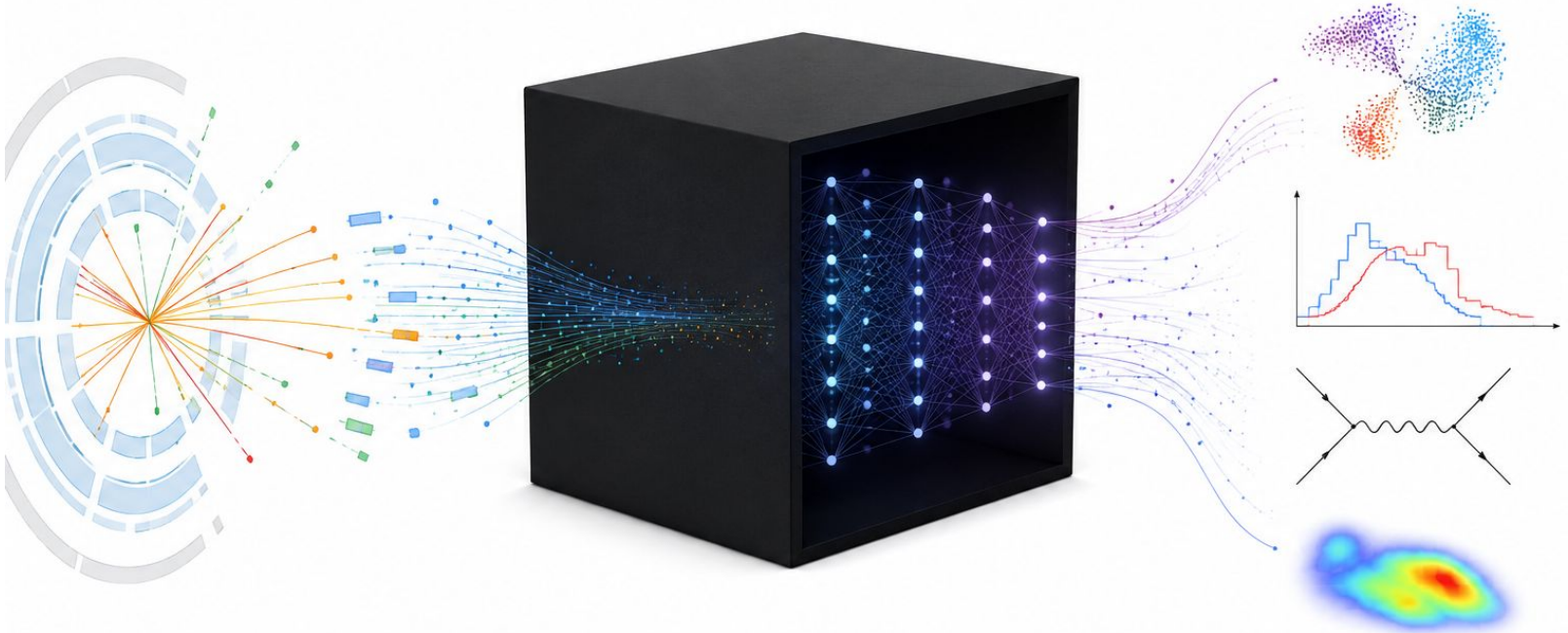
OPENING THE BLACK BOX



What does a Collider Physics DNN Actually Learn ?

Group 3: ML4HEP

Given Tasks

1. The Higgs ML dataset contains 17 primitive detector-level features and 13 engineered features derived by physicists specifically to separate $H \rightarrow \tau\tau$ from the background.
2. Goal is to first train a BDT to obtain a baseline ranking of feature importance.
3. Then do the same classification using DNN using complete set of input features.
4. Apply Integrated Gradient (IG) to interpret the trained data.
5. Answer the following question: does IG confirm what physics predicts? Or does IG reveal something a physicist couldn't easily see?
6. Retrain models using the top IG-ranked features and compare their performance with the full-feature model.

Interpretability Problem

A Neural Network may produce high AUC but high performance does not guarantee physically meaningful reasoning.

Interpretability helps test whether the model can be trusted and simplified.

Central Questions :

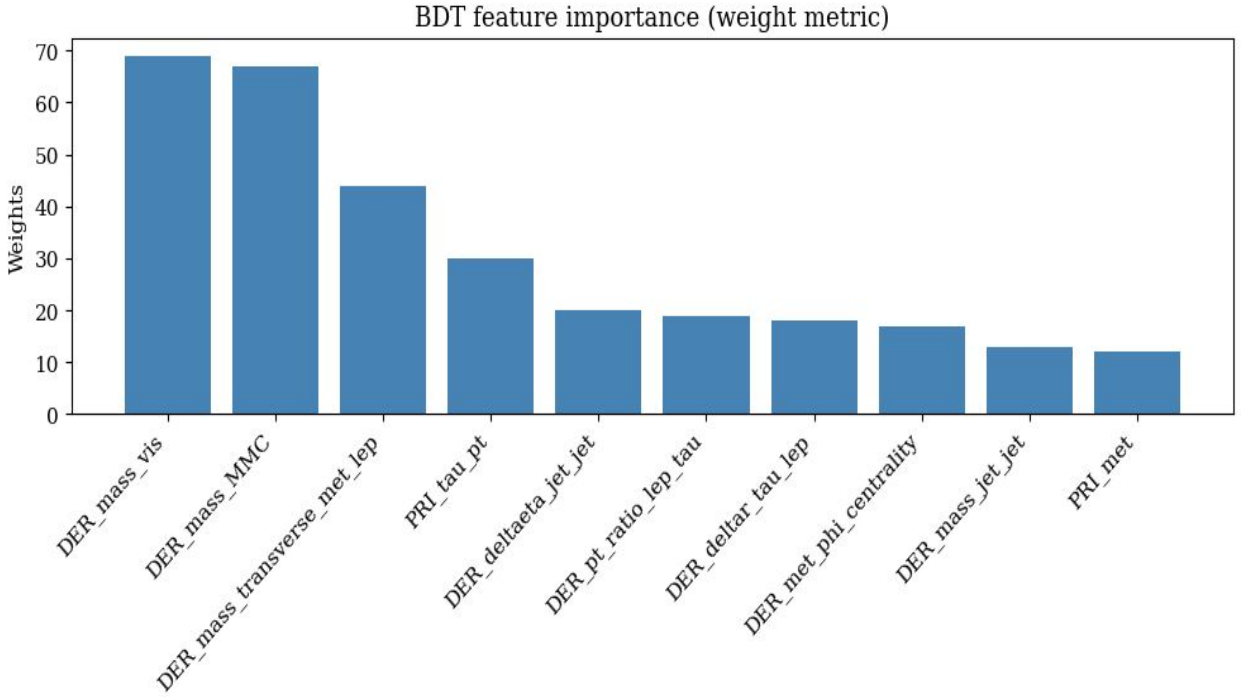
1. What features did the network learn to use?
2. Did it validate the physicists' engineered variables or bypass them?
3. Can the model be compressed without losing much performance?

Goal: Understand input-output behaviour of the deep network.

Boosted Decision Trees (XgBoost)

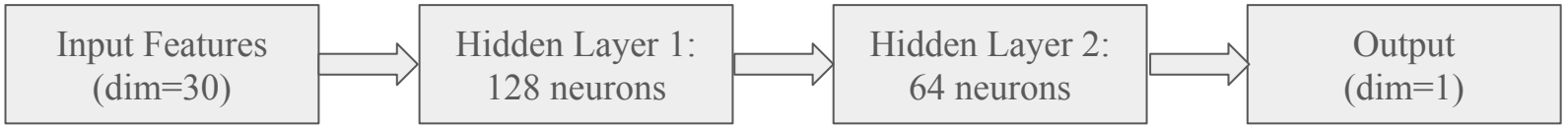
A Boosted Decision Tree combines many small decision trees. Each new tree focuses on correcting the classification errors made by the previous trees.

- Trained using all 30 primitive and engineered features.
- 50 shallow trees with maximum depth 3
- Learning rate: 0.01
- Event and feature subsampling: 50%



Deep Neural Network (DNN):

Constructed a neural network with 2 hidden layers.



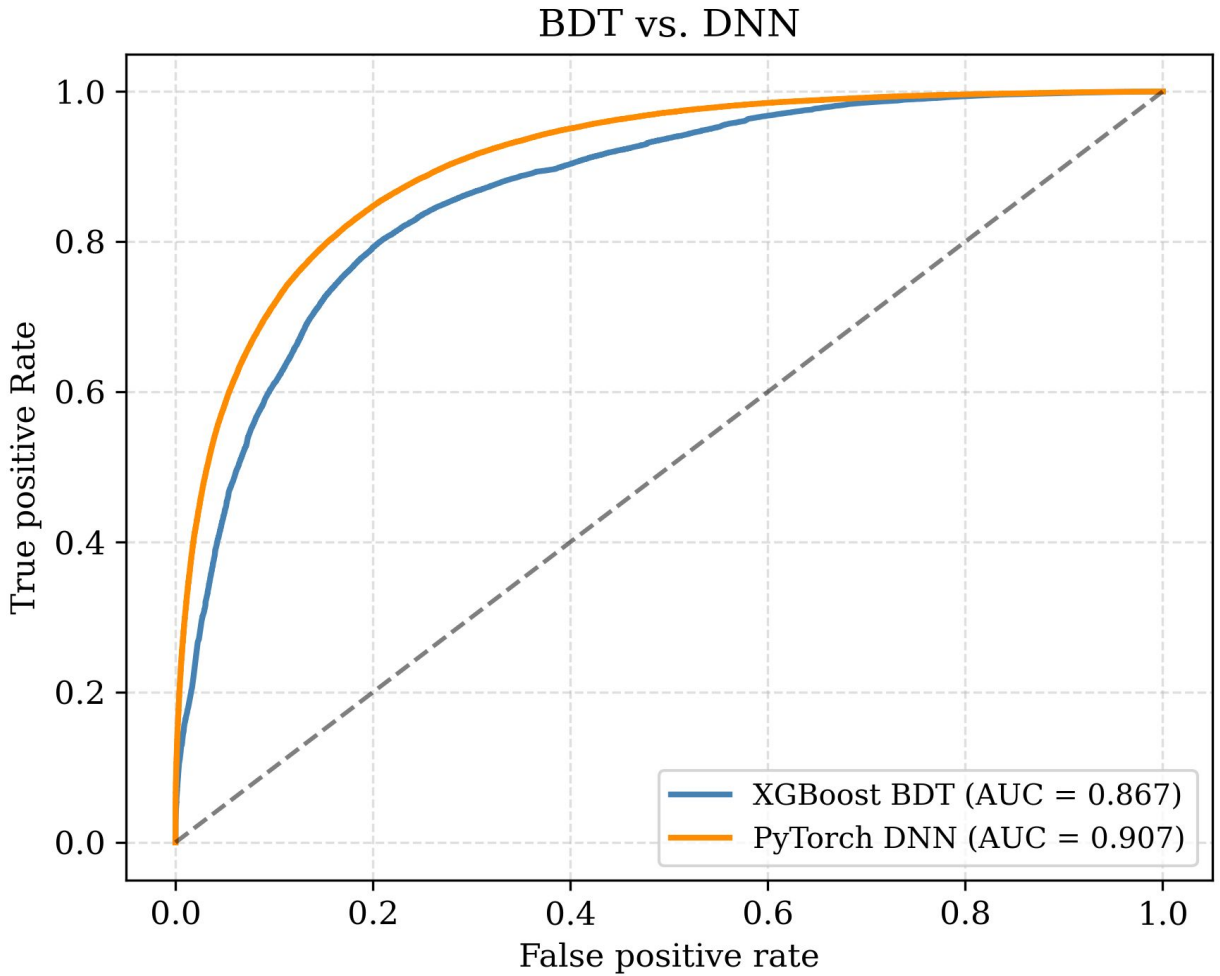
- Activation function used: **SiLu**
- Training-Test split used: **70: 30**
- AdamW optimizer
- Loss function : **weighted binary cross entropy**
- Dropout : 0.2
- Training Set: 572766 events
- Testing Set: 245472 events
- Number of epochs : **30**
- Batch size : 256

BDT vs DNN

Performance comparison:

1. BDT AUC = 0.867
2. DNN AUC = 0.907

We get similar efficiencies, but in this setup, the DNN performs slightly better.



Why Simple Attribution Methods Fail ?

Why not use weights? (Yes ! But interpretable only for Linear Networks)

- For a linear model, $F(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b,$

  so larger weight generally indicate more important features.

- A deep neural network is composed of multiple layers and nonlinear activation functions:

  $$F(\mathbf{x}) = f_L(W_L\, f_{L-1}(W_{L-1} \cdots f_1(W_1\mathbf{x} + b_1)) + b_L)\,.$$

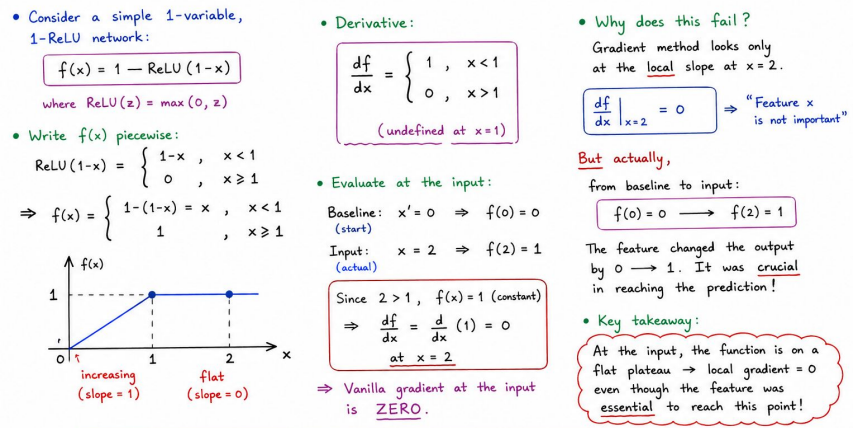
  There is no one-to-one correspondence between an input feature and a single weight. A large weight in one layer may have little effect on the final prediction if another neuron suppresses its contribution.

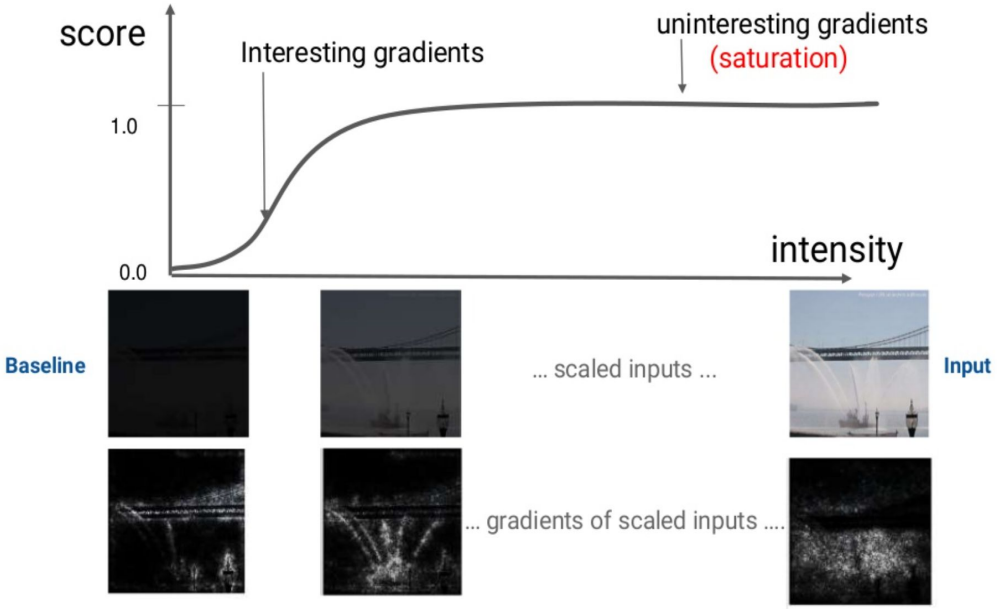
- Weights represent learned connections, **not feature importance**.

Why Simple Attribution Methods Fail ?

Why not use gradients?

- Gradients measure only **local sensitivity**.
- When the network output has reached a plateau, the gradient becomes nearly zero even if the feature was crucial in reaching that prediction. Consequently, the feature may incorrectly appear to have no importance.





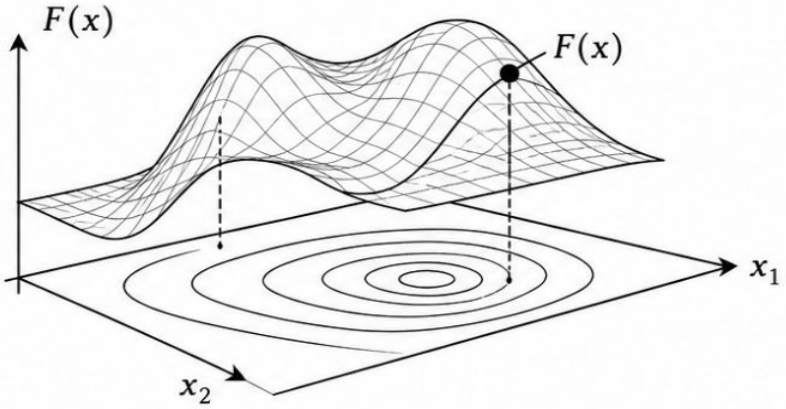
Integrated Gradients (IG) – 2D input space

Function

$F : \mathrm{R}^2 \rightarrow \mathrm{R}$

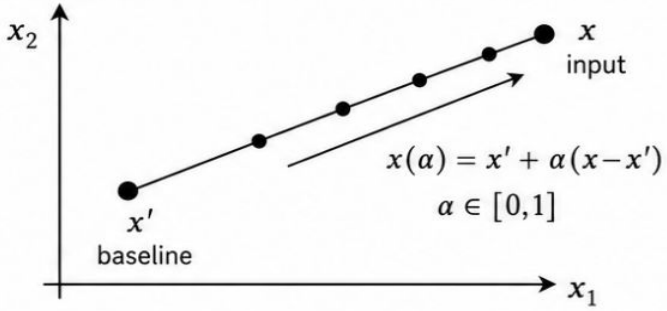
$F(x_1, x_2)$ = prediction (score)

over 2D input



- x' : baseline (reference input)
- x : input to explain
- $x(\alpha)$: point on the straight path, $\alpha \in [0,1]$
- $\frac{\partial F(x(\alpha))}{\partial x_i}$: gradient (how F changes) at $x(\alpha)$
- $\mathrm{IG}_i(x)$: attribution for feature i

Straight path from baseline to input



Integrated Gradients

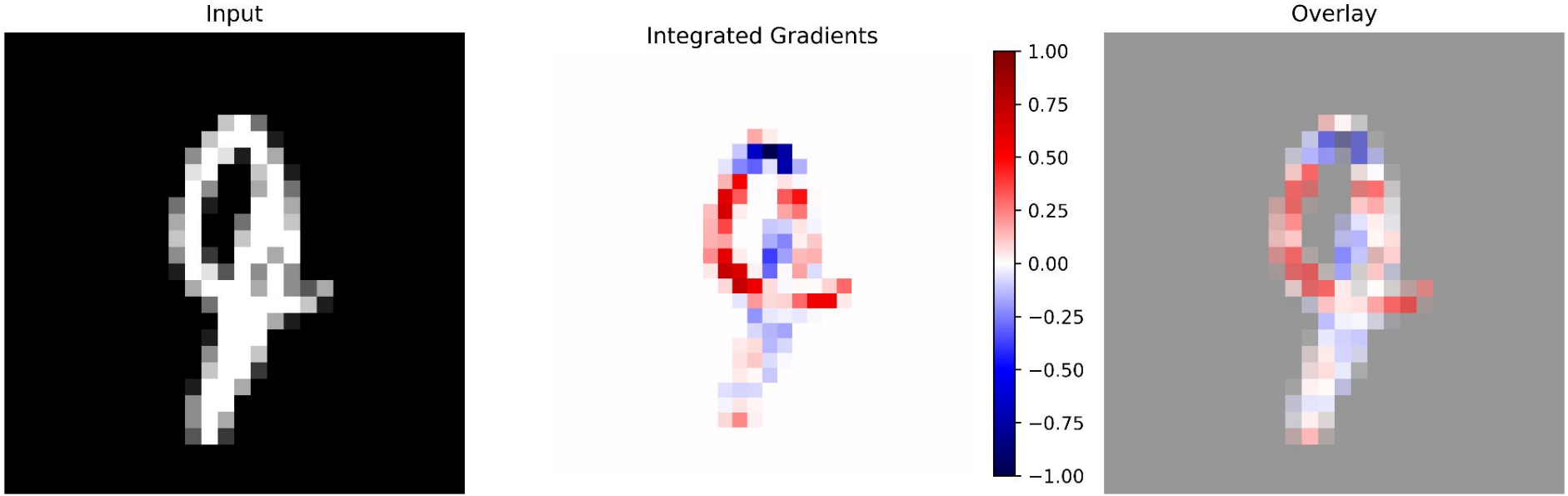
$$\mathrm{IG}_i(x) = (x_i - x_i') \int_0^1 \frac{\partial F(x(\alpha))}{\partial x_i} \, d\alpha$$

↑ average the gradient along the straight path

Intuition

Gradient at a single point can be noisy or zero. IG averages gradients along the path from baseline to input → more stable and satisfies desirable properties (e.g., sensitivity).

IG on MNIST



True: 9

Prediction: 4

Preliminaries: IG Baselines

Question: How to choose baseline ?

- The baseline x′ defines the *question* IG answers — "what makes x different from x′?" — so it must be chosen physically, not by convention.
- A good baseline should carry **no information about the class you're explaining** and should **lie on the data manifold**.

- Examples of baselines:

  Blank baseline, zero-vector (suitable for image classification tasks)

  Background-derived baseline — averaging over many real background events

  Weighted baseline- Weighted average of the bkg events with weights as the cross-section.

How is Integrated Gradients Estimated?

- **During training**
  - The goal is to improve the model by updating its weights.
  - Gradients are computed to determine how the weights should change to reduce the training loss.
- **For Integrated Gradients**
  - The model is already trained, so its weights remain fixed.
  - Instead of asking *"How should the weights change?"*, we ask *"How does the prediction change if the input image changes?"*
  - Therefore, the input image is marked to allow gradient computation (x.requires_grad = True).
  - Backpropagation is then performed from this prediction score to the input image.
  - The resulting gradients indicate how sensitive the prediction is to each input pixel.
- We use Captom, open source library for model interpretability built on PyTorch.

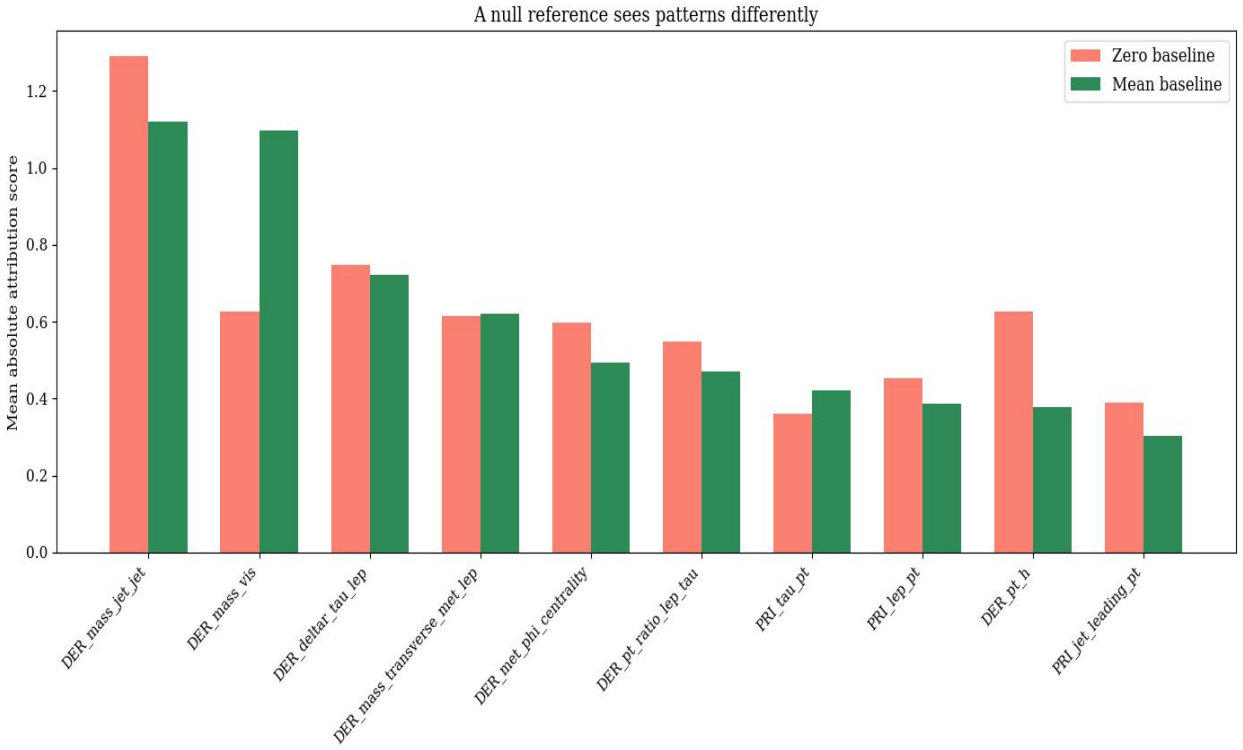
  Animation

Integrated Gradients: Comparing Two Baselines

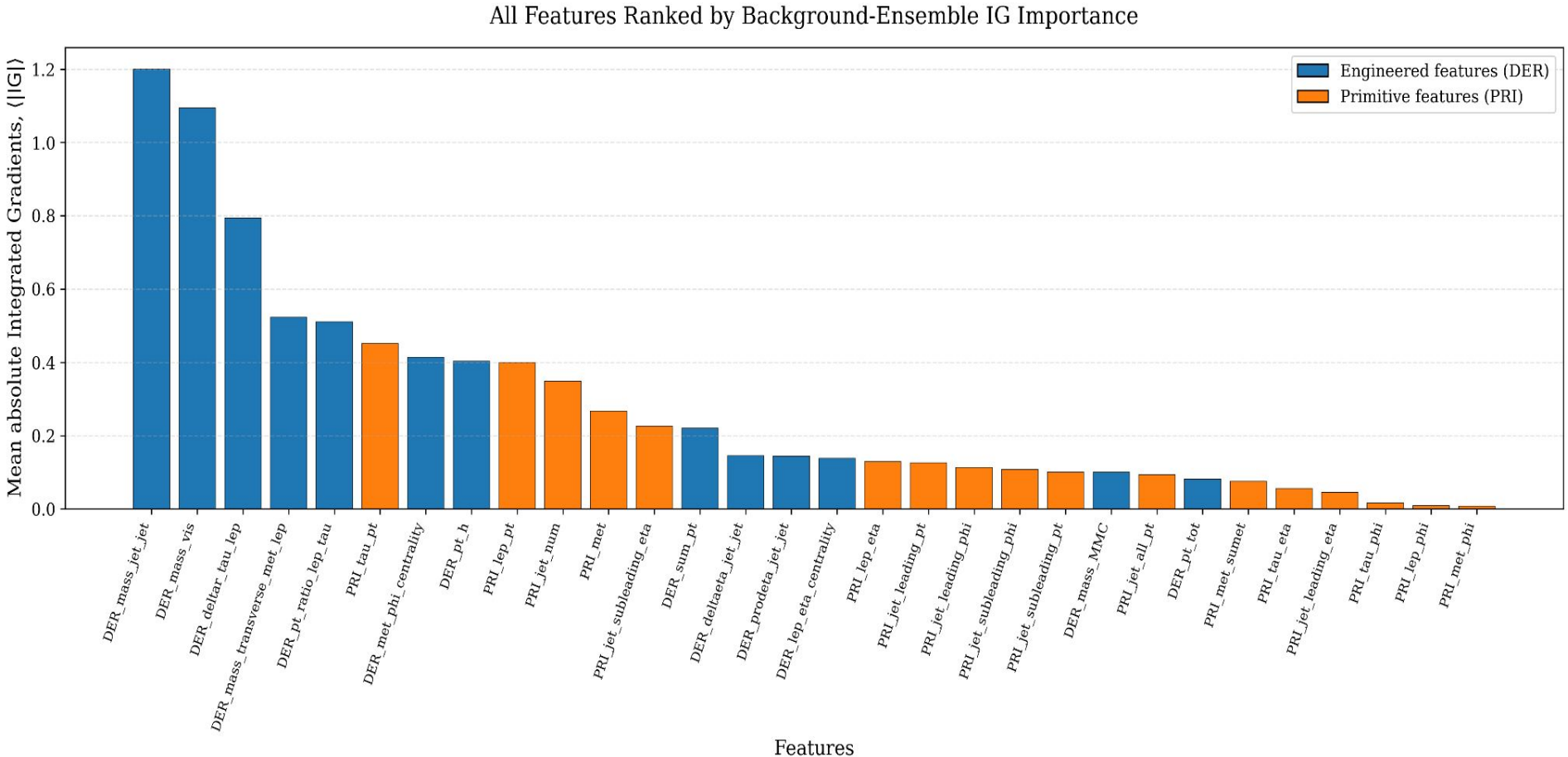
Applied to the trained full 30 featured DNN

Evaluated on 2000 signal events

- Both baselines select similar key features.
- Exact ranking depends on the baseline.
- Seven of the top ten variables are engineered DER_ features.
- The background ensemble is preferred for physics interpretation.

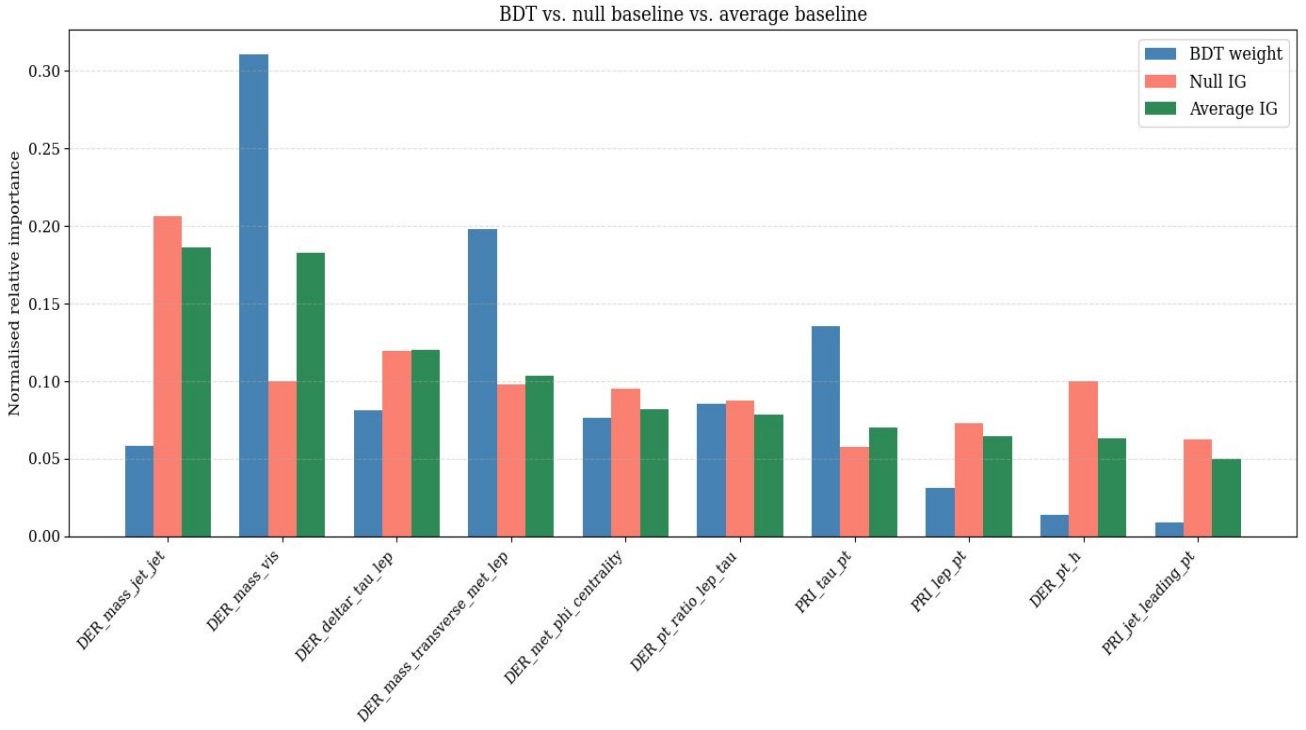


Feature Importance from Background-Ensemble IG



BDT vs Null IG and Average IG

- The ranking is not identical, but the dominant features are broadly consistent.
- Differences in ranking shows that IG depends on the choice of the baseline.

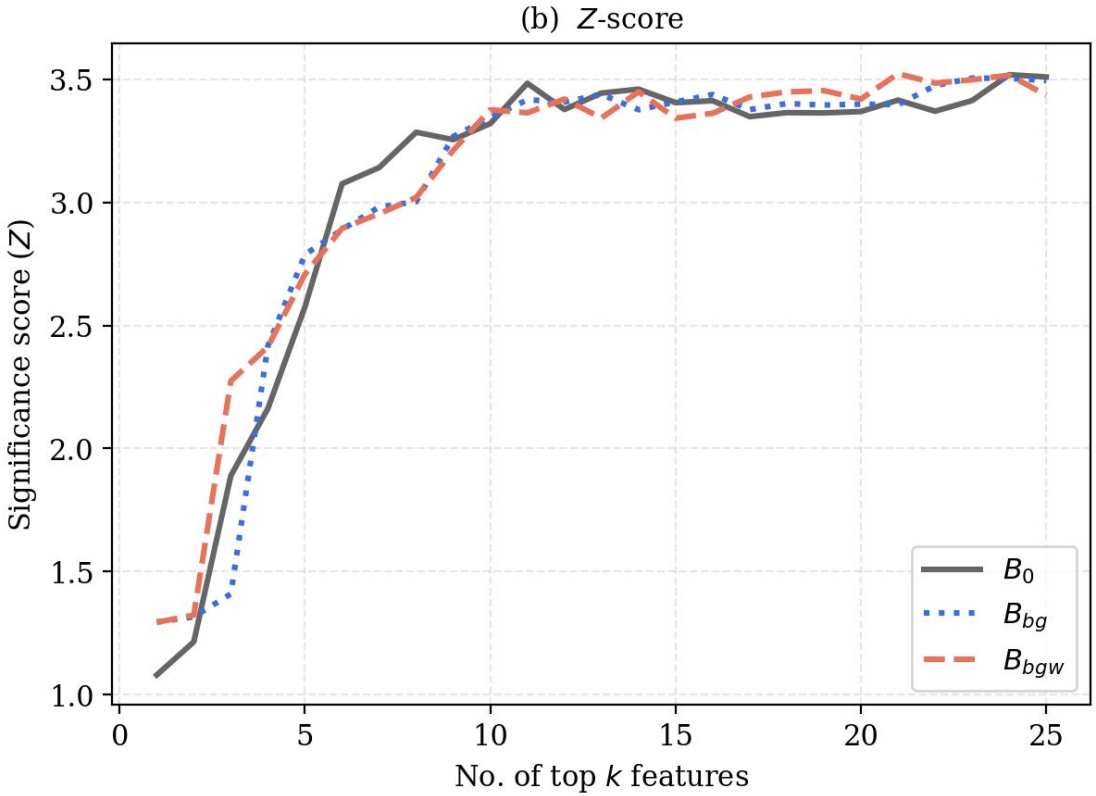


Significance Z

If we keep only the top k features ranked by Integrated Gradients, how much signal-discovery power do we retain?

Significance Z grows as top-k features are added and plateaus by k ≈ 10 at Z ≈ 3.5.

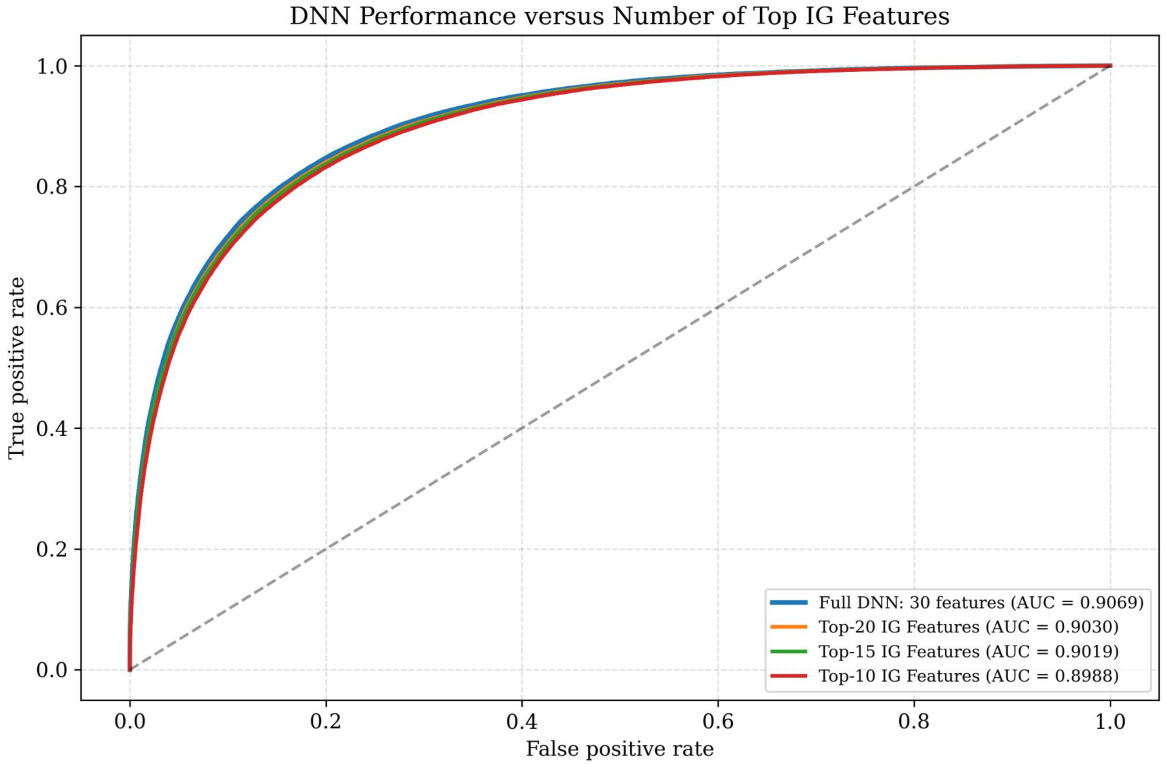
The first handful of IG-ranked features carry almost all the discriminating power; the three baselines converge to the same reach.



Stress testing

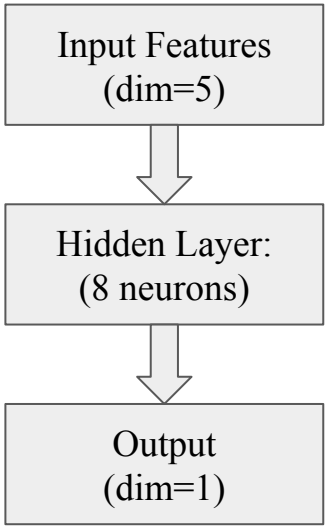
- Can we get a similar performance keeping the same DNN architecture but with less number of features?
- We trained our model using k most important features (k=10,15,20)

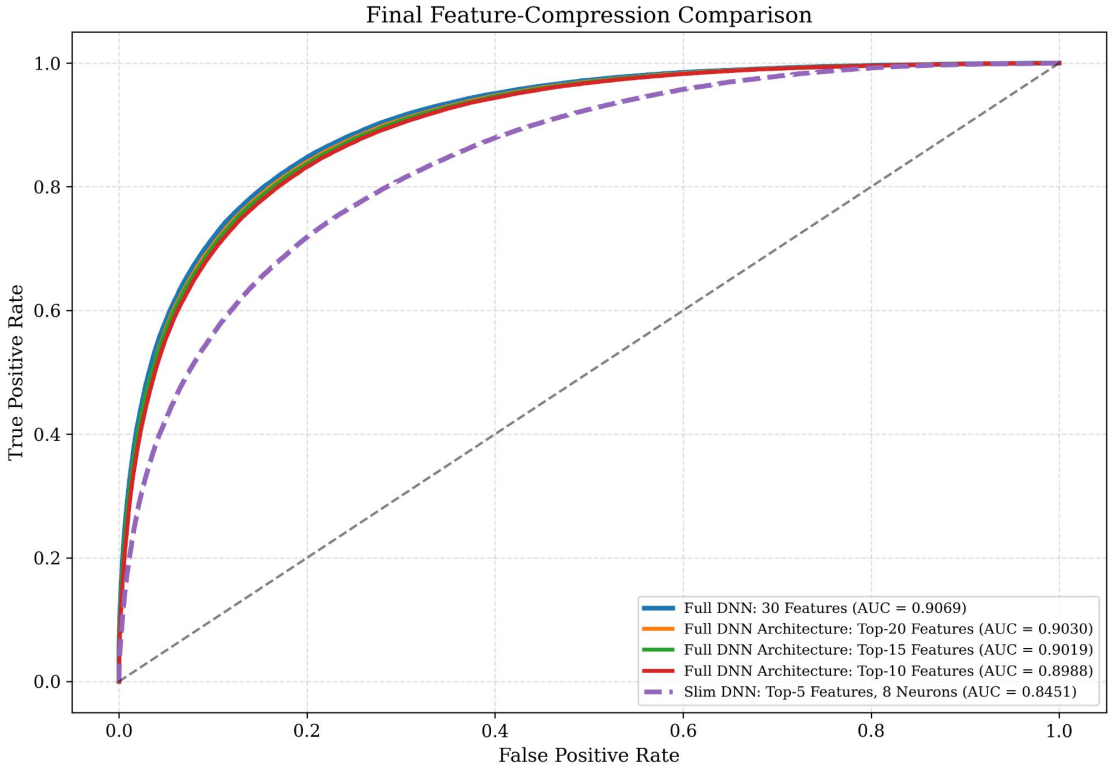
MODEL	AUC
Full DNN	0.9069
Top 20	0.9030
Top 15	0.9019
Top 10	0.8988



A Slim Model : testing feature compression

Can we further reduce the parameters and get a similar performance?





Slim Model AUC: 0.8451

It indicates we could do well with less compute if we understand the physics connection.

Conclusion:

- Performed the classification task of the Higgs ML dataset using BDT and DNN.
- Using Integrated Gradients, we ranked the features and trained the DNN using k topmost ones and found the best features learnt by DNN are engineered, meaning the model actually learnt physics based variables.
- We did something simple but non-trivial — we went from "the DNN works" to "we know what it learned, we can compress it, and we can measure the physics consequence of that compression in terms of signal significance."
- Do we understand it completely ?

Not addressed: interactions between features or the complete internal decision-making process of the network.

Integrated Gradients is a practical tool for opening the "black box" of deep neural networks, helping us understand *which features influence a prediction*, while also reminding us that attribution is not the same as understanding the full physics learned by the model.


THANKYOU

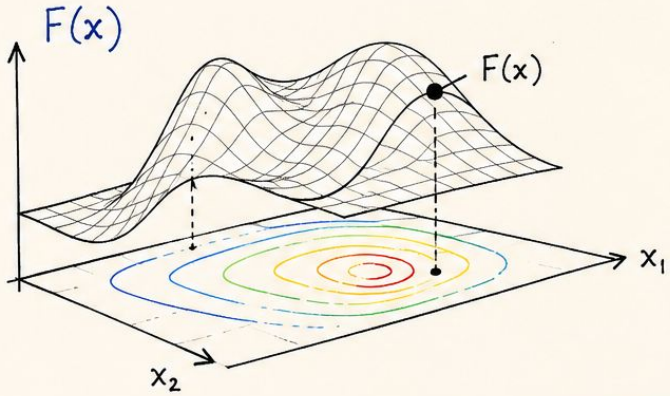
Backup

Integrated Gradients (IG) – 2D input space

Function

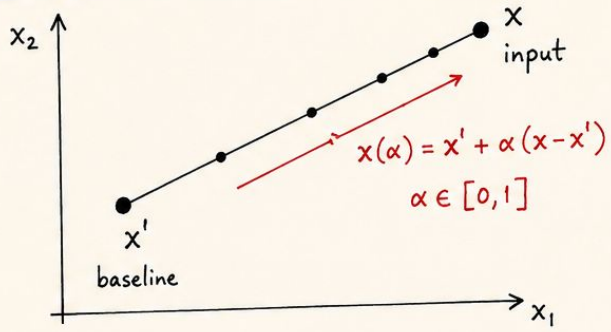
$F : \mathbb{R}^2 \to \mathbb{R}$

$F(x_1, x_2)$ = prediction (score) over 2D input



- x' : baseline (reference input)
- x : input to explain
- $x(\alpha)$: point on the straight path, $\alpha \in [0,1]$
- $\frac{\partial F(x(\alpha))}{\partial x_i}$: gradient (how F changes) at $x(\alpha)$
- $IG_i(x)$: attribution for feature i

Straight path from baseline to input



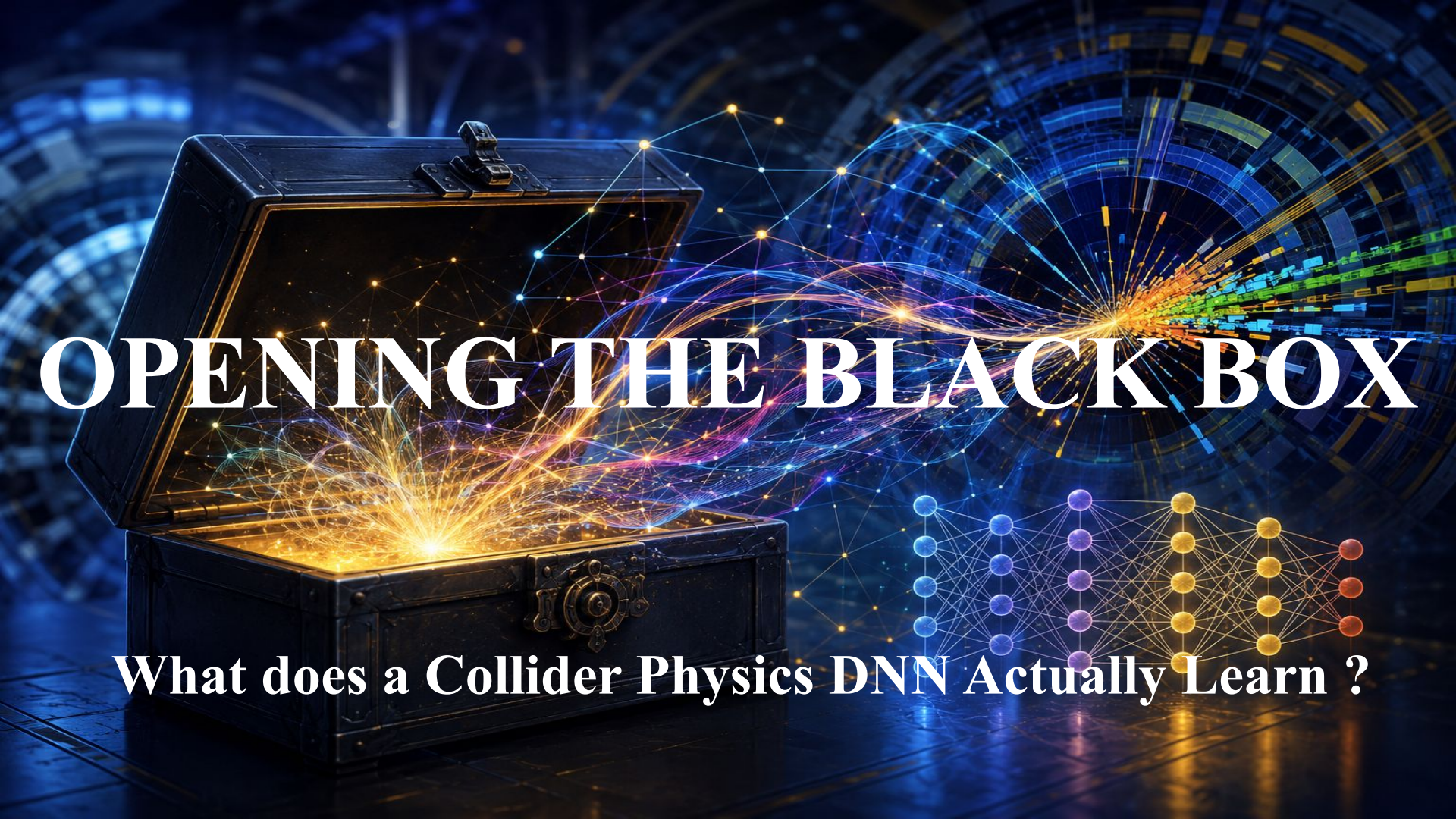
Integrated Gradients

$$IG_i(x) = (x_i - x_i') \int_0^1 \frac{\partial F(x(\alpha))}{\partial x_i} \, d\alpha$$

average the gradient along the straight path

Intuition

Gradient at a single point can be noisy or zero. IG averages gradients along the path from baseline to input → more stable and satisfies desirable properties (e.g., sensitivity).


OPENING THE BLACK BOX
What does a Collider Physics DNN Actually Learn ?

Can we estimate gradient in the previous (second last) layer?

Yes, but that solves a different problem.

Short answer: You can compute gradients w.r.t. the second-last layer, and saturation is often less severe there, but that no longer tells you which input features are important.

① Why does input gradient fail?

Suppose your network is

$$x \longrightarrow h_1 \longrightarrow h_2 \longrightarrow y = F(x)$$

(x: 30 physics variables; h_1, h_2: hidden layers; y: output score)

Normally we compute $\boxed{\frac{\partial F}{\partial x}}$

"If I slightly change an input feature, how will the prediction change?"

- The problem is that if the output neuron is saturated, $\boxed{\frac{\partial F}{\partial h_2} \approx 0}$
- By the chain rule,

$$\frac{\partial F}{\partial x} = \underbrace{\frac{\partial F}{\partial h_2}}_{\approx 0} \cdot \frac{\partial h_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial x}$$

$\Rightarrow$ everything ≈ 0

So input gradients vanish!

② What if we compute gradients at the second-last layer?

Compute $\boxed{\frac{\partial F}{\partial h_2}}$ or $\boxed{\frac{\partial F}{\partial h_1}}$

This is valid. People actually do this.

Examples:

- Grad-CAM
- Concept Activation Vectors
- Representation analysis

Now you are asking "Which hidden neurons are important?"

Does it solve saturation? Sometimes.

The hidden neurons may lie in regions where gradients are non-zero, so yes, the gradient may be much larger than at the input.

But there's a catch.

Suppose neuron 57 is important... You find $\boxed{\frac{\partial F}{\partial h_{57}} = 2.3}$ Great. Now what?

You still don't know whether

- DER_mass_MMC
- PRI_tau_pt
- PRI_met

} caused neuron 57 to activate.

You've merely moved the explanation one layer earlier.

The mapping $x \to h_1 \to h_2$ is still highly nonlinear.

③ Why IG stays at the input

Physicists want answers like

"DER_pt_tot contributes 18%."

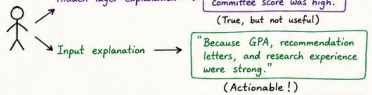
not

"Hidden neuron 83 contributes."

Input features have physical meaning. Hidden neurons do not.

④ Another viewpoint

Suppose you ask "Why was this student admitted?"



⑤ Then why not compute IG on the second-last layer?

You actually can.

Define the second-last layer activations as the "input" to the remaining network and compute Integrated Gradients there. Researchers sometimes do this to understand learned representations.

However, the attribution is then over latent features, not the original physics variables.

Takeaway:
Gradients in the second-last layer can be useful for understanding representations, but to explain physics in terms of original variables, we use Integrated Gradients at the input.

☆ Key point

There are TWO separate issues:

1. Saturation – Gradients become very small near the final prediction, making local gradients unreliable.
2. Interpretability – Even if hidden-layer gradients are not saturated, hidden neurons are not directly interpretable in terms of physics variables.

Integrated Gradients addresses the FIRST issue by integrating gradients along the path from a baseline to the input. It also preserves attribution at the INPUT level, where the features have physical meaning.
It does NOT attempt to explain the semantics of hidden neurons.

Engineered features (DER_)
Physicist-designed discriminators, built to separate H→ττ from background

Mass variables

- DER_mass_MMC — estimated Higgs candidate mass (MMC phase-space integration)
- DER_mass_transverse_met_lep — transverse mass between MET and lepton
- DER_mass_vis — invariant mass of hadronic tau + lepton
- DER_mass_jet_jet — invariant mass of the two jets *(undefined if <2 jets)*

Momentum / energy

- DER_pt_h — |vector-sum pT| of tau + lepton + MET
- DER_pt_tot — |vector sum| of MET + tau + lepton + leading/subleading jets
- DER_sum_pt — scalar sum of pT of tau, lepton and jets (like H_T)
- DER_pt_ratio_lep_tau — pT(lepton) / pT(hadronic tau)

Angular / topology

- DER_deltar_tau_lep — ΔR separation between hadronic tau and lepton
- DER_deltaeta_jet_jet — |Δη| between the two jets *(undefined if <2 jets)*
- DER_prodeta_jet_jet — product of the two jet pseudorapidities *(undefined if <2 jets)*
- DER_met_phi_centrality — centrality of MET φ w.r.t. tau and lepton
- DER_lep_eta_centrality — centrality of lepton η w.r.t. the two jets *(undefined if <2 jets)*

Primitive features (PRI_)
Raw detector-level quantities

Hadronic tau

- PRI_tau_pt — transverse momentum
- PRI_tau_eta — pseudorapidity
- PRI_tau_phi — azimuth

Lepton (e or μ)

- PRI_lep_pt — transverse momentum
- PRI_lep_eta — pseudorapidity
- PRI_lep_phi — azimuth

Missing energy

- PRI_met — missing transverse energy (MET)
- PRI_met_phi — azimuth of MET
- PRI_met_sumet — total transverse energy in the detector

Jets

- PRI_jet_num — number of jets (0–3, capped at 3)
- PRI_jet_leading_pt / _eta / _phi — leading jet kinematics *(undefined if 0 jets)*
- PRI_jet_subleading_pt / _eta / _phi — subleading jet kinematics *(undefined if <2 jets)*
- PRI_jet_all_pt — scalar sum of pT of all jets