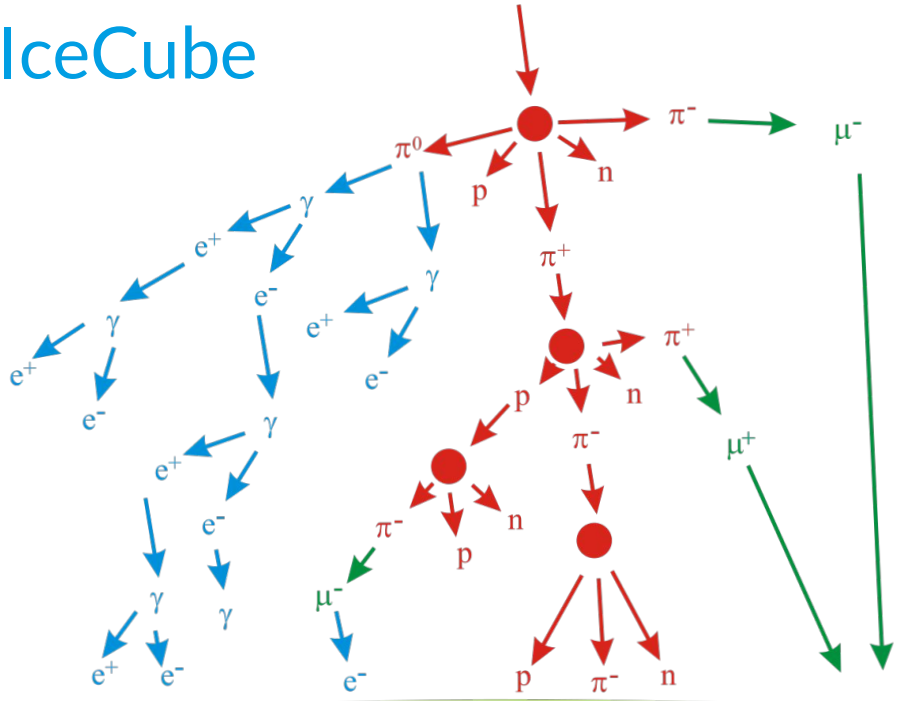
Towards Reducing the CPU Time for Air-Shower Simulations at IceCube

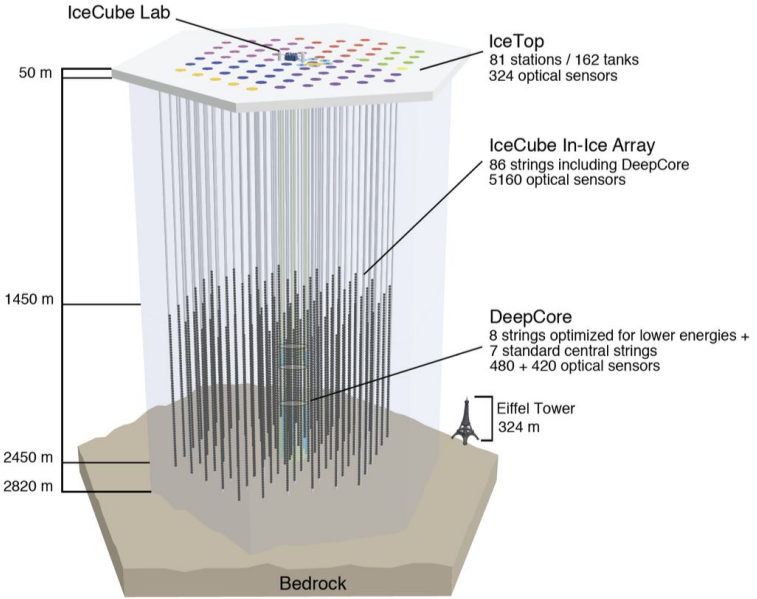
Navid K. Rad, Jakob van Santen

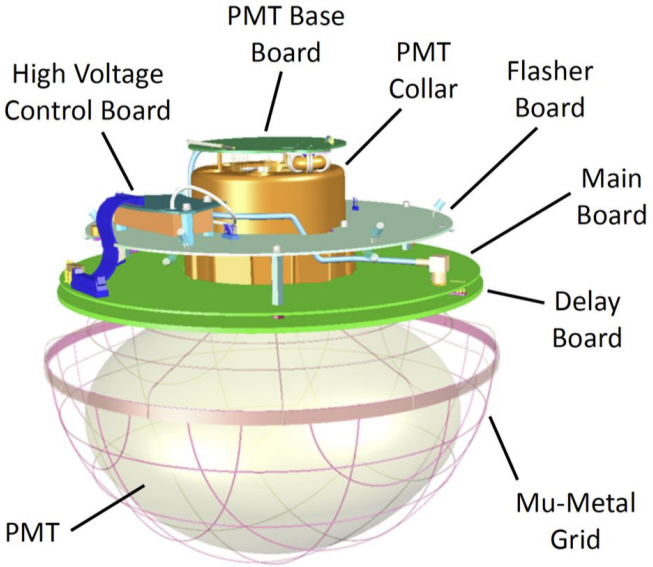
First Annual KISS meeting
Feb 13. 2024, Hamburg



IceCube Neutrino Observatory

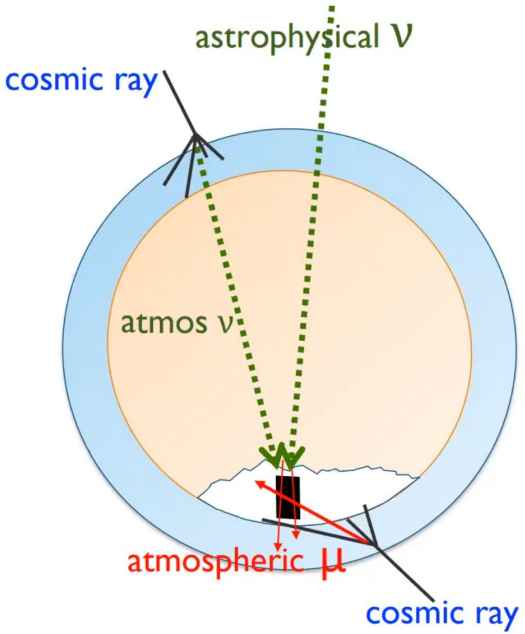
- Cubic-kilometer Neutrino detector
- **5160** Digital Optical Modules (DOM) in the glacial Antarctic ice **(depths of 1450-2450m)**
- Each DOM: a 25 cm PMT, high voltage power supply and digi & com. electronics





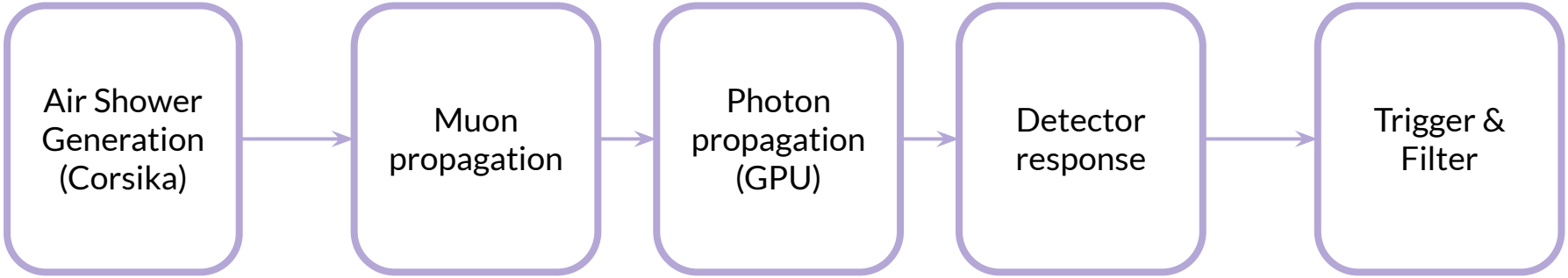
Background for high energy astrophysics neutrinos

- Atmospheric Muons and neutrinos:
  - Produced from the interaction of cosmic rays with the atmosphere
  - Even @ 1.5 km below ice, detected at high rates!
    - atmospheric **muons:** **~1000/s ~ O(10^3) Hz**
    - atmospheric **neutrinos: ~ 1/5min ~ O(10^{-3}) Hz**
    - **astrophysical neutrinos ~ 1/month ~O(10^{-6}) Hz**
  - Need to reduce background by factors 10^3-10^9
- The challenge is the rare background events:
  - Muon + few or no other low energy muons
  - a lone atmospheric neutrino

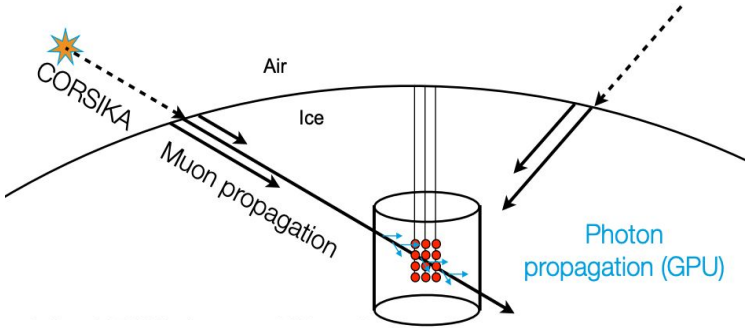


⇒ Very large simulations are needed!

IceCube simulation chain:

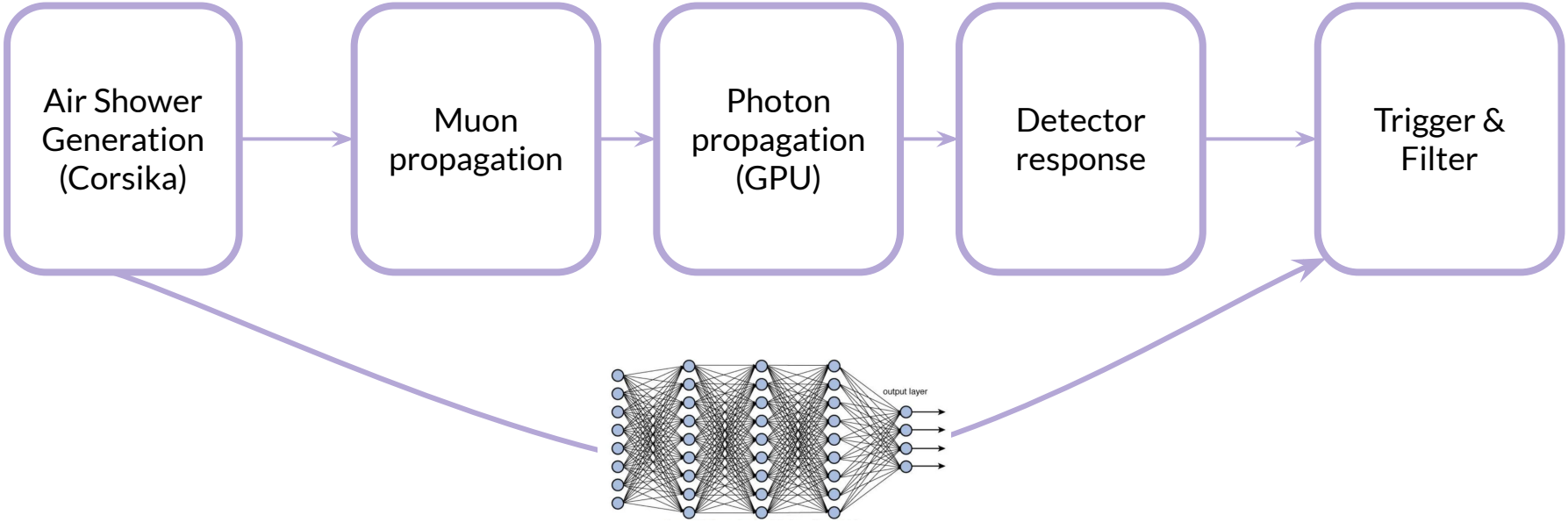


- CORSIKA simulates Air Shower:
  - Interaction of the primary cosmic rays with atmospheric nuclei
  - EM and Hadronic showers, $\pi^{\pm}$, π^{0}, K, **μ, ν** are produced and propagated to the ice surface
- Muons are propagated through the ice:
  - account for the stochastic energy losses
- Photon propagation:
  - ice properties depend on depth ⇒ photons need to be tracked individually…
  - **The most computationally intensive part of simulation (but parallelizable)**



⇒ Only 2% of simulated events pass the Trigger & Filter

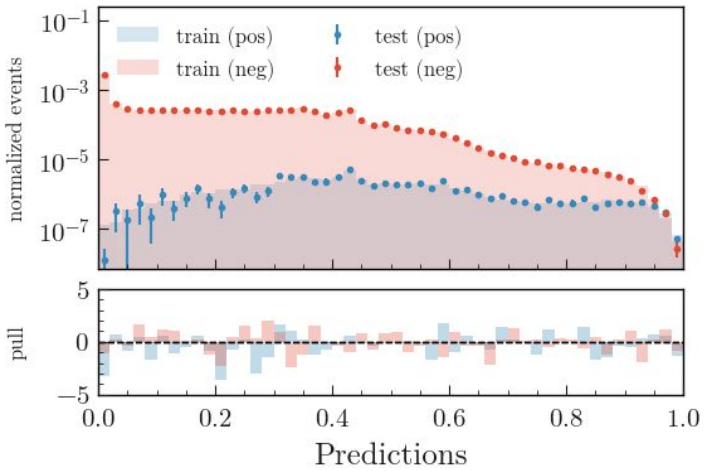
Taking a shortcut?

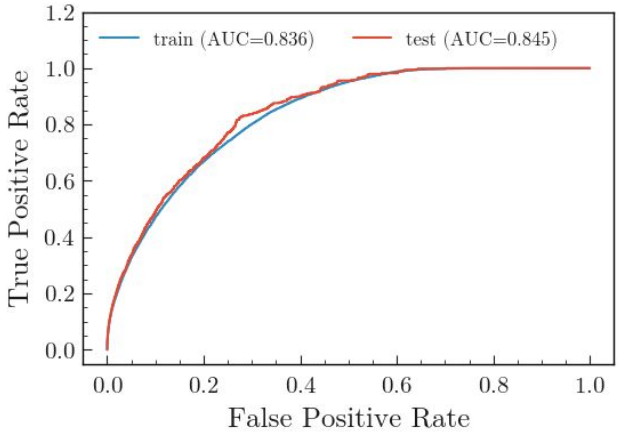


Use Machine Learning to try to predict probability that a certain air shower will pass the Trigger/Filter

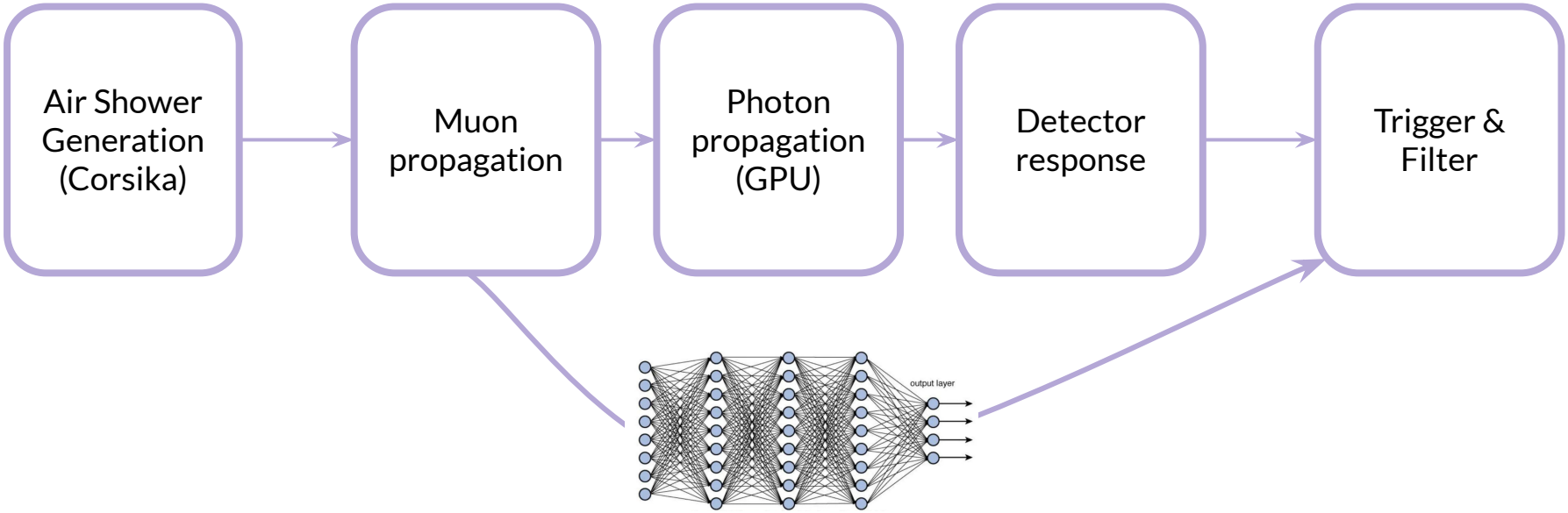
First attempts:

- Using available information of the **primary particle only**:
  - energy
  - particle type (H, He, Fe…)
  - directional information
  - interaction height
- Train and hyper tune a NN on the available simulations (balanced sample)





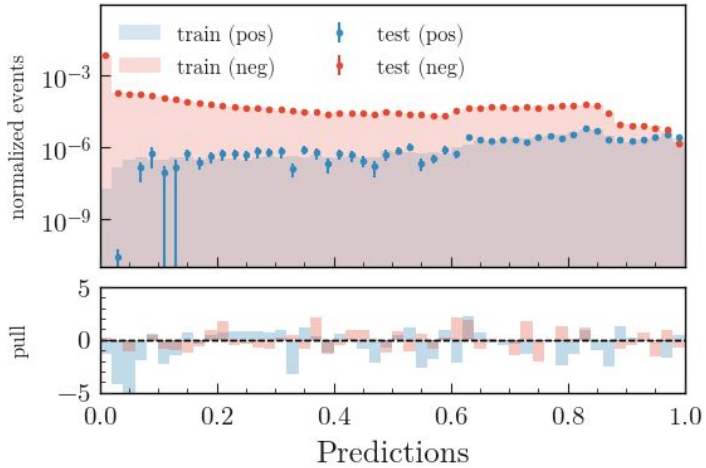
⇒ Even after hyper tuning, difficult to gain significant discrimination

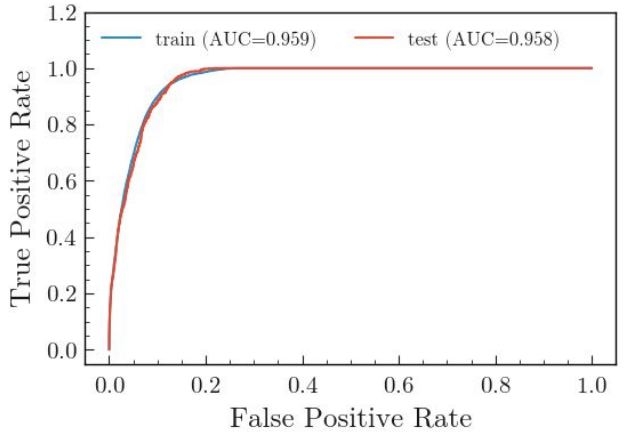


Use Machine Learning to try to predict probability that a certain air shower will pass the Trigger/Filter

Muons come to the rescue?

- Use muon/muon bundle information as well:
  - muon multiplicity,
  - muon bundle energy, fraction etc
  - distance of muons from the “shower”
- Train and hyper tune a NN on the available simulations (balanced sample)





⇒ Muon information significantly improves performance

Potential gain in CPU time

- Assumptions:
  - air shower generation time << simulation time
  - evaluation time of the model << simulation time
  - CPU time goes as $N_{accepted}$
- Method:

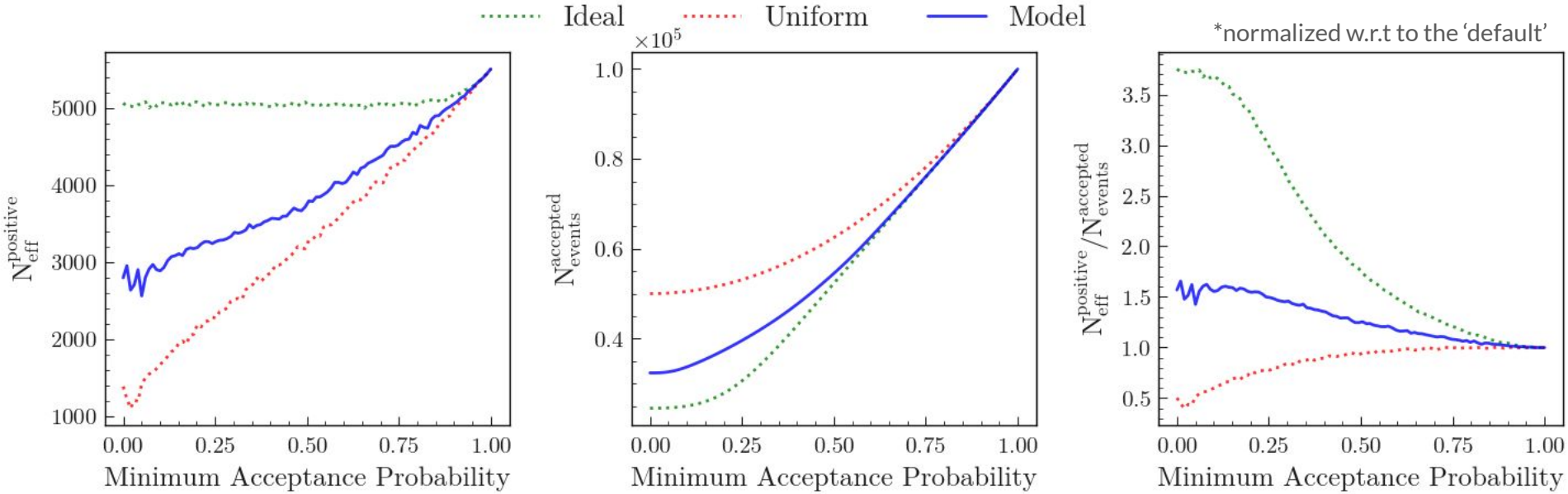
1. Interpret the predicted score ($\mathbf{p}_i$) as "acceptance probability" of the event
2. Assign corresponding weight to each event as $w_i = 1/p_i$
3. Choose (scan) the minimum acceptance probability threshold

- Simple "speed up" Metric: $N_{\mathrm{eff}}^{\mathrm{positive}} / N_{\mathrm{accepted}}$
  - $N_{accepted}$: number of accepted events (based on their p_i)
  - N_{eff}: effective sample size of the accepted positive events
    (size of an unweighted sample that would the same relative uncertainty)

$$N_{eff} = \frac{(\sum_i w_i)^2}{\sum_i w_i^2}$$

Potential gain in CPU time

- Construct
  - **Ideal:** "prediction" based on truth information → Best case
  - **Uniform:** "prediction" based on uniform distribution → Worst case



⇒ Improvement of about 1.5x compare to default scenario

Summary and Outlook

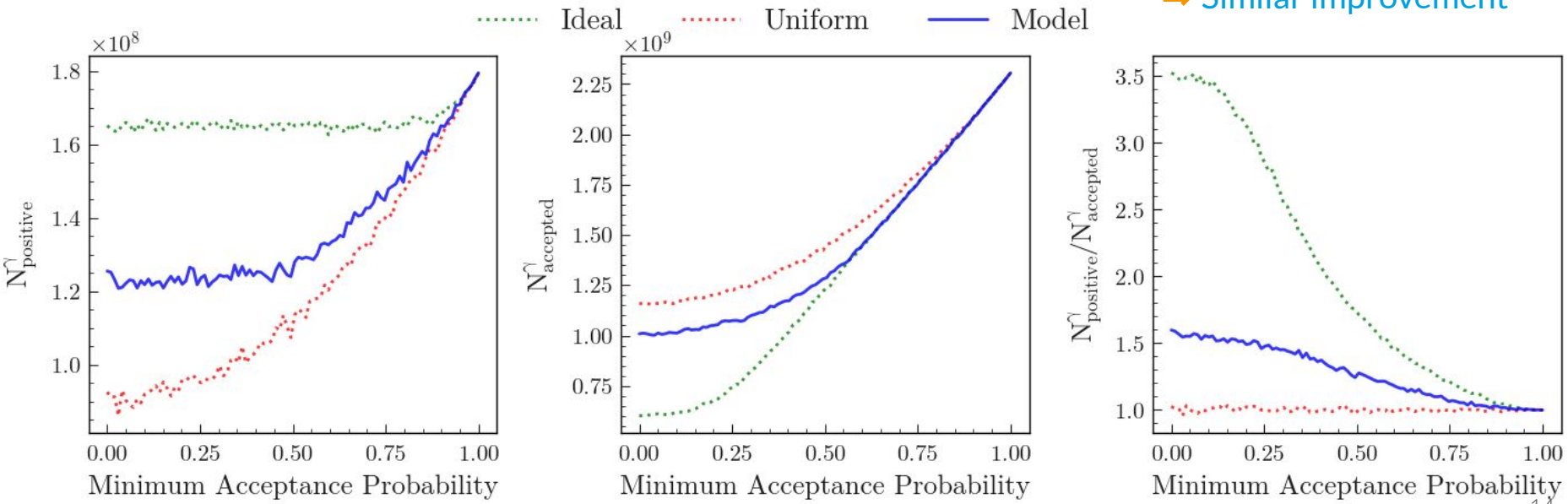
- First attempts show some improvement in CPU time:
  - Proof of concept that acceleration is possible!
- Still room to improve the model
  - Larger samples still available for training:
    - deepen the network and train longer
    - explore different architecture
  - Feature engineering
  - Go to the next level of selection (L3)
- Dedicated Loss:
  - incorporate the speed up improvement in the loss function
- Active learning
  - Use the model to “suggest” air showers and use the simulation as the teacher

Backup

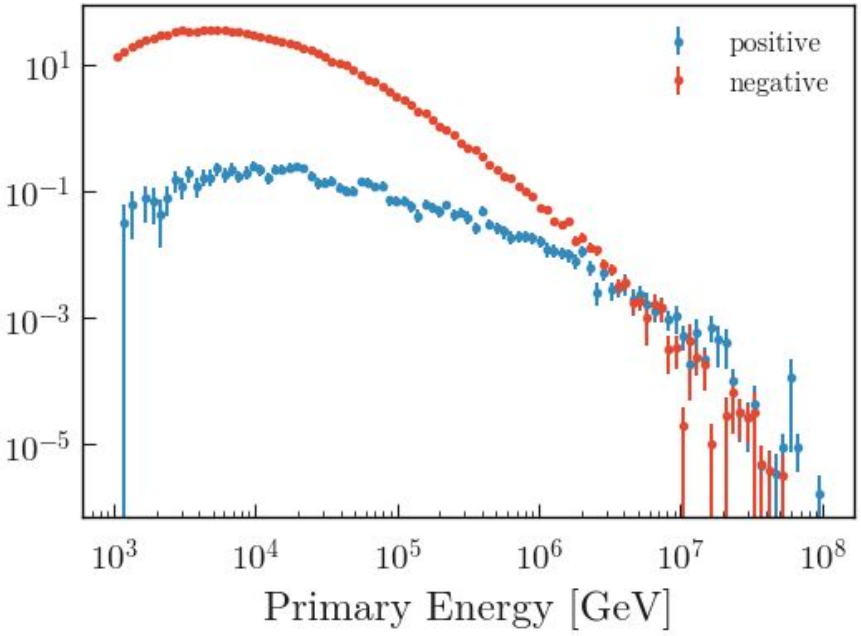
Potential gain in CPU time (slightly more realistic)

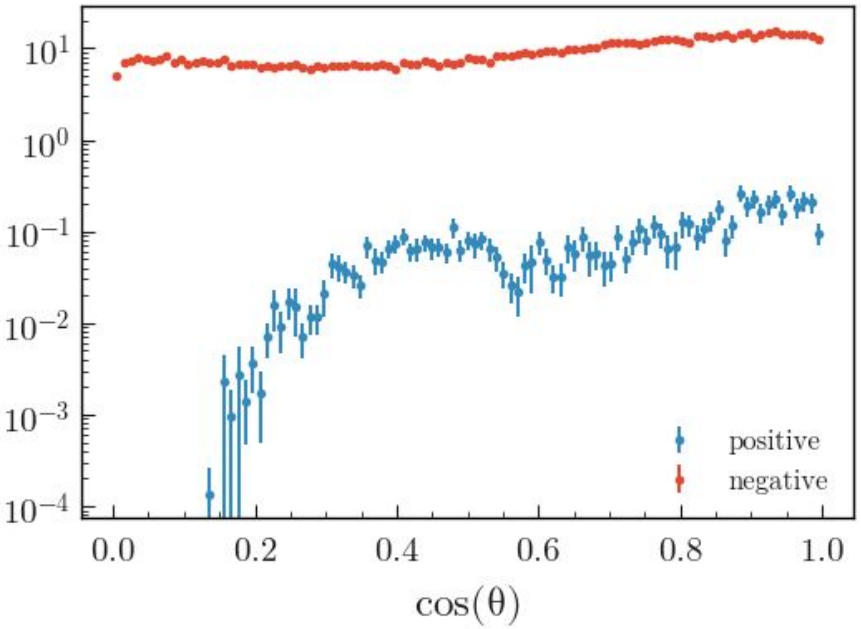
- Modify Assumptions:
  - CPU time ~~goes as $N_{accepted}$~~ goes as $N(\gamma)$
- Slight more realistic "speed up" Metric: $N_{\text{positive}}(\gamma)/N_{\text{accepted}}(\gamma)$
  - $N_{accepted}(\gamma)$: total number of photons in the accepted events
  - $N_{positive}(\gamma)$: total number of photons in the accepted positive events

⇒ Similar improvement



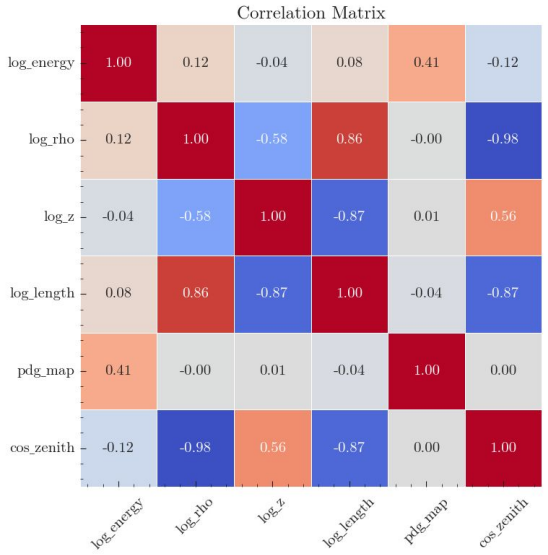
Primary Distributions



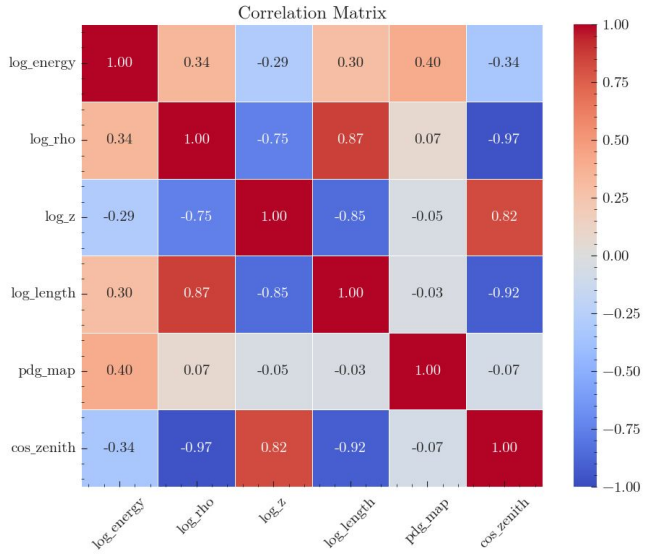


Primary Correlations

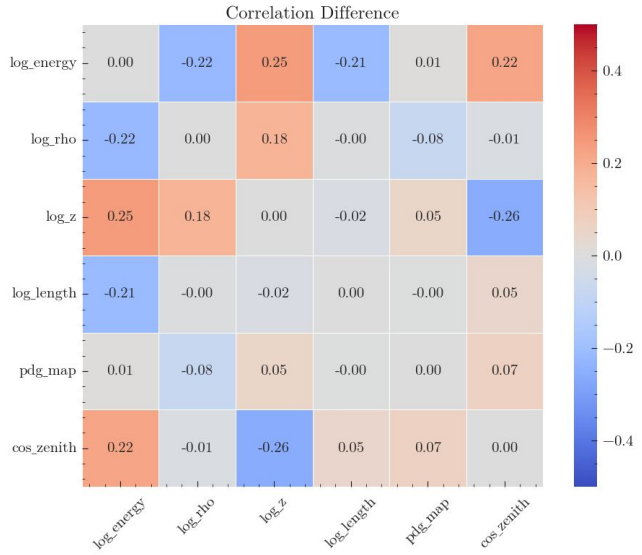
Negatives



Positives



Difference



Model: "Primary Hypertuned"

Layer	Size	Activation	Param #
batch_normalization	8		32
Dense_0	224	leaky_relu	2016
dropout	224		0
Dense_1	224	leaky_relu	50400
Dense_2	224	leaky_relu	50400
Dense_3	112	leaky_relu	25200
Dense_4	112	leaky_relu	12656
Dense_5	112	leaky_relu	12656
Dense_6	56	leaky_relu	6328
Dense_7	56	leaky_relu	3192
Dense_8	56	leaky_relu	3192
Dense_9	28	leaky_relu	1596
Dense_10	28	leaky_relu	812
Dense_11	28	leaky_relu	812
Dense_12	14	leaky_relu	406
Dense_13	14	leaky_relu	210
Dense_14	14	leaky_relu	210
Dense_15	8	leaky_relu	120
dense	1		9

Total params: 170,247
Trainable params: 170,231
Non-trainable params: 16

IceCube simulation challenges:

- The challenge:
  - Majority of the simulated showers are not triggered and do not pass the initial filtering (~2%)
  - Lots of CPU+GPU time is wasted on showers that are thrown away, way before getting to the analysis level.
- Many attempted solutions:
  - Bias the generated distributions:
    - e.g. on average proton primaries end up with lower muon multiplicity
  - Parametrize the muon bundle properties
  - Hard energy cut: kill the shower if no particles about the energy threshold remain
  - Soft energy cut: rejection probability based on the expected number of muons above certain energy

⇒ Only work in specific cases, and only to some degree… need a more general solution!