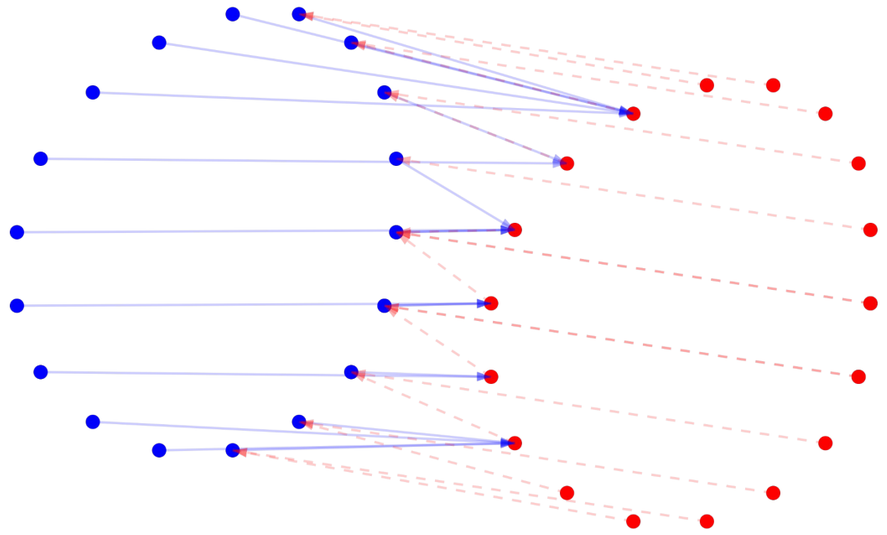
Complete differentiable simulation pipeline:

1. Generate initial parameter guess
2. Compare simulation to real data:
    - PMT charge differences
    - Normalized timing differences (time - mean_time) / std_time
3. Compute gradients automatically
4. Update all parameters simultaneously

Key advantage: Single backward pass gives gradients for ALL parameters

Loss Construction (Per PMT)

- Loss = Metric comparing how well simulation matches data that we would like to minimize
- Finding closest PMT pairs between sim/data:
  - Need differentiable matching between active PMTs
  - Use softmin weighted by distance parameter τ
  - Smooth approximation instead of direct minimum
- Time alignment:
  - Remove mean time from sim and data
  - Accounts for unknown t0 offsets
- Combined loss components:
  - L_charge: Compare charge ratios of matched pairs
  - L_time: Compare aligned timing differences
  - Total Loss = L_charge + λ L_time



Loss Construction (Centroid)

- Centroid Loss: Euclidean distance between charge-weighted centroids

$$\vec{C}_{true} = \frac{\sum_i q_{true,i} \cdot \vec{x}_{true,i}}{\sum_i q_{true,i} + \epsilon} \qquad L_{centroid} = \|\vec{C}_{true} - \vec{C}_{sim}\|$$

- Intensity Loss: Logarithmic ratio of total charges

$$L_{intensity} = \left| \log \left(\frac{\sum_i q_{sim,i}}{\sum_i q_{true,i} + \epsilon} \right) \right|$$

- Temporal Loss: Charge weighted difference between standard deviations of time distributions

$$\mu_{t,true} = \frac{\sum_i q_{true,i} \cdot t_{true,i}}{\sum_i q_{true,i} + \epsilon} \qquad \sigma_{t,true} = \sqrt{\frac{\sum_i q_{true,i} \cdot (t'_{true,i})^2}{\sum_i q_{true,i} + \epsilon}} \qquad t'_{true,i} = t_{true,i} - \mu_{t,true}$$

$$L_{time} = |\sigma_{t,true} - \sigma_{t,sim}|$$

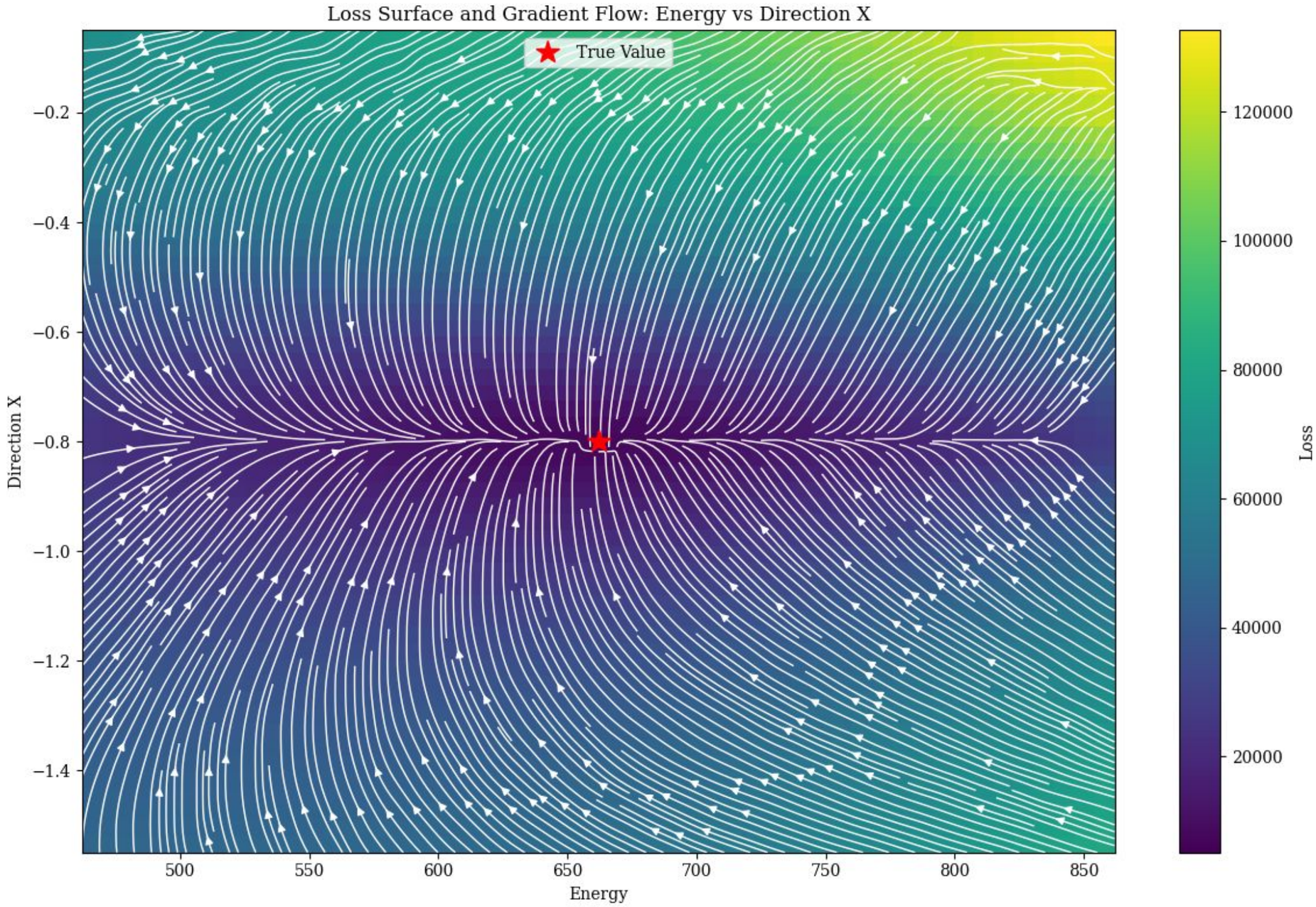
$$L_{\text{total}} = \lambda_{\text{centroid}} L_{\text{centroid}} + \lambda_{\text{time}} L_{\text{time}} + \lambda_{\text{intensity}} L_{\text{intensity}}$$

2 Parameter Variation

We can vary two parameters and look at the gradients extracted. The color is the loss.

The streamlines are the values of gradients at those locations

Energy/Direction

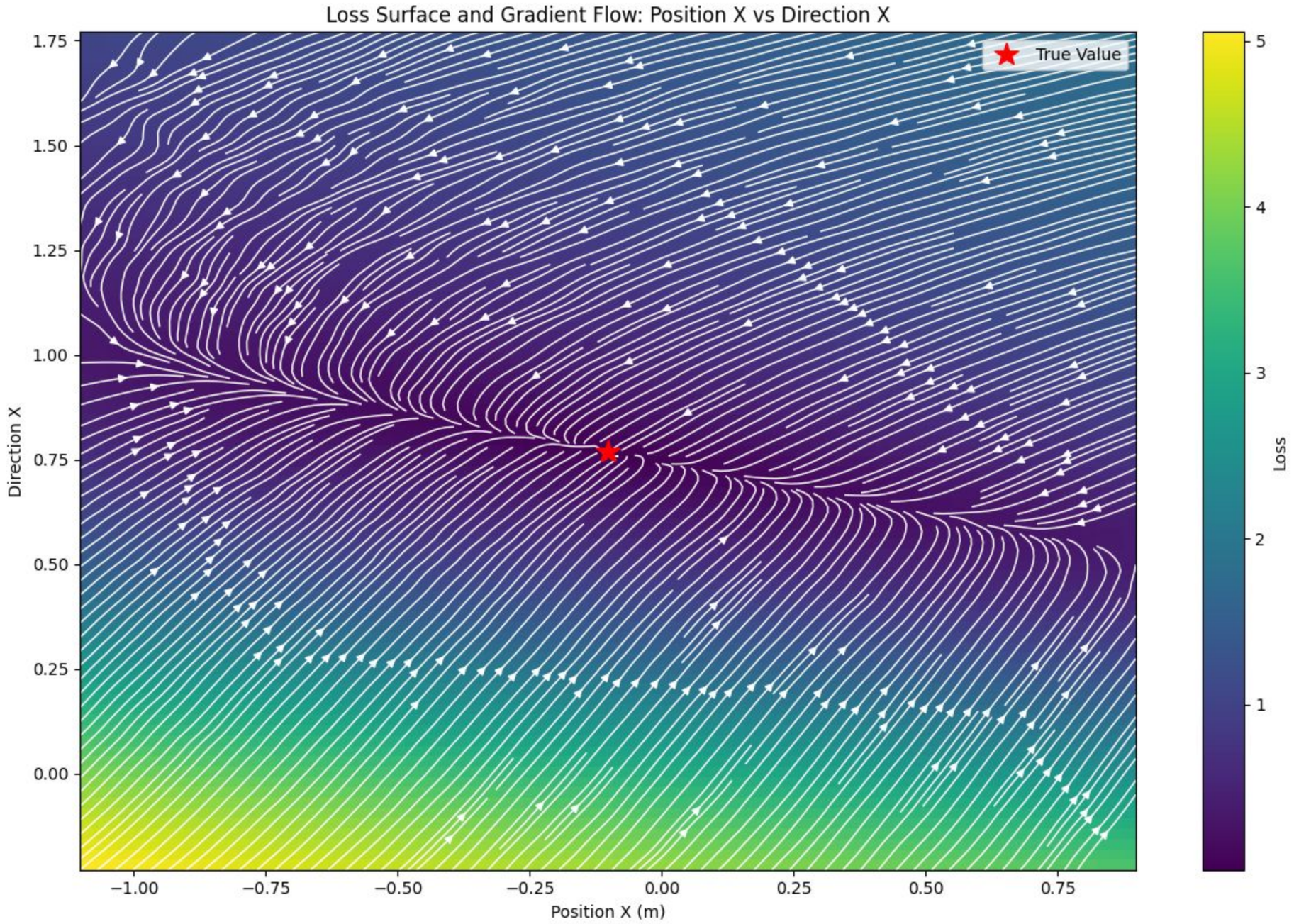


2 Parameter Variation

We can vary two parameters and look at the gradients extracted. The color is the loss.

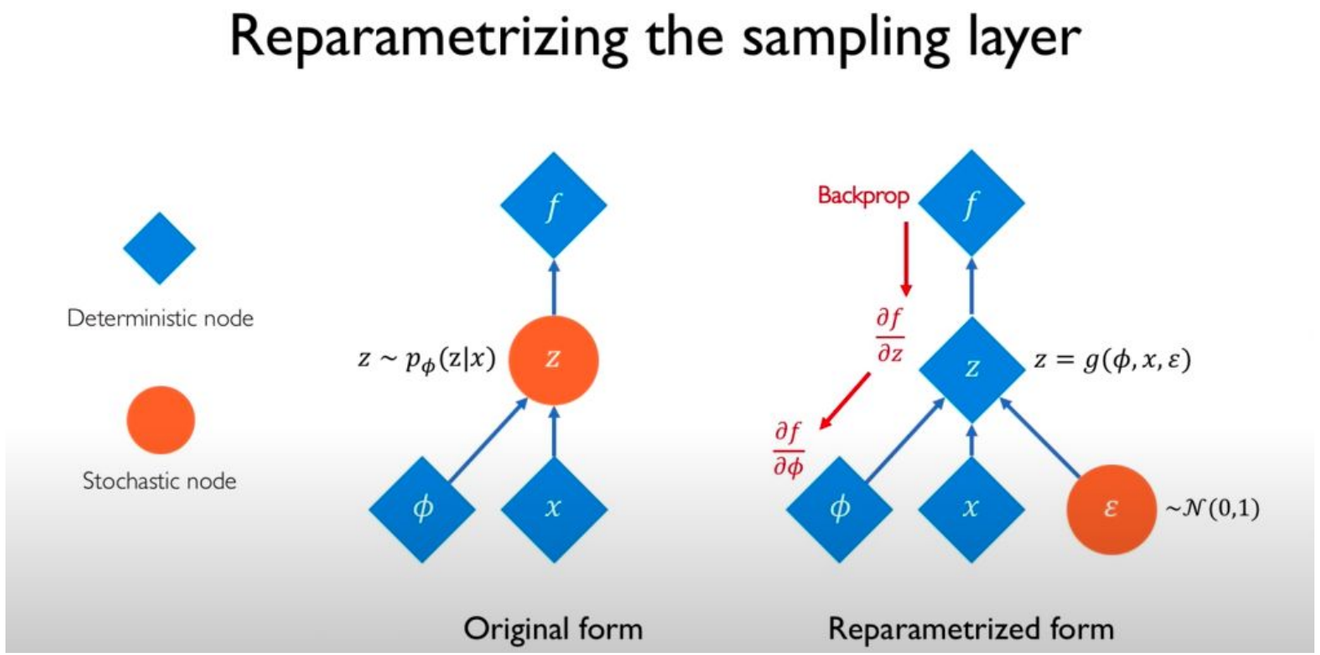
The streamlines are the values of gradients at those locations

Energy/Direction

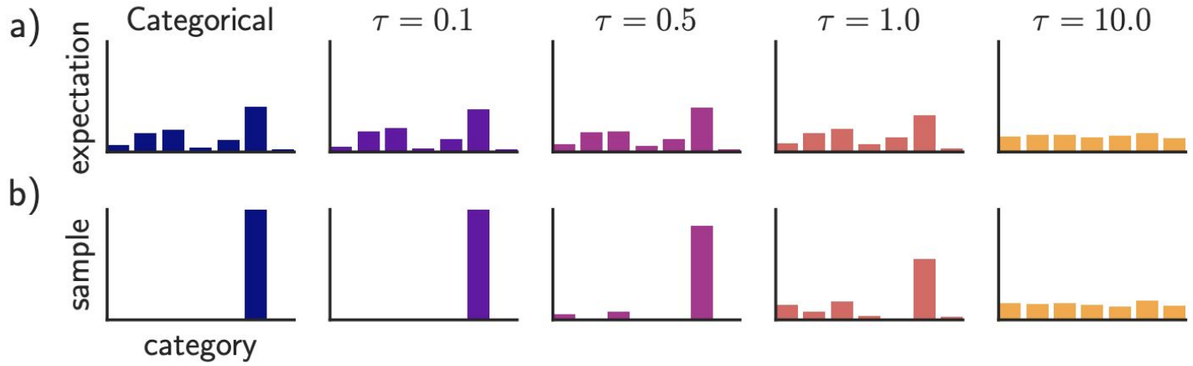


Reparameterization Trick

- Problem: Stochastic nodes in the computation graph ($z \sim p_\varphi(z|x)$) prevent gradient flow during backpropagation.
- Solution: Reparameterization moves randomness outside the gradient chain by expressing $z = g(\phi, x, \varepsilon)$ with $\varepsilon \sim N(0,1)$, allowing gradients to flow through deterministic paths.
- VAE Application: This trick enables Variational Autoencoders to be trained with standard backpropagation while maintaining stochastic sampling.



Gumbel Softmax



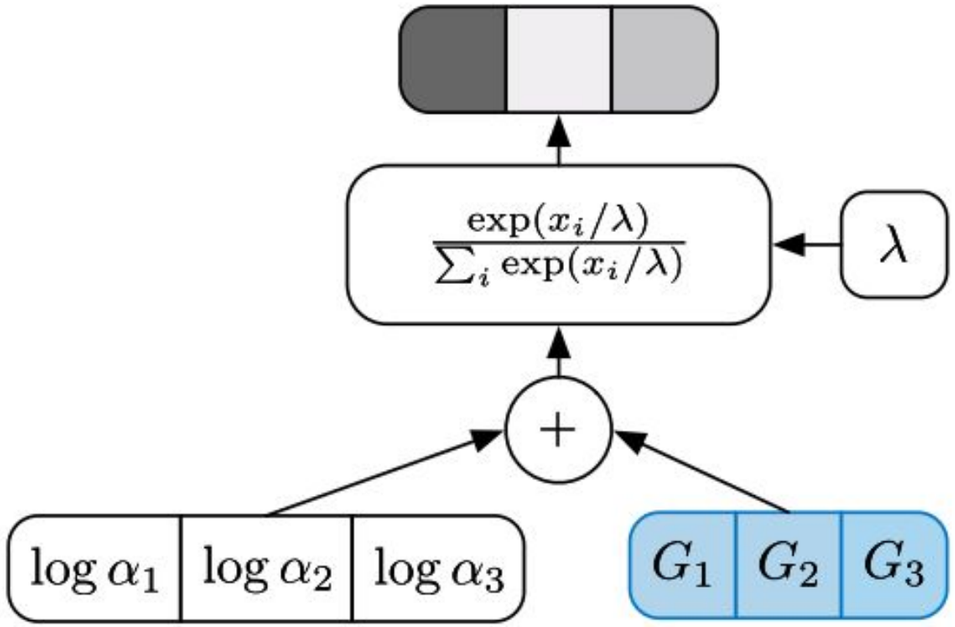
Reparameterization for Discrete Variables:

- Gumbel-Softmax applies the reparameterization trick to categorical distributions, allowing gradient flow through otherwise non-differentiable discrete sampling operations.

The τ parameter balances between accurate sampling (low τ) and gradient quality (high τ)

Continuous Relaxation:

- By replacing discrete sampling with a differentiable softmax approximation, Gumbel-Softmax enables end-to-end training while maintaining the ability to sample discrete outcomes during inference.



Training Results

Difference in Parameters True vs Simulation:

Energy: -4.14 MeV, Percent: -0.59 %

Position: 0.015, difference/pmt_radius: 0.379

Direction: 0.0032, Angular: 0.186 degrees

Event Parameters:

Energy: 700.90 MeV

Initial Position: (-0.46, -0.20, 0.55)

Initial Direction: (0.48, 0.20, -0.86)