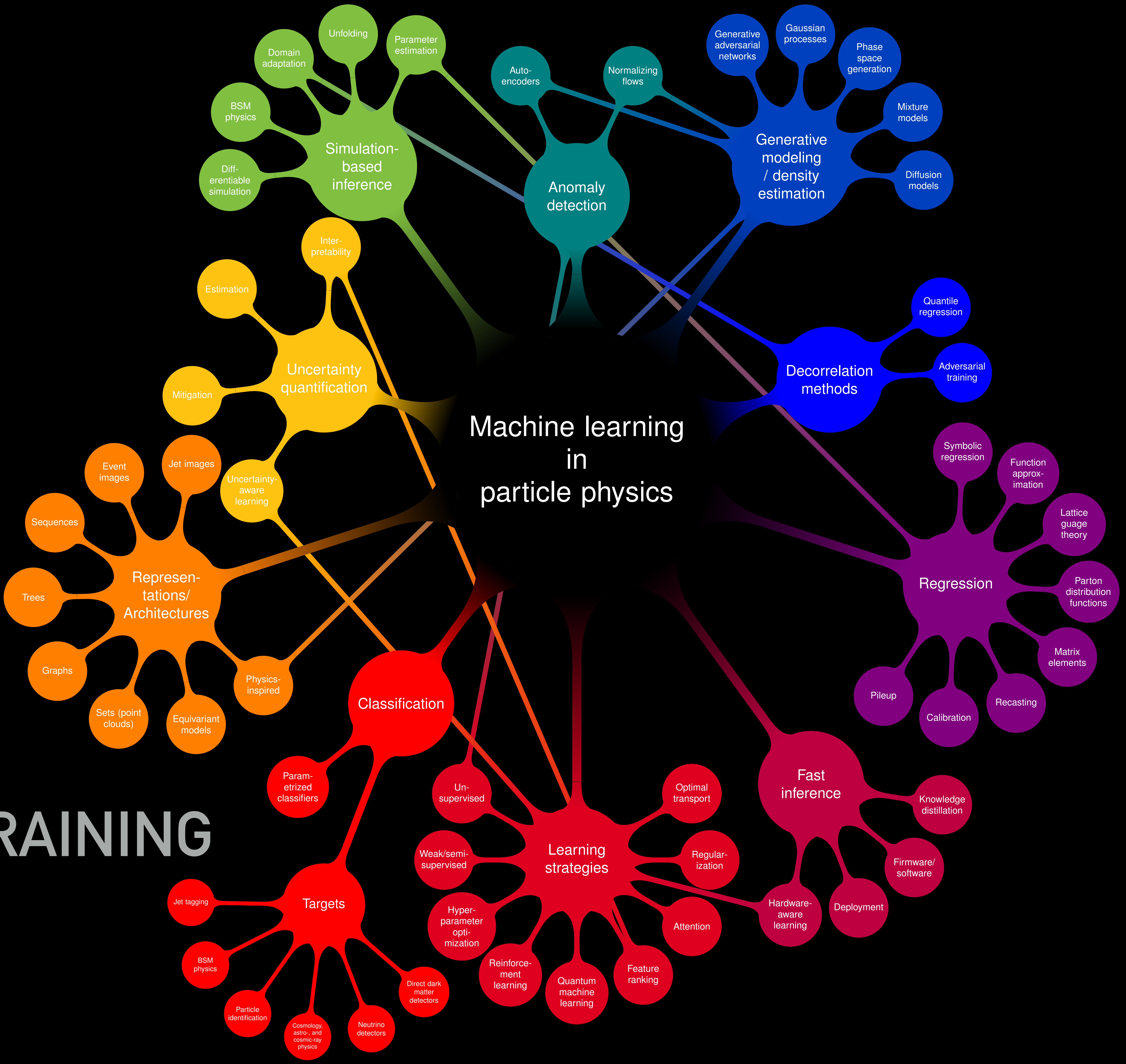
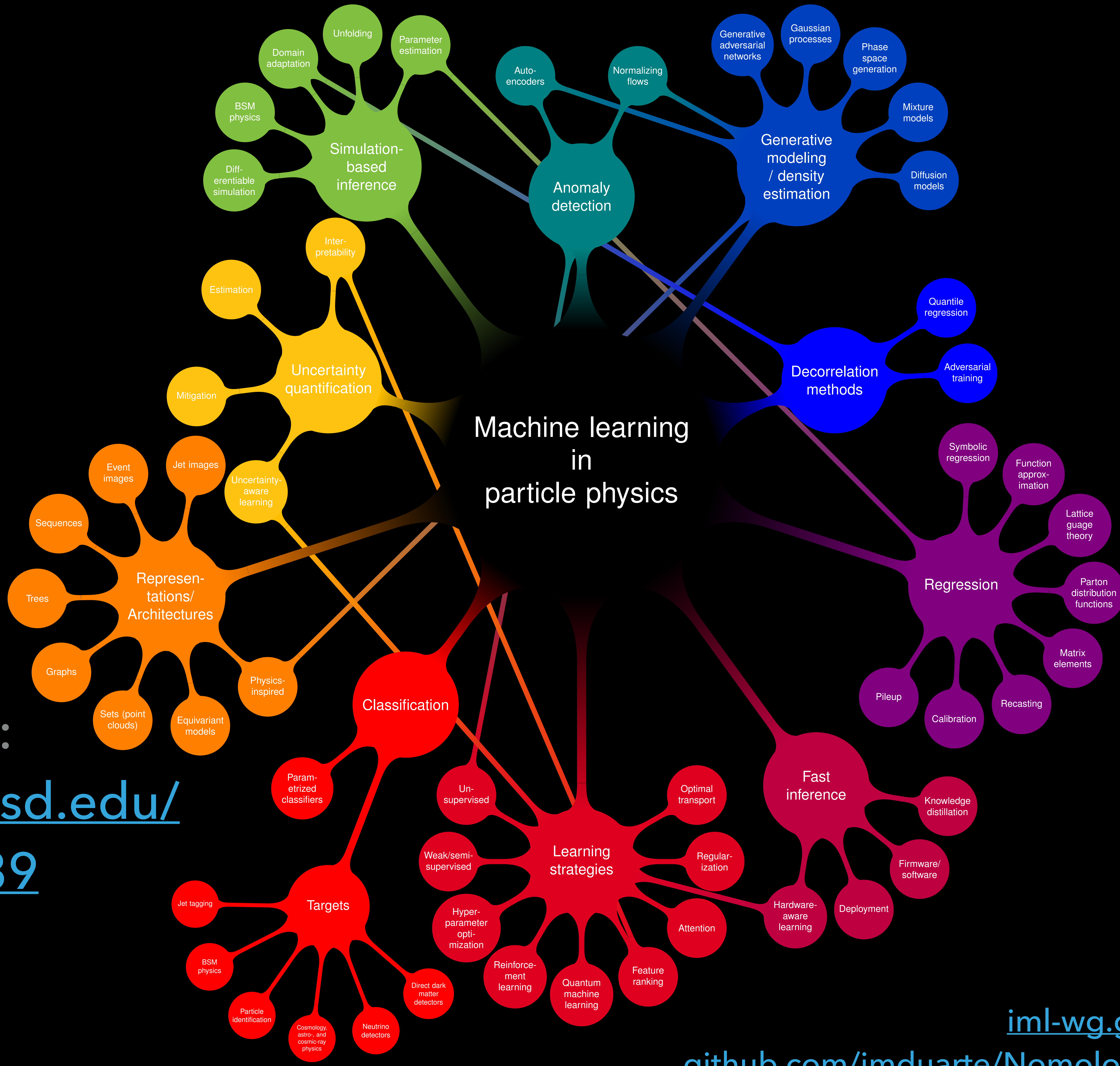
UCSD



JAVIER DUARTE

US ATLAS ML TRAINING

JULY 25, 2023



Machine learning in particle physics

- Simulation-based inference
  - Unfolding
  - Parameter estimation
  - Domain adaptation
  - BSM physics
  - Differentiable simulation
- Anomaly detection
  - Auto-encoders
  - Normalizing flows
- Generative modeling / density estimation
  - Generative adversarial networks
  - Gaussian processes
  - Phase space generation
  - Mixture models
  - Diffusion models
- Decorrelation methods
  - Quantile regression
  - Adversarial training
- Regression
  - Symbolic regression
  - Function approximation
  - Lattice guage theory
  - Parton distribution functions
  - Matrix elements
  - Recasting
  - Calibration
  - Pileup
- Fast inference
  - Knowledge distillation
  - Firmware/ software
  - Deployment
  - Hardware-aware learning
- Learning strategies
  - Optimal transport
  - Regularization
  - Attention
  - Feature ranking
  - Quantum machine learning
  - Reinforcement learning
  - Hyperparameter optimization
  - Weak/semi-supervised
  - Unsupervised
- Classification
  - Parametrized classifiers
  - Targets
    - Jet tagging
    - BSM physics
    - Particle identification
    - Cosmology, astro-, and cosmic-ray physics
    - Neutrino detectors
    - Direct dark matter detectors
- Representations/ Architectures
  - Event images
  - Jet images
  - Sequences
  - Trees
  - Graphs
  - Sets (point clouds)
  - Equivariant models
  - Physics-inspired
- Uncertainty quantification
  - Interpretability
  - Estimation
  - Mitigation
  - Uncertainty-aware learning

Full course:

jduarte.physics.ucsd.edu/phys139_239

iml-wg.github.io/HEPML-LivingReview

github.com/jmduarte/Nomological_Net_ML_Particle_Physics

I. BASICS

II. DATA REPRESENTATIONS & SYMMETRIES

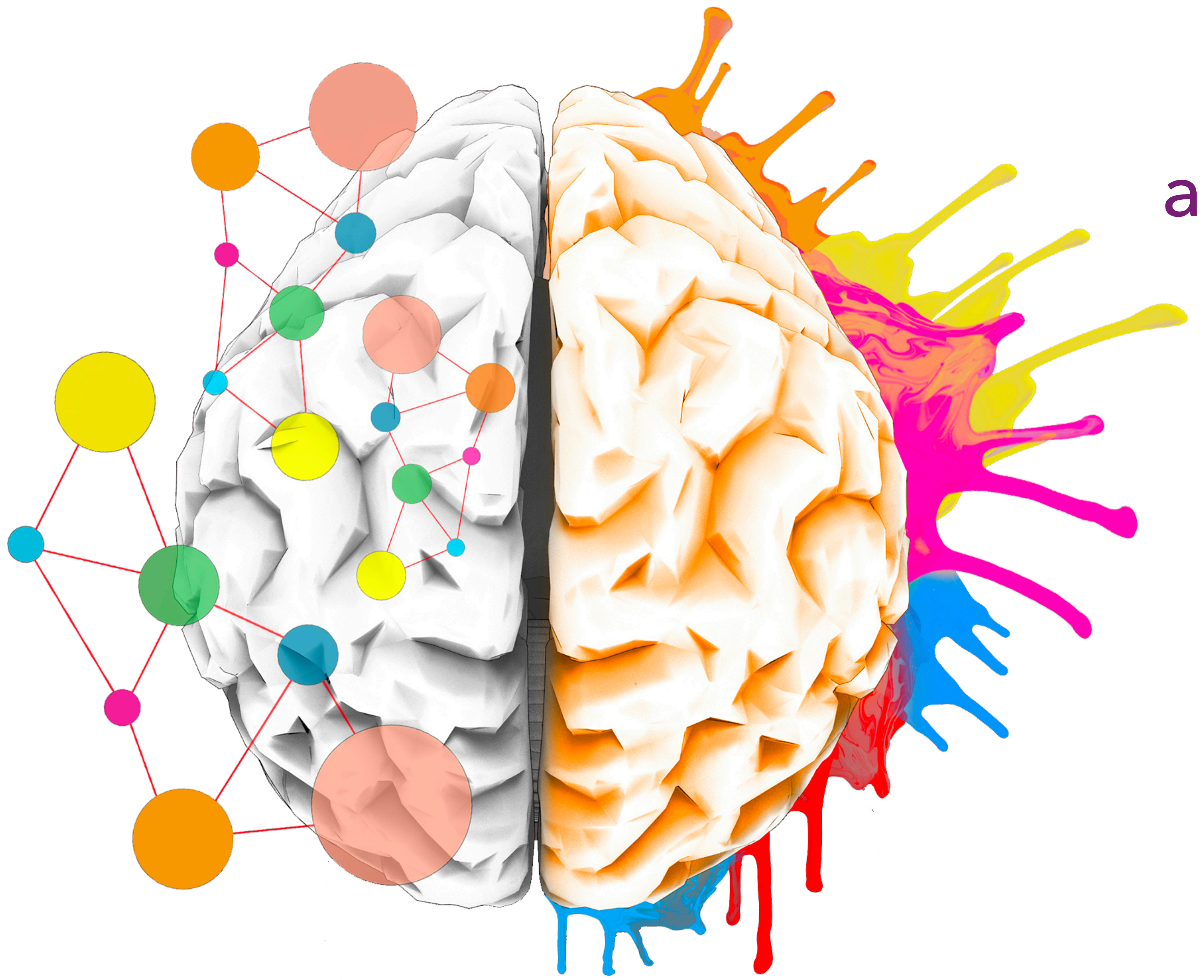
III. ANOMALY DETECTION

IV. GENERATIVE MODELING

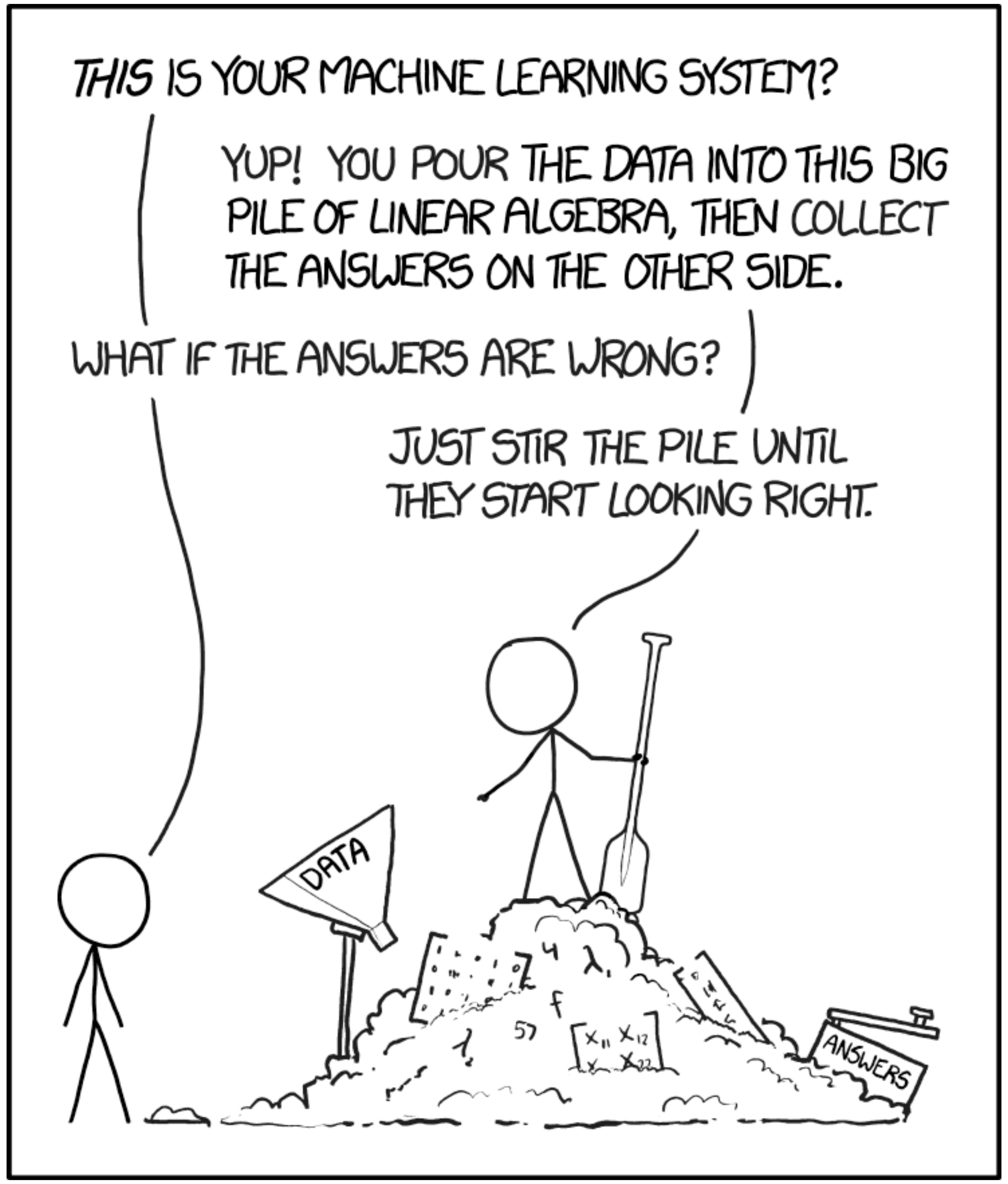
V. SUMMARY & OUTLOOK

WHAT IS MACHINE LEARNING?

- Science and art of learning automatically from data and experience



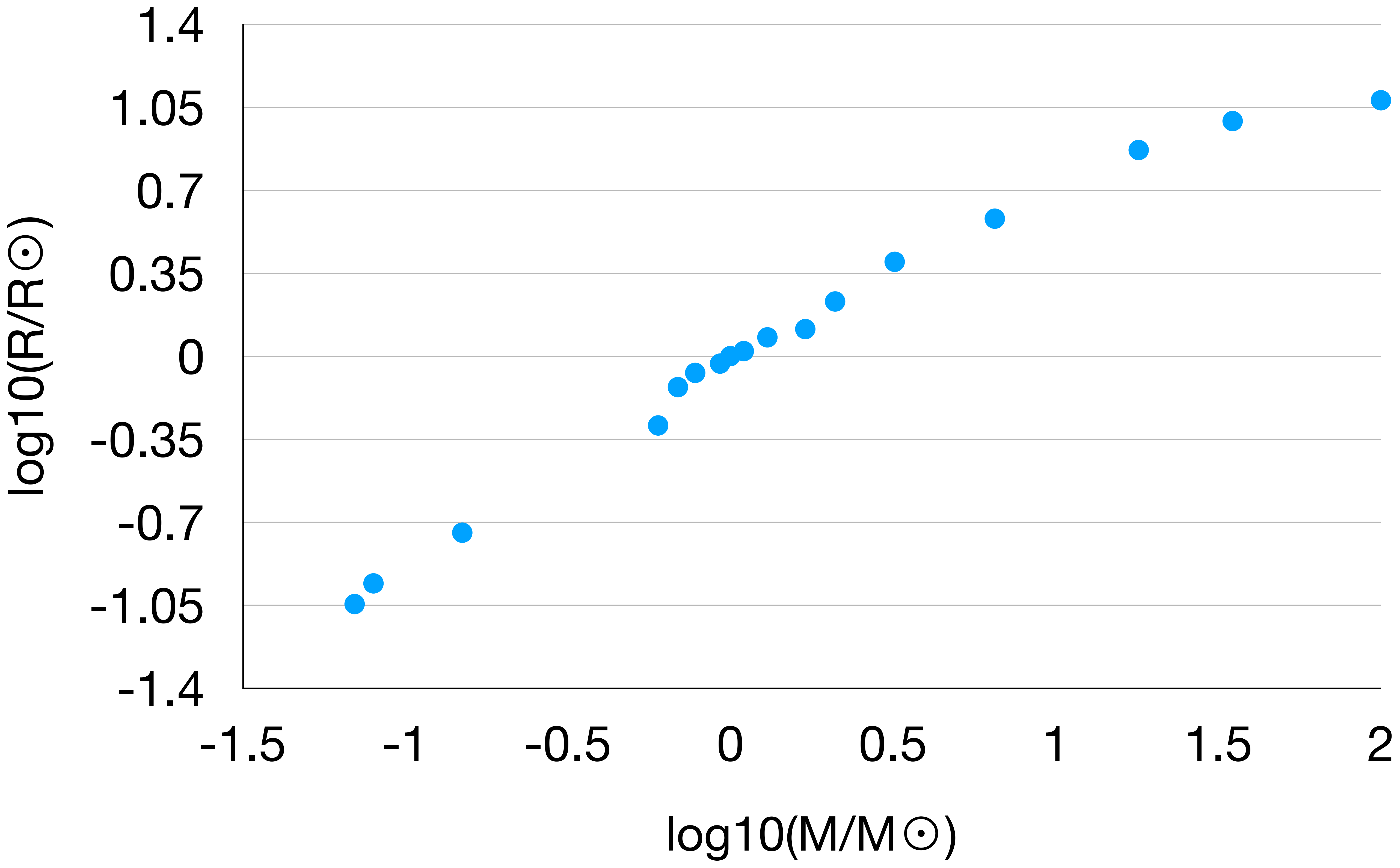
Also, a lot of calculus, linear algebra, statistics, group theory, …


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

- Large overlap with data mining:
  - ML focuses on algorithms, DM on discovering patterns

SUPERVISED LEARNING

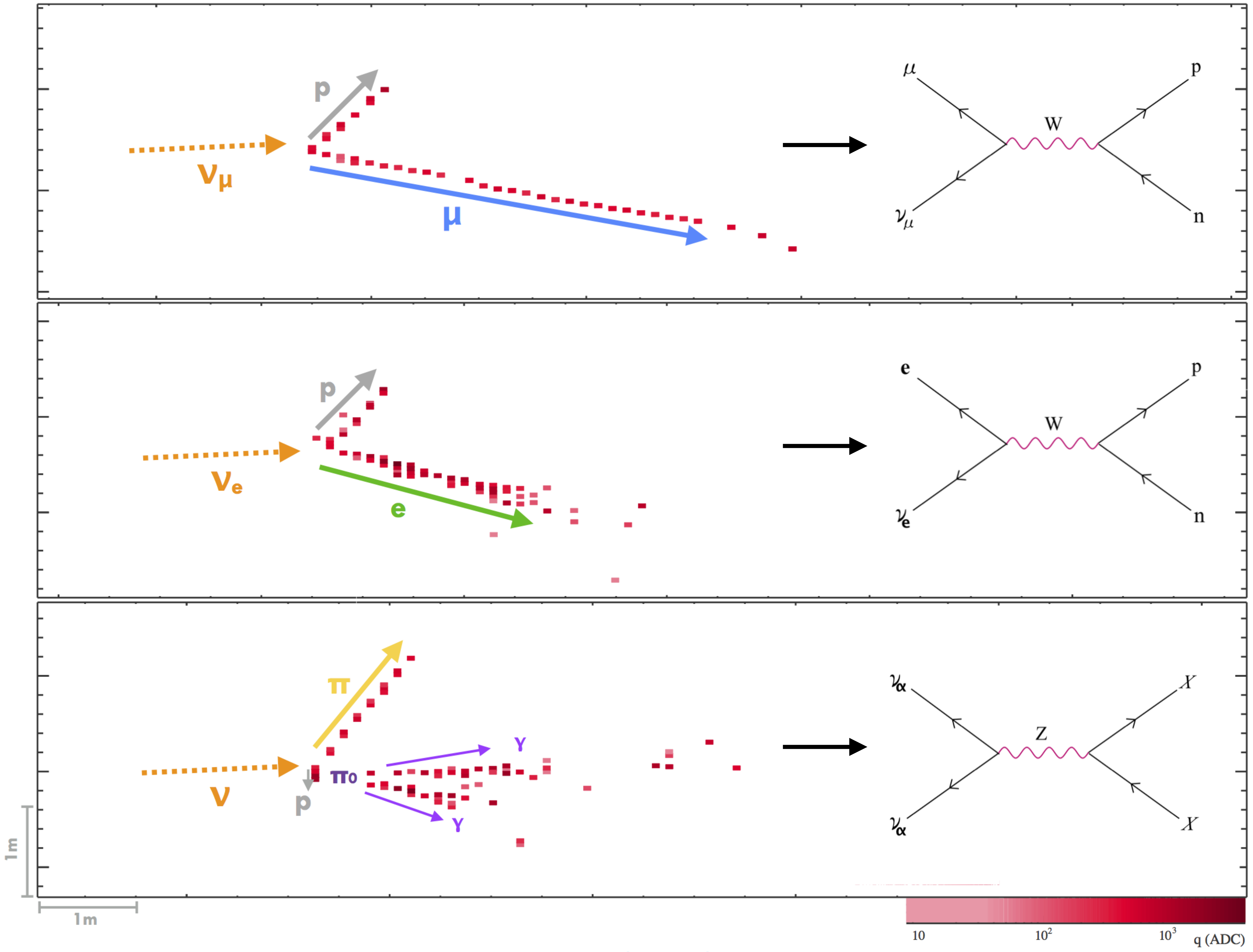
- Learn a function $f : X \to Y$ from an input space X (observations) to an output space Y (targets), using a set of labeled examples$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$.
- Example 1: Predict stellar radius given stellar mass



SUPERVISED LEARNING

- Learn a function $f : X \to Y$ from an input space X (observations) to an output space Y (targets), using a set of labeled examples$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$.

- Example 2: Classify images of neutrino interactions



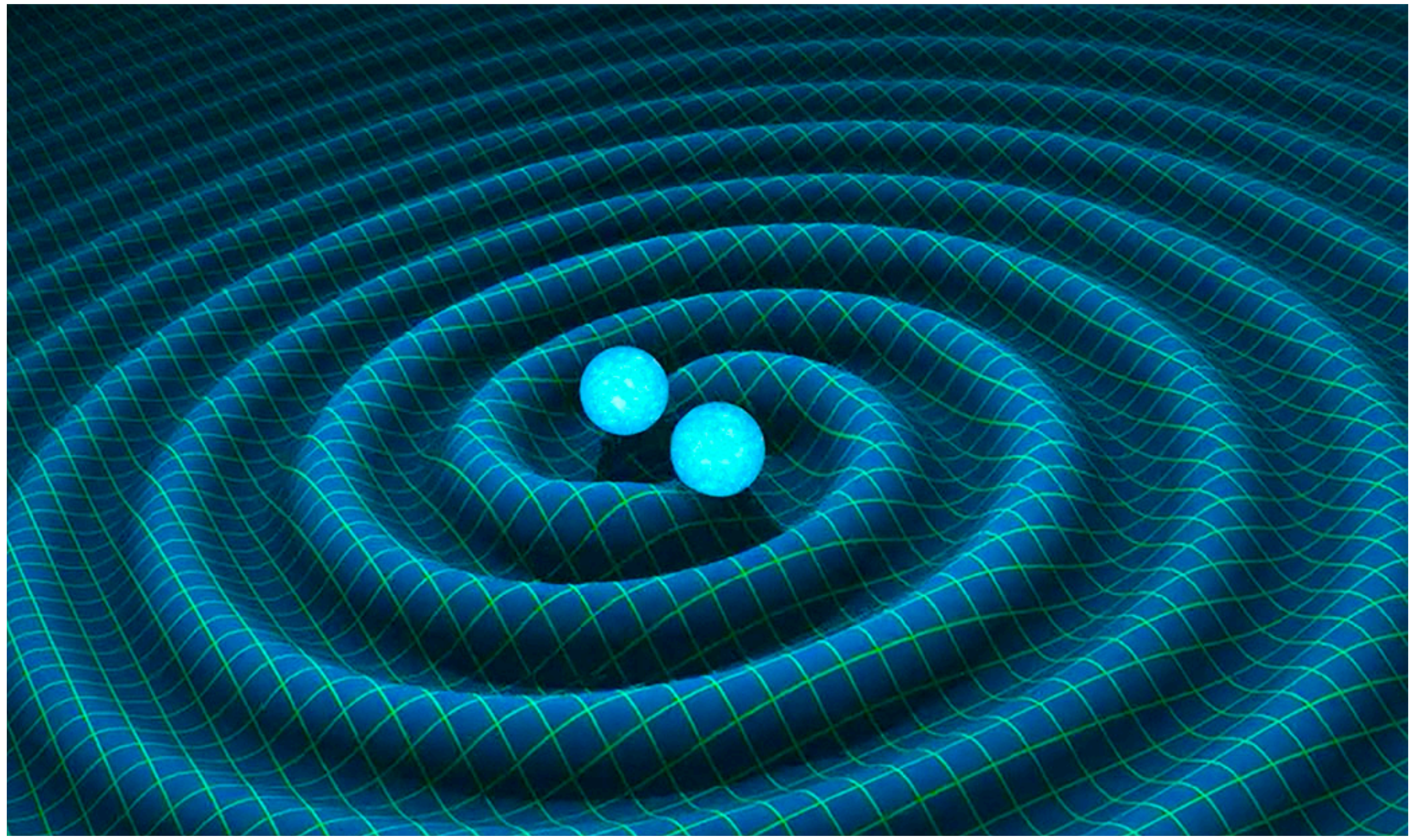
arXiv:1604.01444

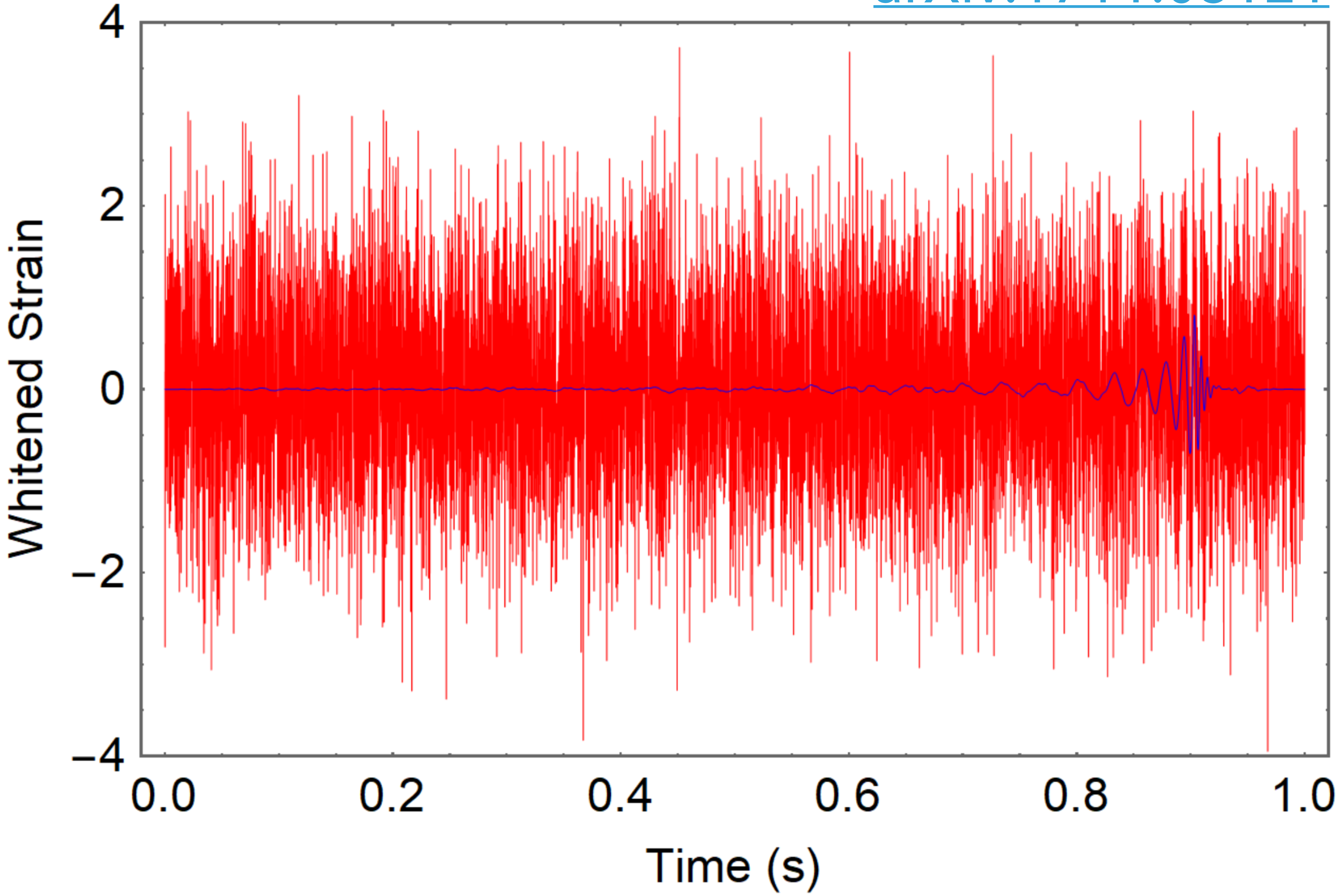
SUPERVISED LEARNING

- Learn a function $f : X \to Y$ from an input space X (observations) to an output space Y (targets), using a set of labeled examples$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$.

- Example 3: Reduce noise in a time-series trace to identify a gravitational wave signal

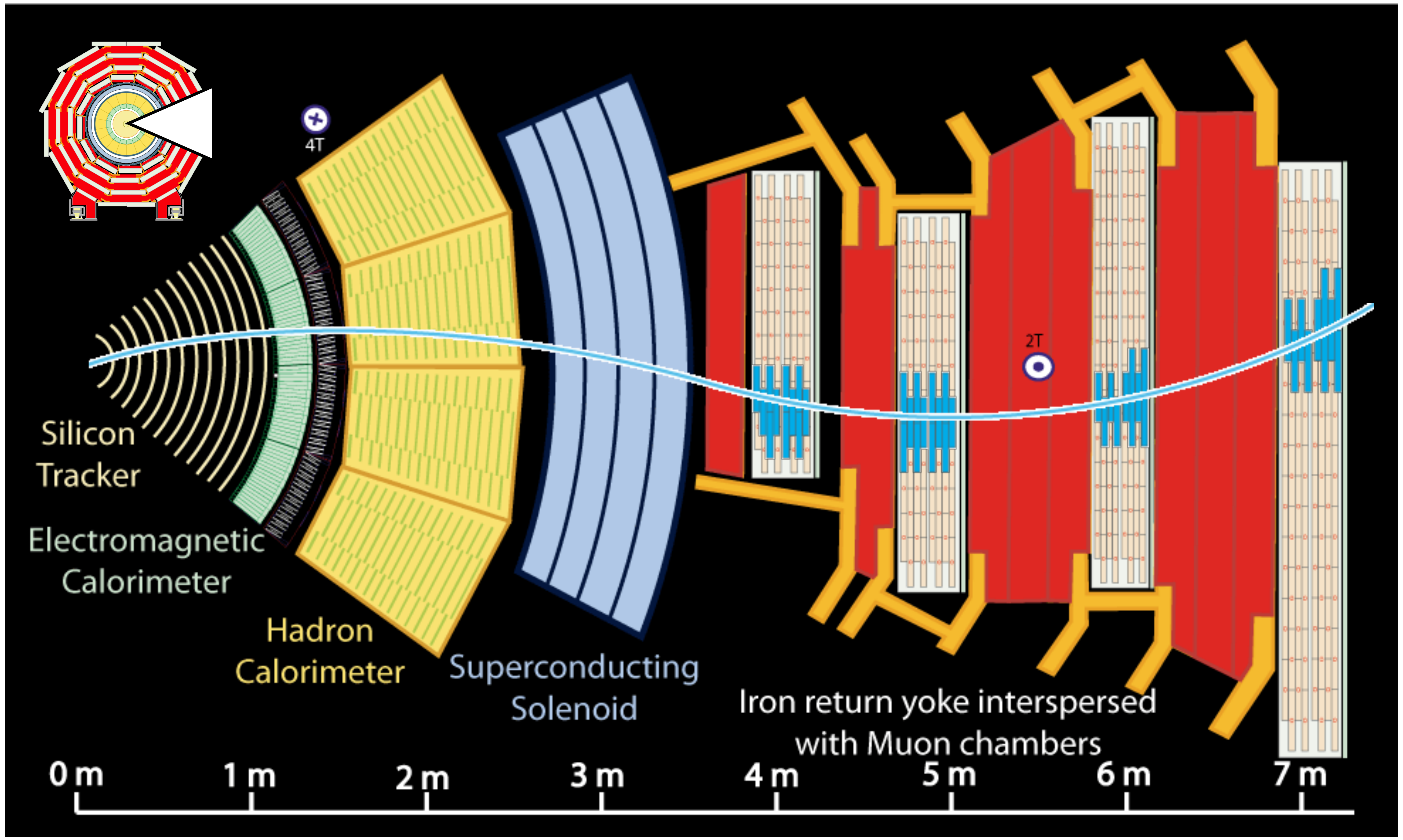
arXiv:1711.03121





SUPERVISED LEARNING

- Learn a function $f : X \rightarrow Y$ from an input space X (observations) to an output space Y (targets), using a set of labeled examples$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$.
- Example 4: Estimate particle momentum, charge, type, etc. from detector hits

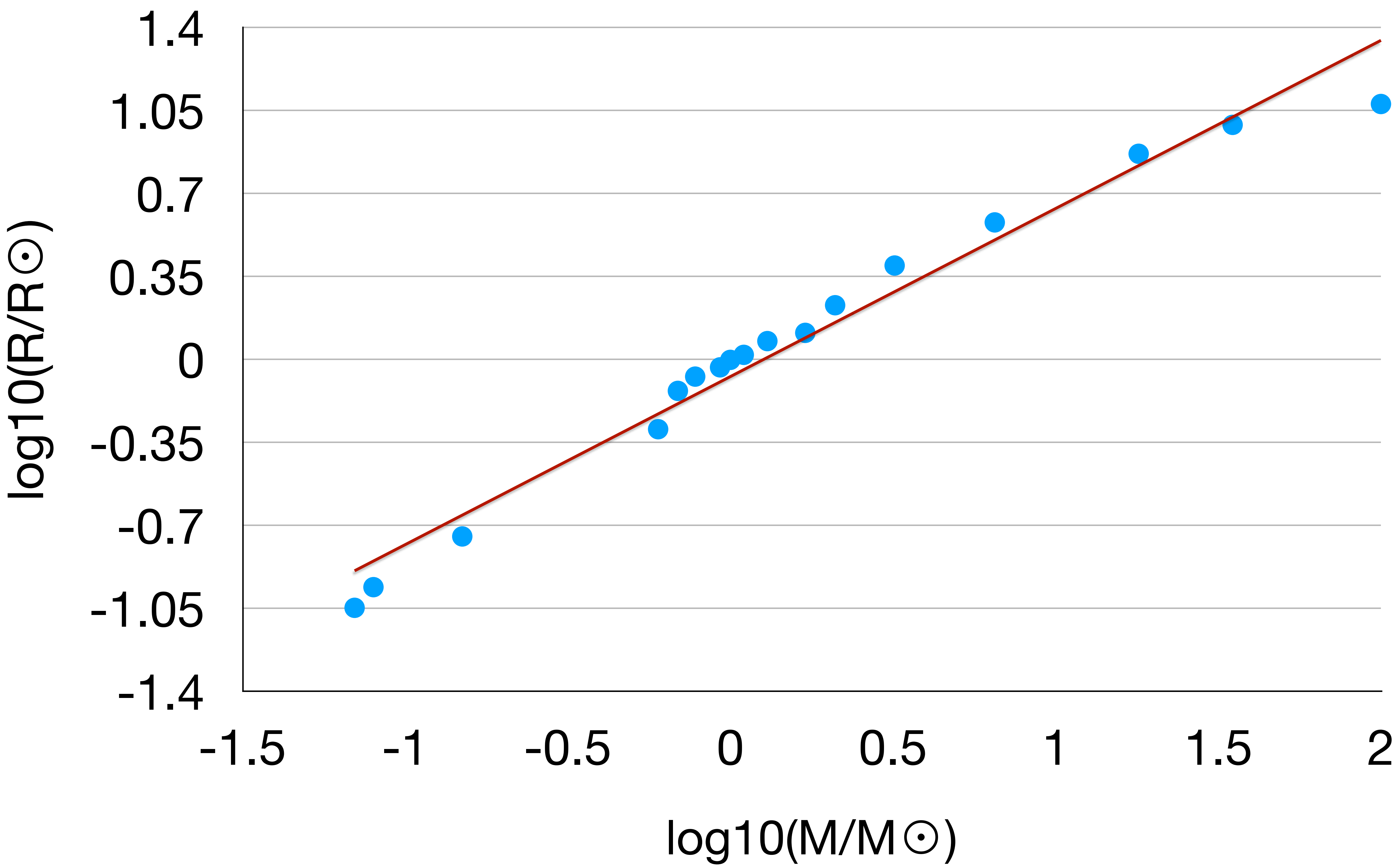


$$\rightarrow \begin{pmatrix} E \\ p_x \\ p_y \\ p_z \end{pmatrix}, q, \text{type}, p_{\text{pileup}}, \ldots$$

arXiv:2101.08578

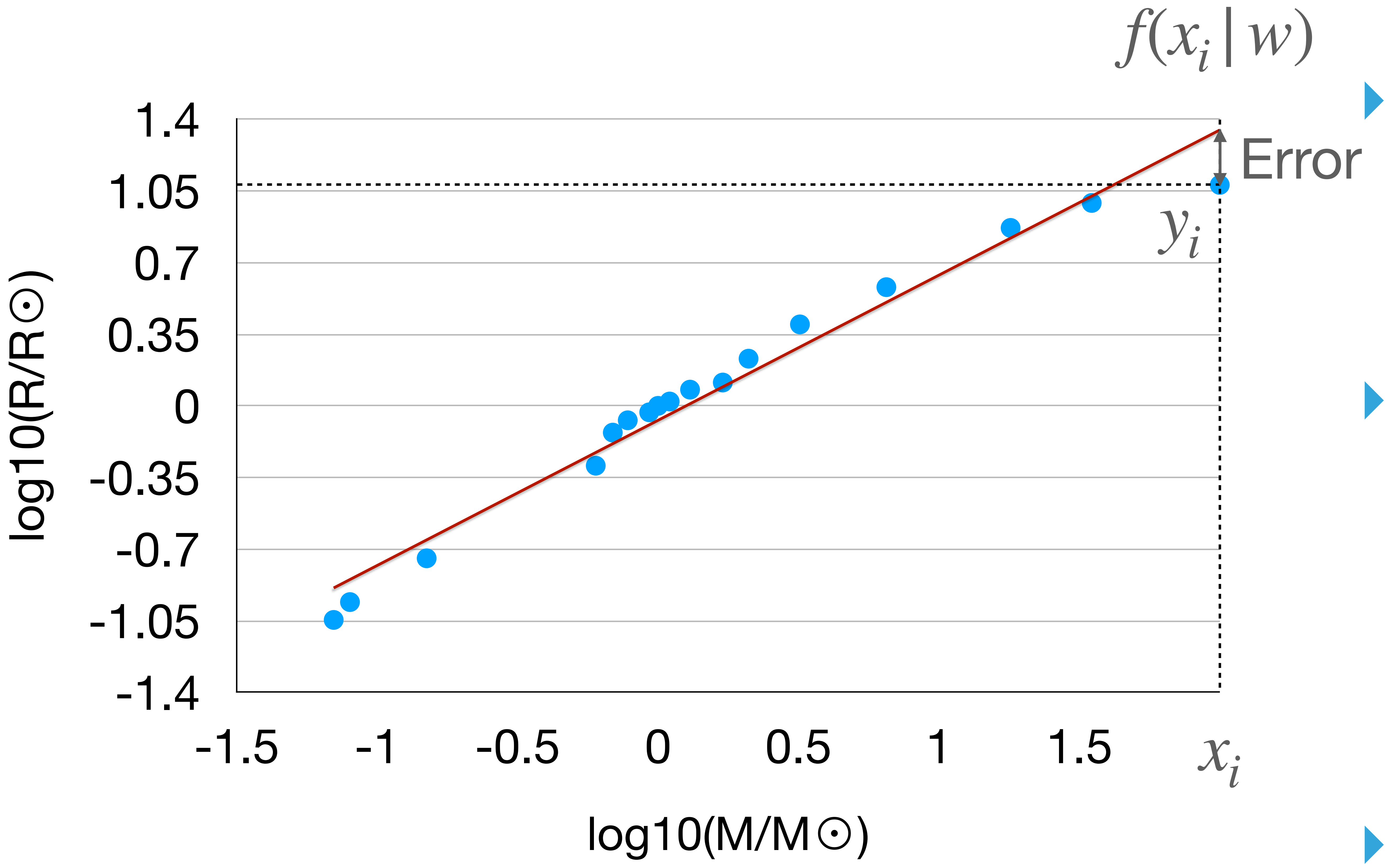
MACHINE LEARNING APPROACH

- Collect a **labeled** training set (supervision)
  - Often requires **simulation** where the “ground truth” is known



- Train a model using a learning algorithm (find patterns in the data)

LINEAR REGRESSION



- **Linear model**:

  $f(x \mid w) = w^\intercal x \quad (w \in \mathbb{R}^{D+1})$

- How do we select the parameters w?
  - We want $y_i \approx f(x_i \mid w)$

- **Squared loss**: $L(y, y') = (y - y')^2$

  (**Least squares**)

Learning objective: $\arg\min_{w} \sum_{i=1}^{N} L(y_i, f(x_i \mid w)) = \arg\min_{w} \sum_{i=1}^{N} (y_i - w^\intercal x_i)^2$

OPTIMIZING THE LEARNING OBJECTIVE

- In supervised learning, we want to optimize the objective

$$l(w) = \sum_{i=1}^{N} L(y_i, f(x_i \,|\, w))$$

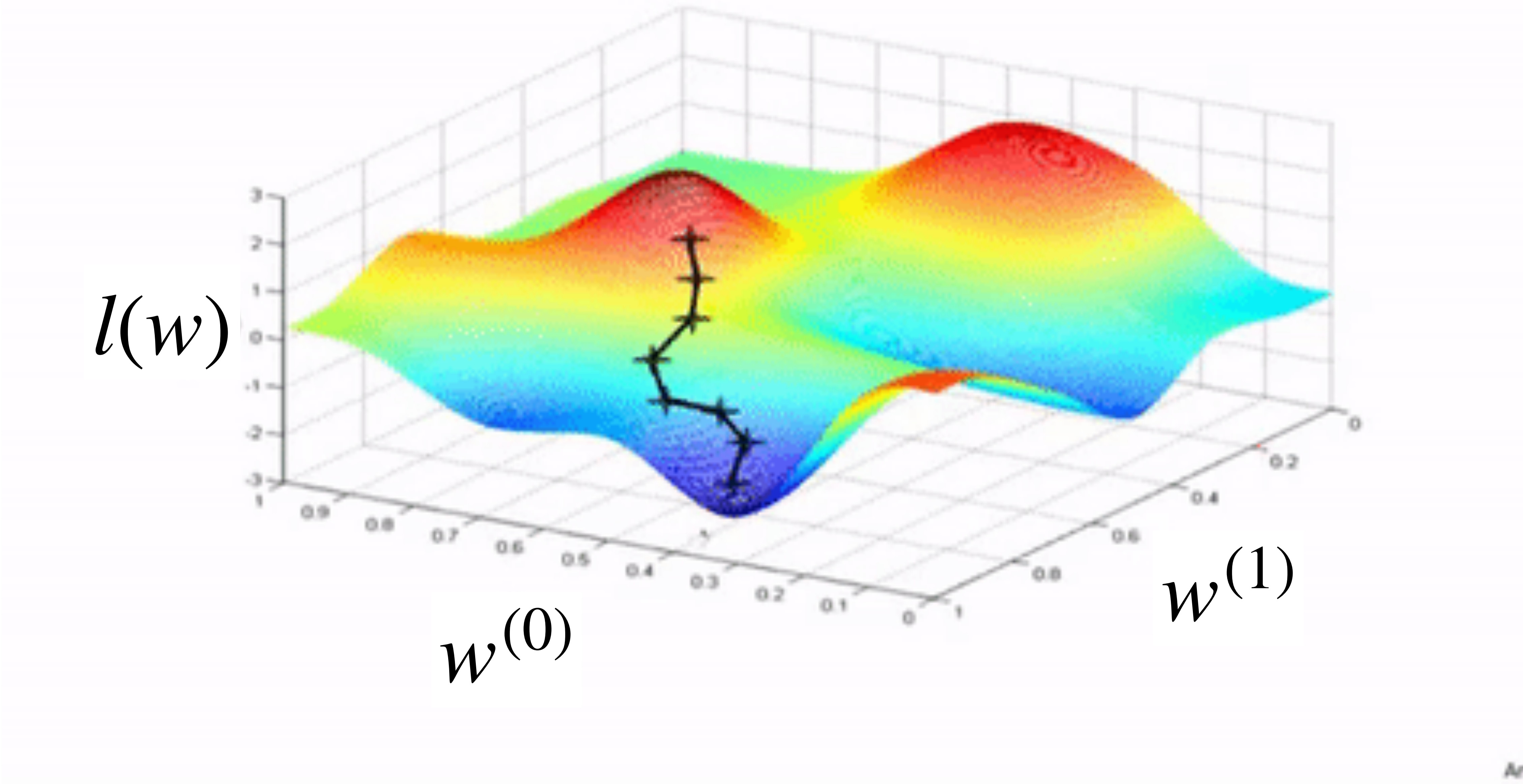
- For linear regression, there is a closed-form solution, but in general?
- We need an optimization algorithm to find the optimal (or just "good") w

GRADIENT DESCENT

- Set $w(t=0)$ to some values (e.g., $w(0)=0$ or some random value)
- At iteration t,
  - Compute the gradient $\nabla_w l(w(t))$: direction of steepest increase of $l(w)$ at $w(t)$
  - Take a small step in the opposite direction:

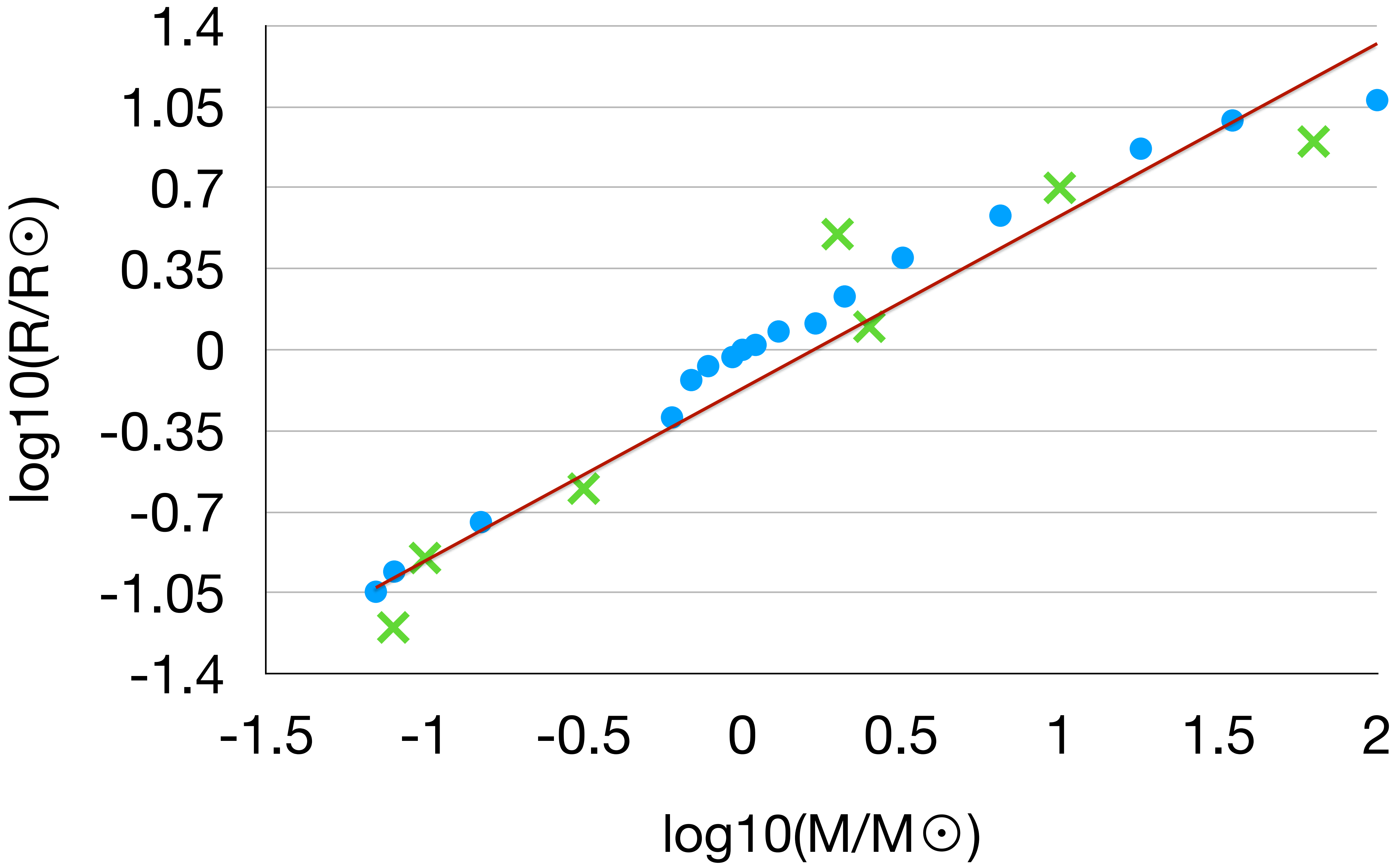
$$w(t+1) = w(t) - \eta \nabla_w l(w(t))$$

Step size / learning rate

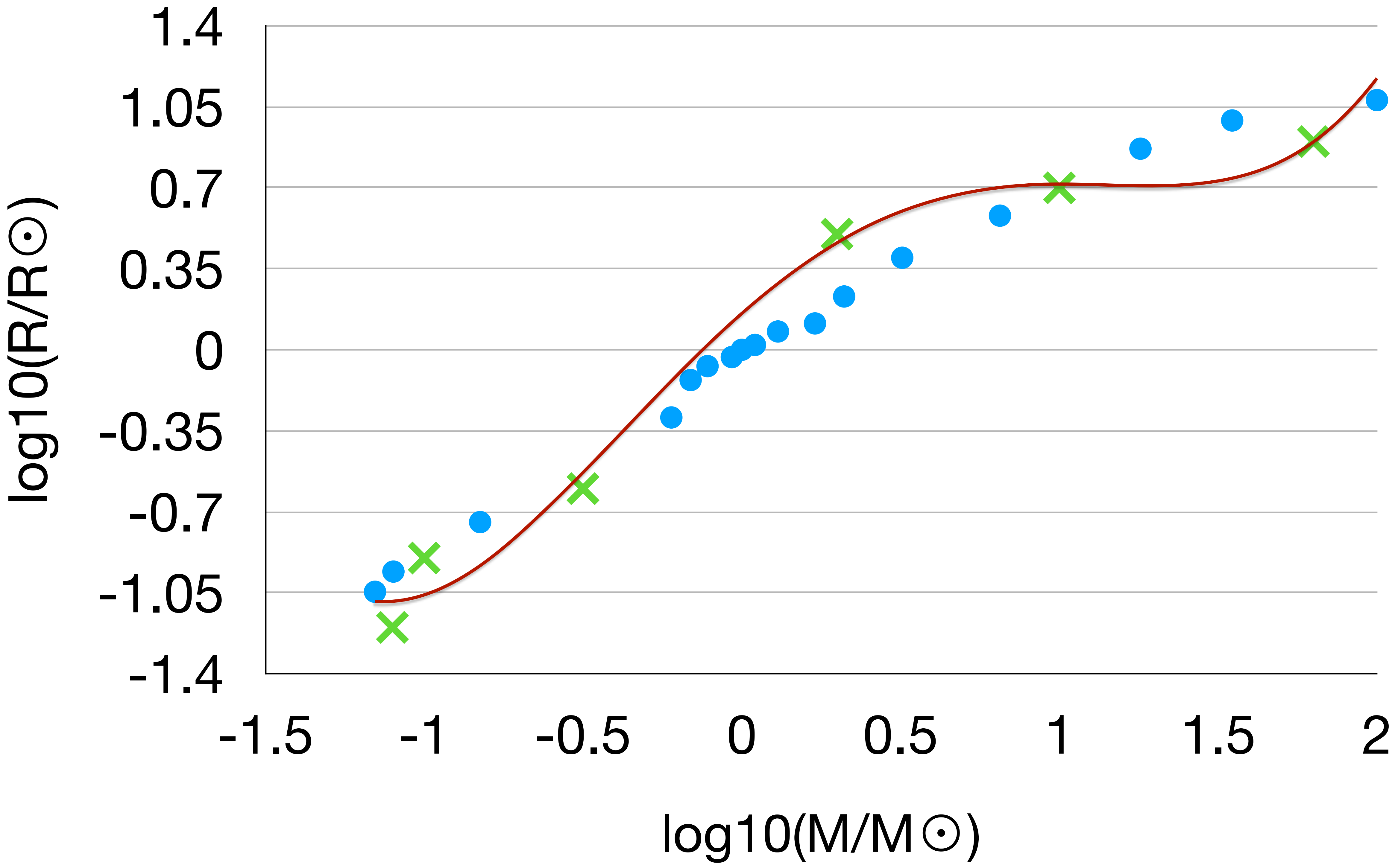


BUT DOES YOUR MODEL GENERALIZE?

- Fitting the training dataset perfectly (error = 0) does not necessarily mean the model will work well on new test data!



Linear fit: ok on both training and testing



Polynomial fit (degree 4): excellent on training, bad on testing

BIAS-VARIANCE TRADEOFF

- If L is the squared loss, we can decompose the expected test error:

$$\begin{aligned}\mathbb{E}\left[L_P(f(x\,|\,w_S)\right] &= \mathbb{E}_S\mathbb{E}_{(x,y)\sim P(x,y)}\left[L(y, f(x\,|\,w_S))\right] \\ &= \mathbb{E}_{(x,y)\sim P(x,y)}\left[\underbrace{\mathbb{E}_S\left[(f(x\,|\,w_S) - F(x))^2\right]}_{\text{Variance}} + \underbrace{(F(x) - y)^2}_{\text{(Squared) bias}}\right]\end{aligned}$$

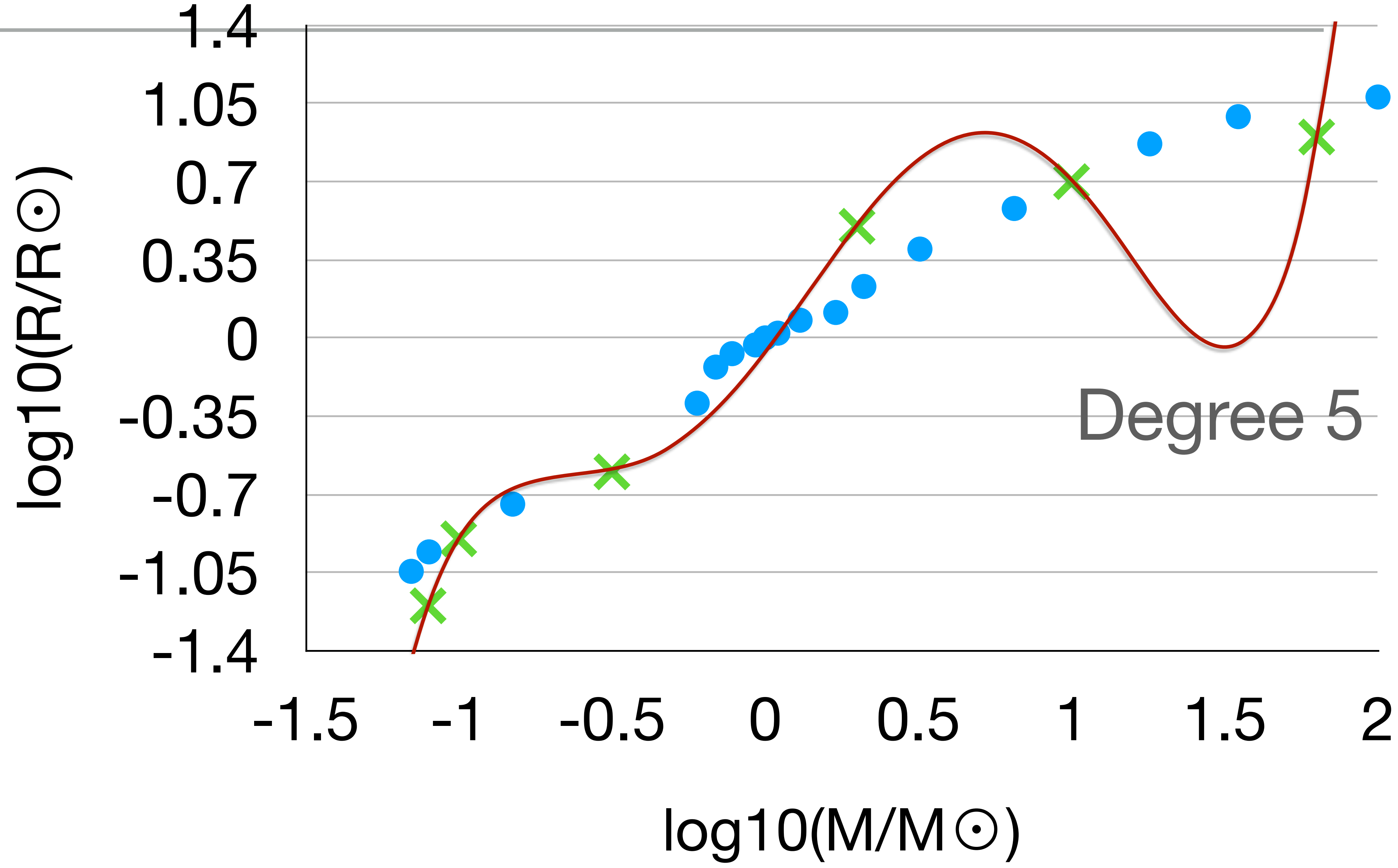
- where $F(x) = \mathbb{E}_S\left[f(x\,|\,w_S)\right]$ is the average prediction of our model over different possible training datasets

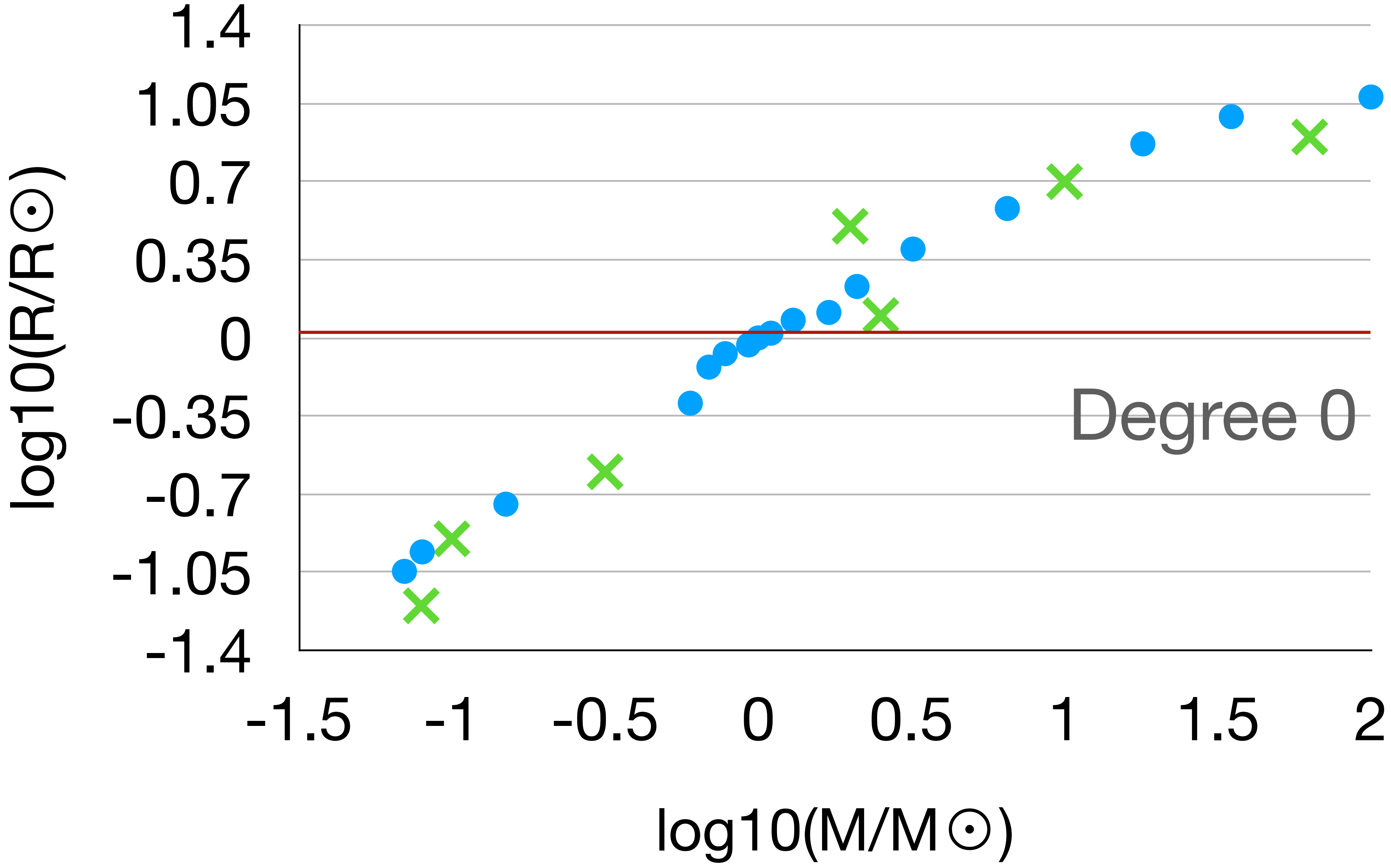
- Variance: difference in predictions when training on different datasets

- Bias: difference from ground truth

OVERFITTING VS. UNDERFITTING

- Overfitting implies high variance (unstable model class)
  - Variance increases with model complexity
  - Variance decreases with more training data
- Underfitting implies high bias
  - Even with no variance, model class has high error
  - Underfitting happens whenever model complexity is too low



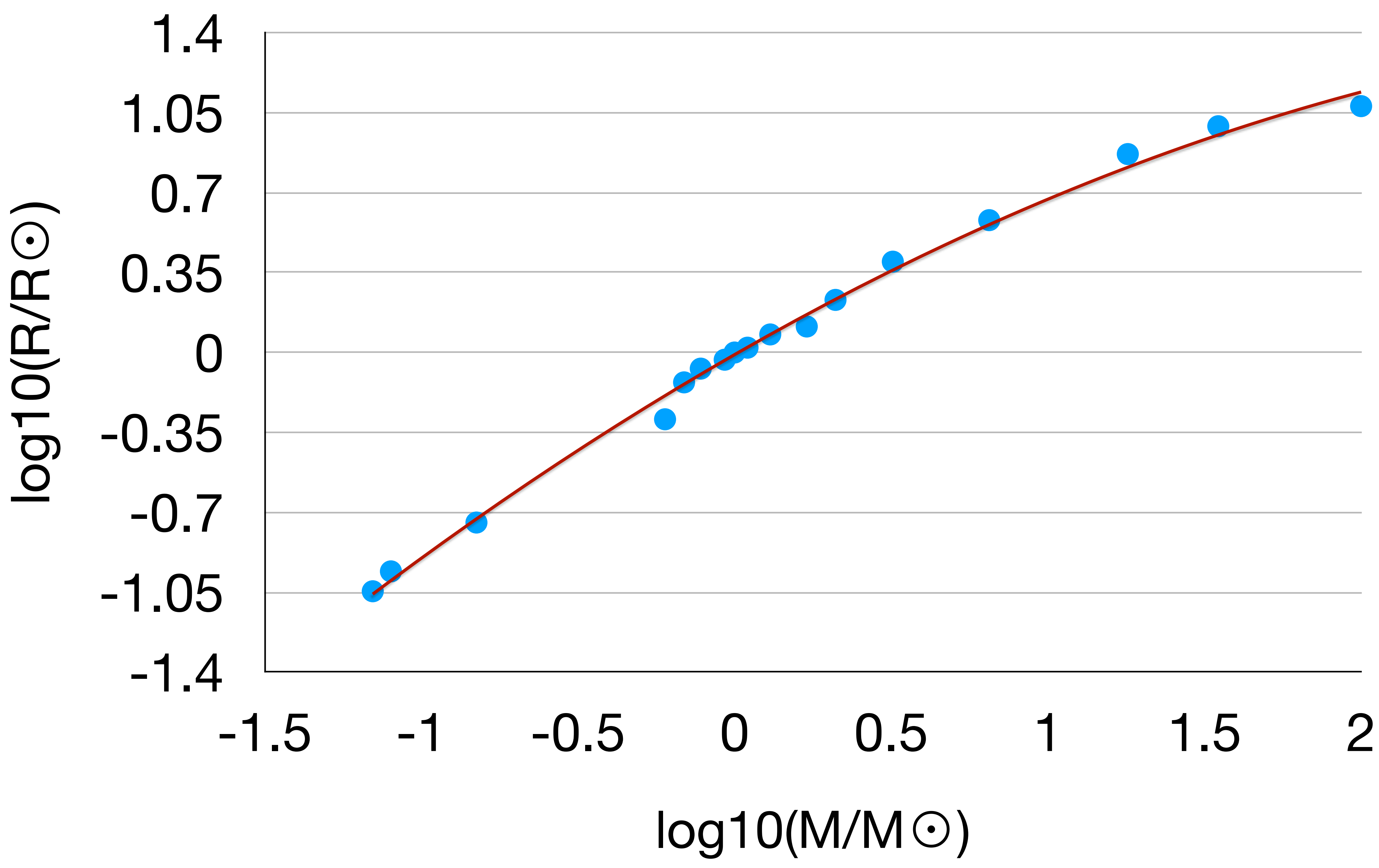


GETTING MORE OUT OF LINEAR MODELS

features / embedding of x

- Replace our input vector x with some $\phi(x)$ to make our model more expressive
- For example, if $\phi(x) = (1, x, x^2)$ then our model becomes:

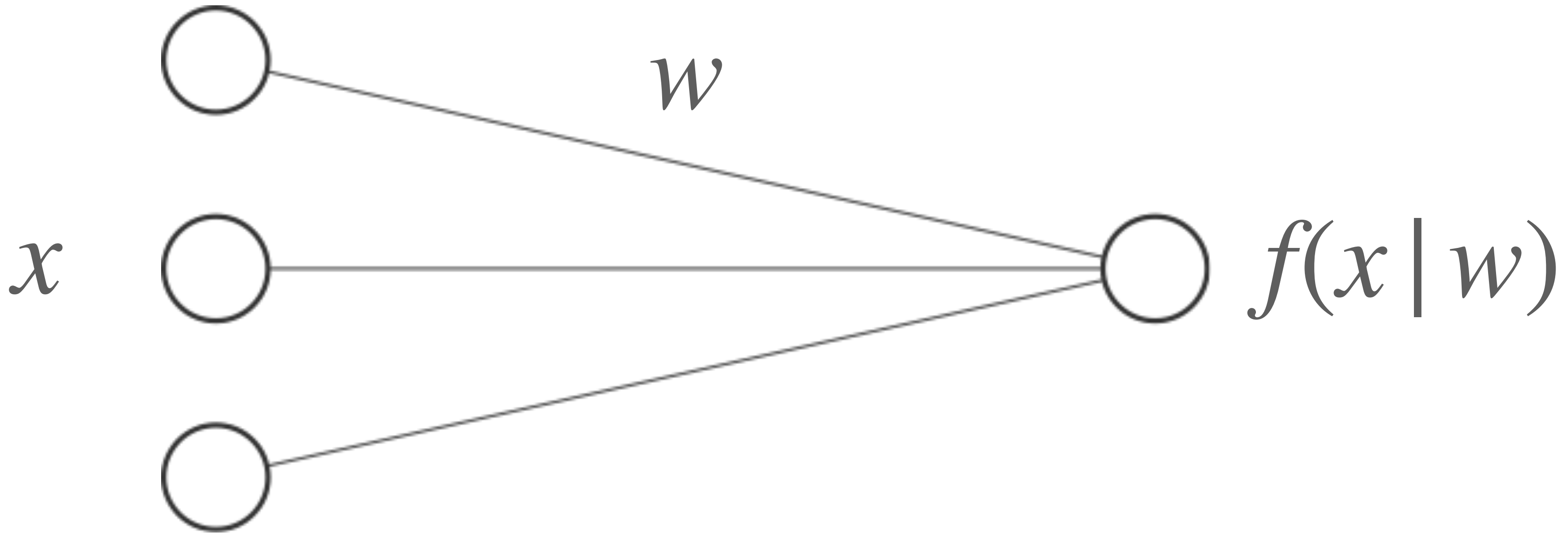
$$f(x \mid w) = w^\intercal \phi(x) = w_0 + w_1 x + w_2 x^2$$



- The model is still **linear** in the parameters w!
- More expressive than a line $w_0 + w_1 x$, so the fit is better (i.e., training error is lower)

LINEAR MODELS: WORKHORSE OF MACHINE LEARNING

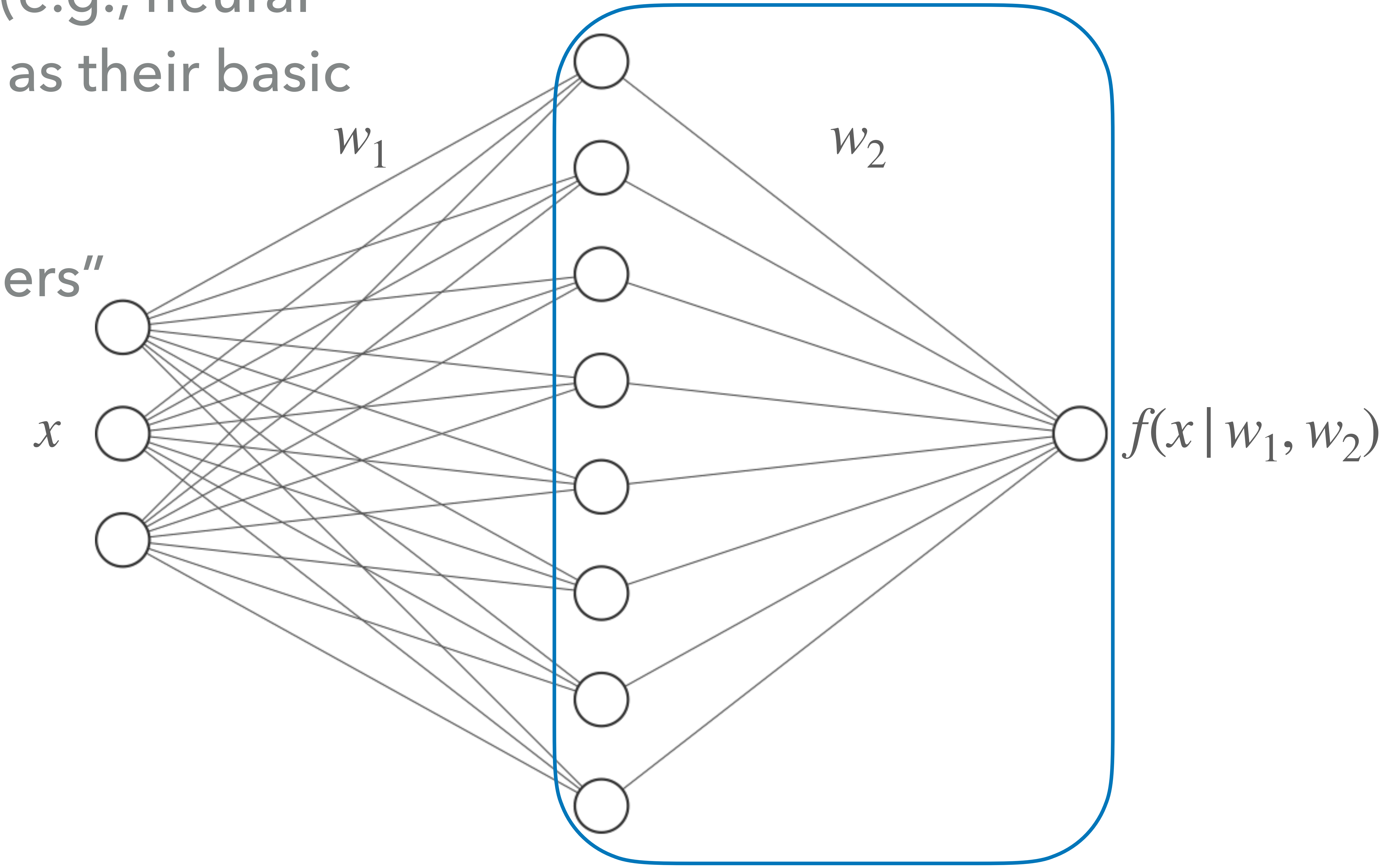
- Linear models on top of good features can yield excellent results
- More complex model classes (e.g., neural networks) have linear models as their basic building block
  - NNs are “automatic featurizers”



Linear model: $f(x \mid w) = w^\intercal x$

Neural network: linear model after inputs are mapped to features through a nonlinear transformation

$$f(x \mid w_1, w_2) = w_2^\intercal \sigma(w_1^\intercal x)$$



MODEL SELECTION

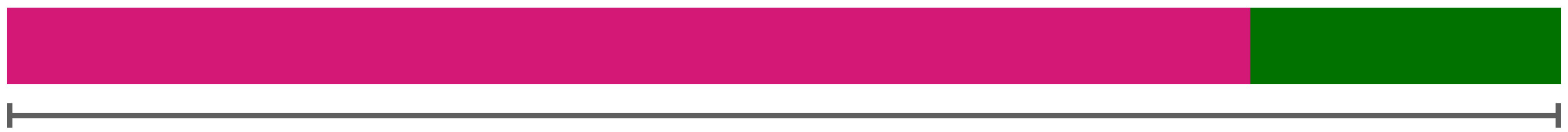
- We only have a finite training dataset
- We cannot measure the true test error
- Simple model classes underfit
- Complex model classes overfit

Bias-variance tradeoff

(but not so straightforward for deep neural networks!)

- **Goal**: Select the model class with the lowest test error

VALIDATION SET



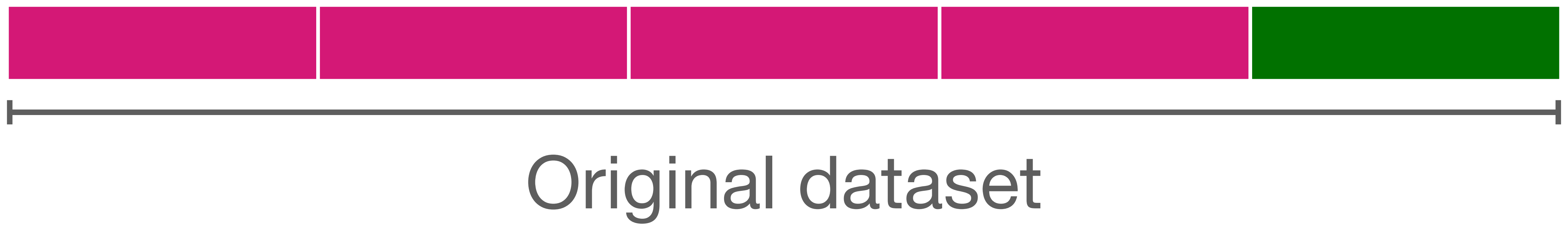
Original dataset

- Split the original dataset into a training and validation set
- Train model on the training set
- Evaluate on the validation set to estimate the test error
- Select the model class that gives the lowest estimated error
- Optionally, re-train the selected model class on the whole dataset (training + validation)
- **Issue**: we would like both training and validation sets to be as large as possible (so that the estimate is better), but they must not overlap!

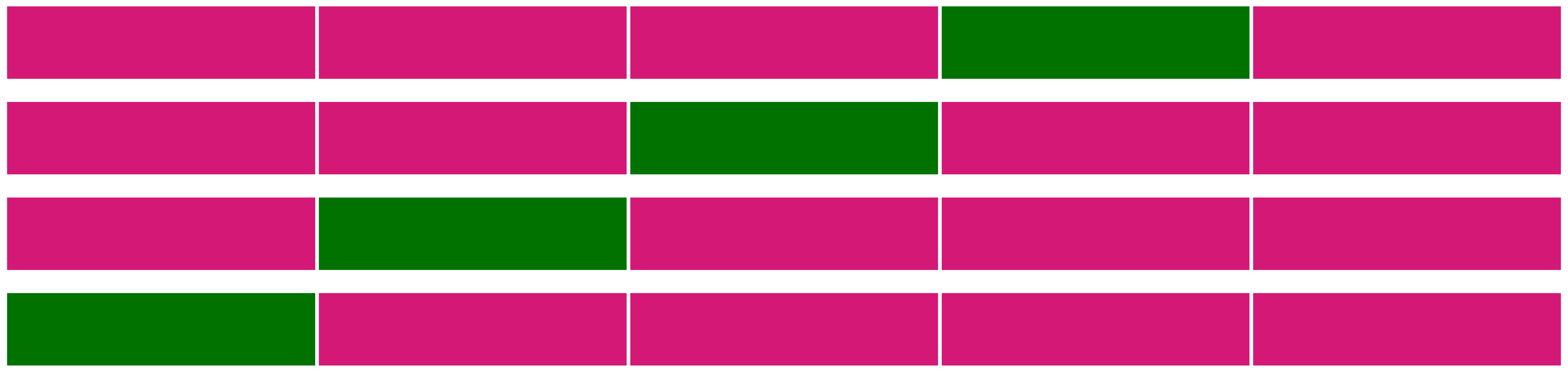
k-FOLD CROSS-VALIDATION

- Split the original dataset into k equal parts (e.g, $k = 5$)

- Train on the $k - 1$ parts and validate on the remaining one



- Repeat for every choice of the $k - 1$ parts and average the validation errors



- **Advantage**: use all data as validation to improve the estimate of the test error, at the cost of more computation (k trainings)

SUPERVISED LEARNING PIPELINE

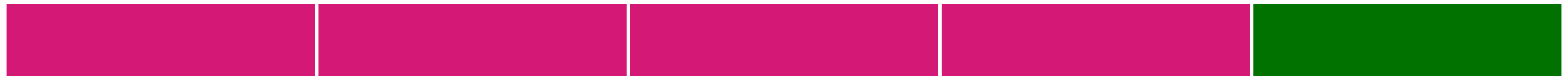
- Training dataset: $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $x \in \mathbb{R}^D$ and $y \in \mathbb{R}$

or $\phi(x)$ instead of x

- Model / hypothesis class: $f(x \,|\, w) = w^\intercal x$ (linear models)
- Loss function: $L(y, y') = (y - y')^2$ (squared loss)
- Optimization algorithm to minimize the learning objective:

$$\arg\min_{w} \sum_{i=1}^{N} L(y_i, f(x_i \,|\, w))$$

- Cross validation and model selection:



- Testing and deployment

Important: if a testing set is available, never use it to make decisions on the model!

I. BASICS

II. DATA REPRESENTATIONS & SYMMETRIES

III. ANOMALY DETECTION

IV. GENERATIVE MODELING

V. SUMMARY & OUTLOOK

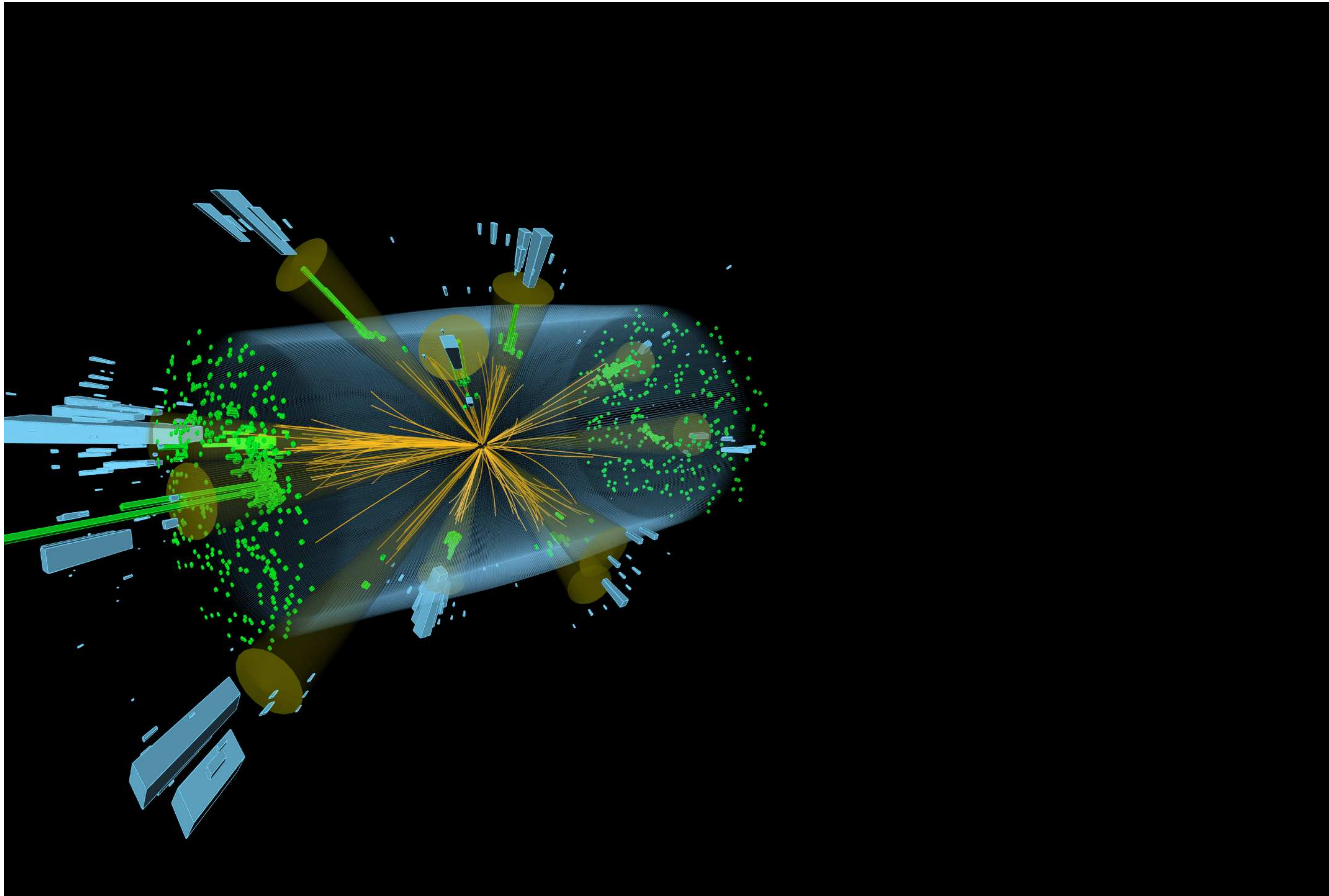
REPRESENTATIONS ⟷ INDUCTIVE BIAS ⟷ ALGORITHMS

High-level (expert) variables	Shallow neural network, boosted decision tree, …
Ordered list of particles	1D convolutional neural network, recurrent neural network
Images	2D convolutional neural network
Set of particles	Deep set (energy flow network)
Graph of particles	Graph neural network
Lorentz scalars/vectors	Lorentz-equivariant network

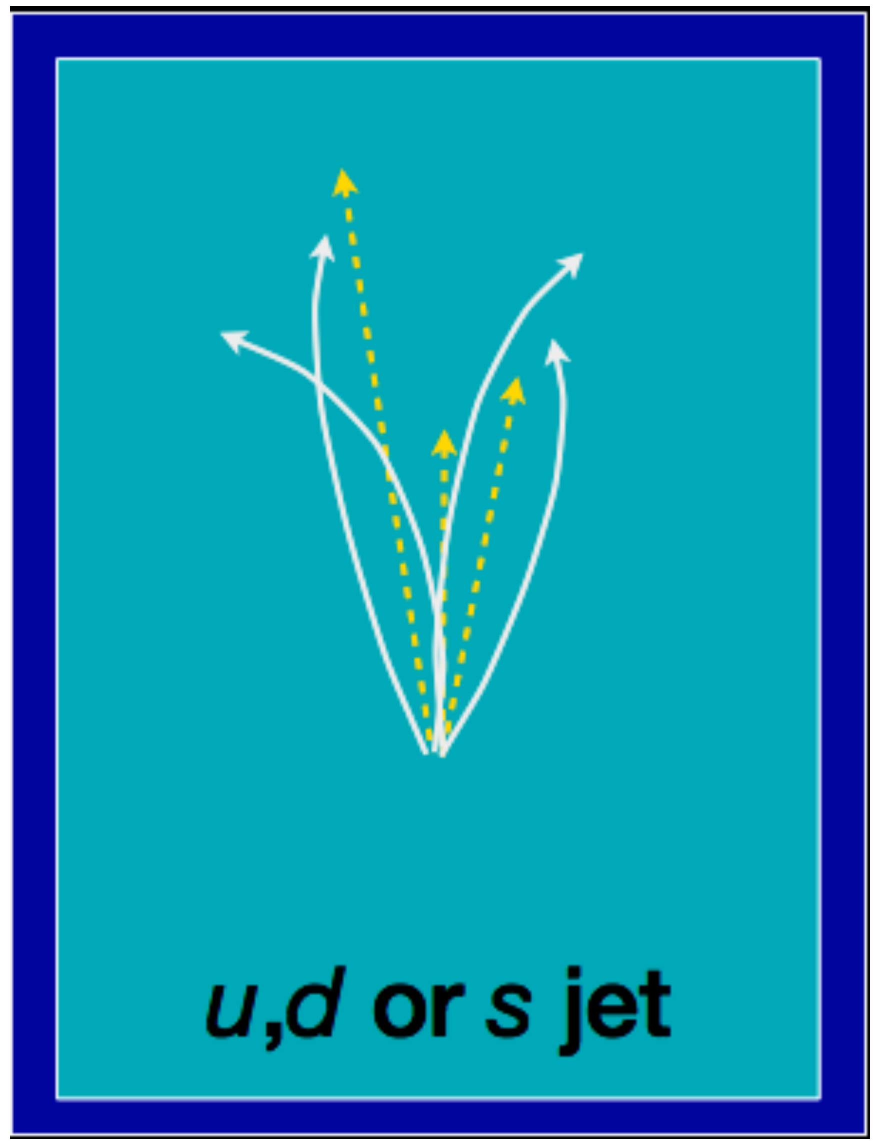
TRANSFORMERS

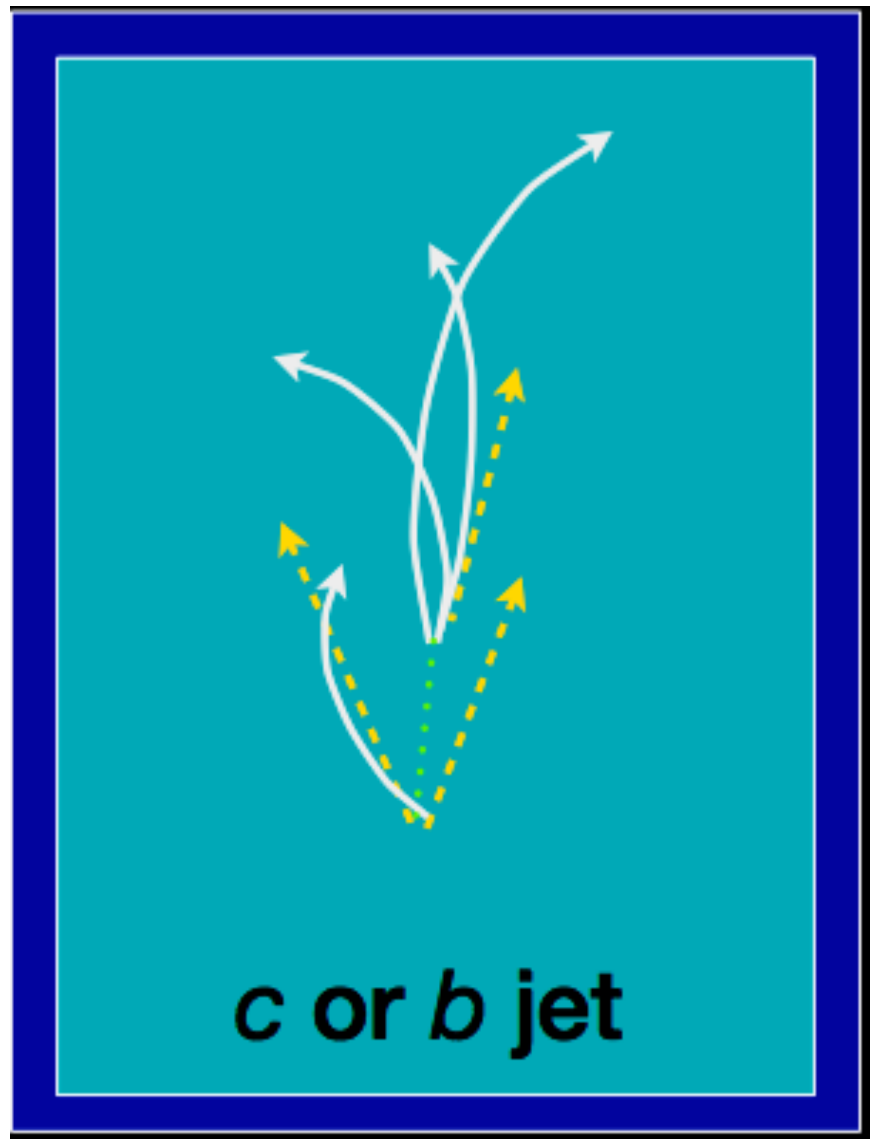
COLLISION EVENT

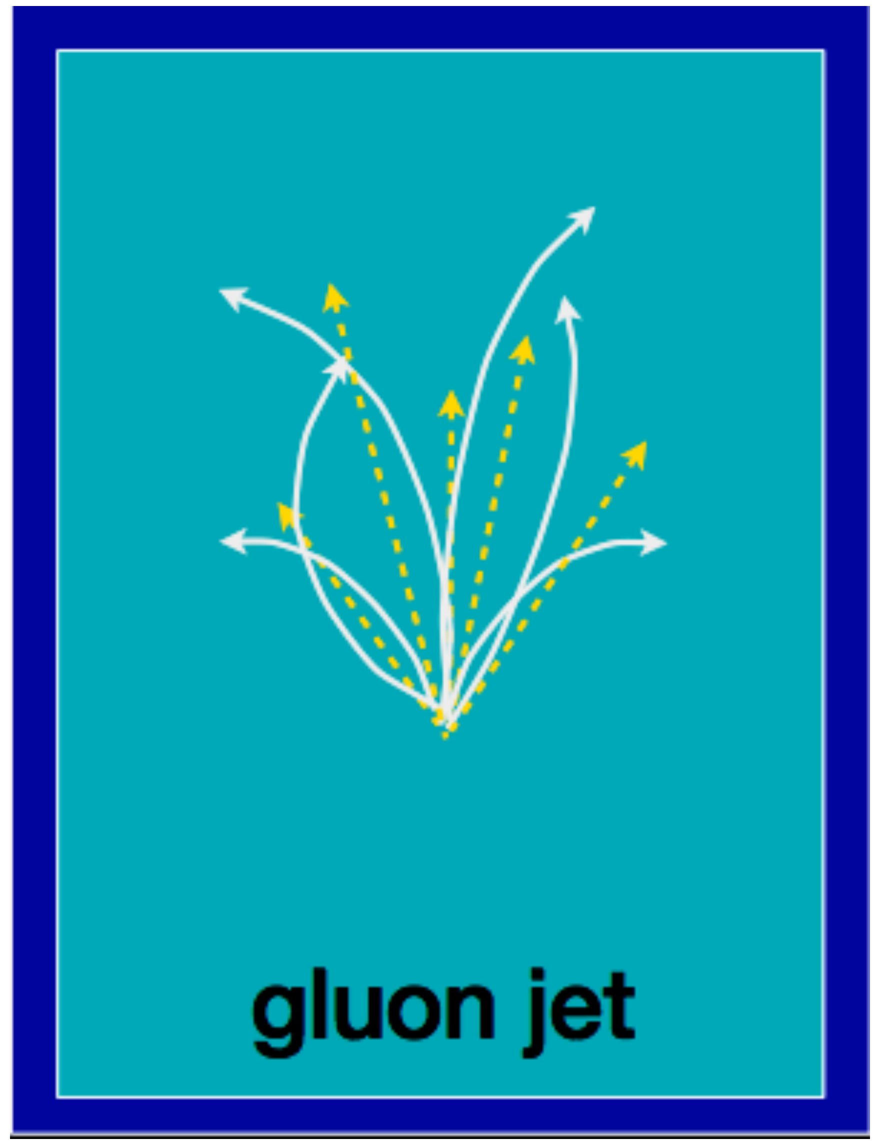
- After "particle-flow reconstruction," can think of event as a collection of points in momentum space
- For jets (localized clusters of particles), dimensionality $(N_{\text{particles}} \sim 100,\ 4+M)$
- Variable jet length requires:
  - Preprocessing into another rep. (tab. data, jet images, …)
  - Truncation to fixed size
  - Graph NN

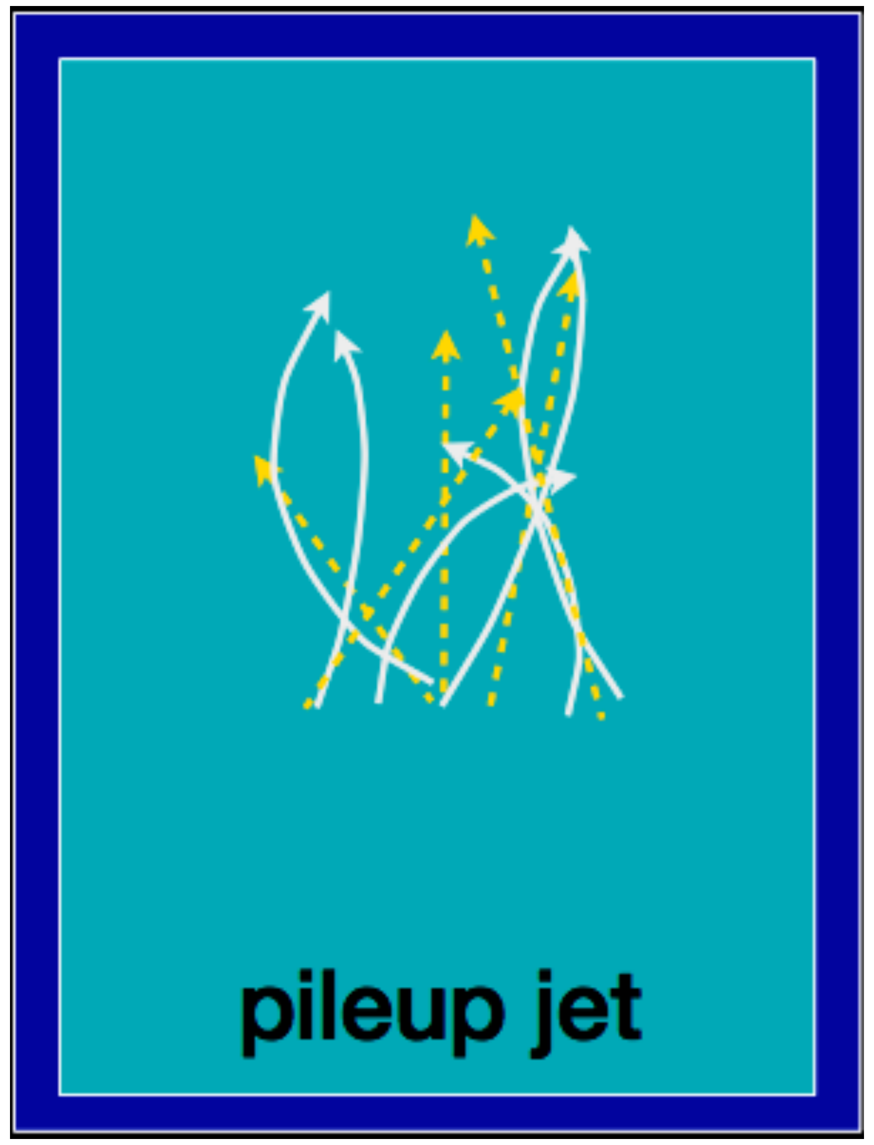


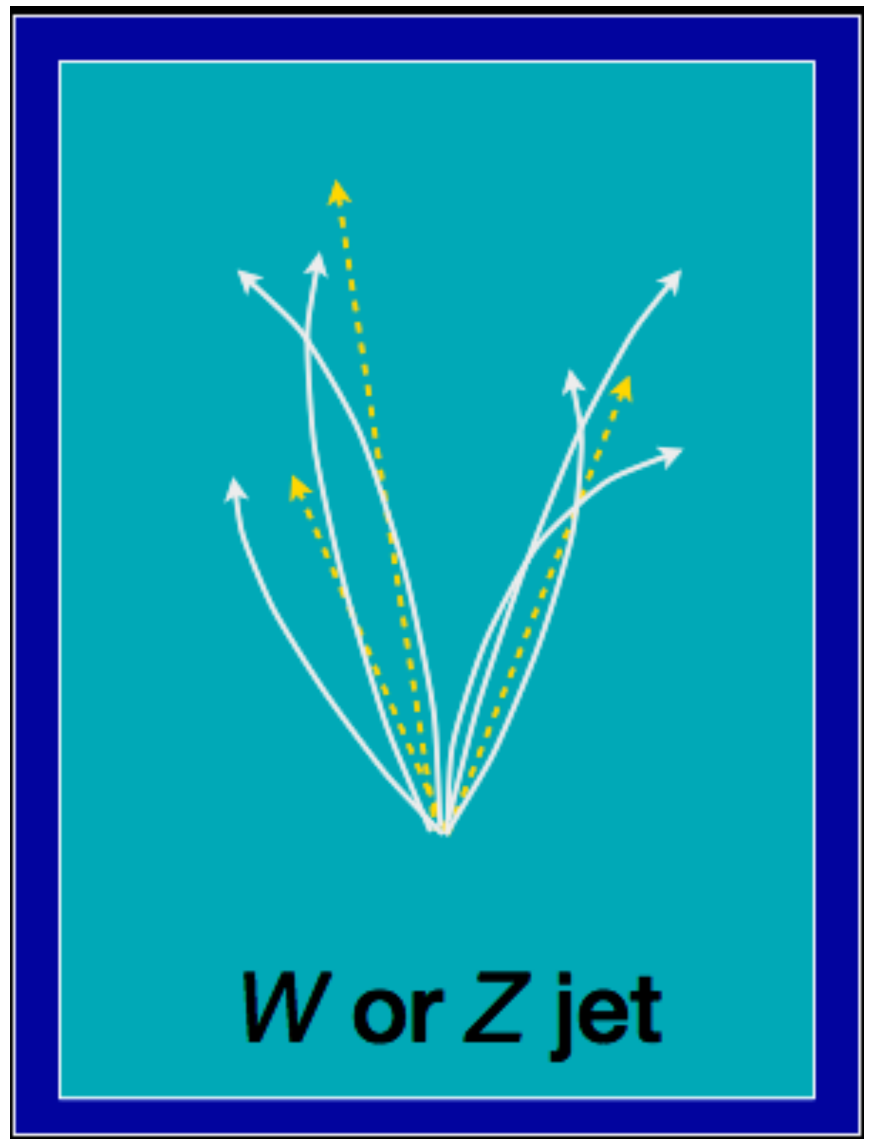
TASK: JET CLASSIFICATION

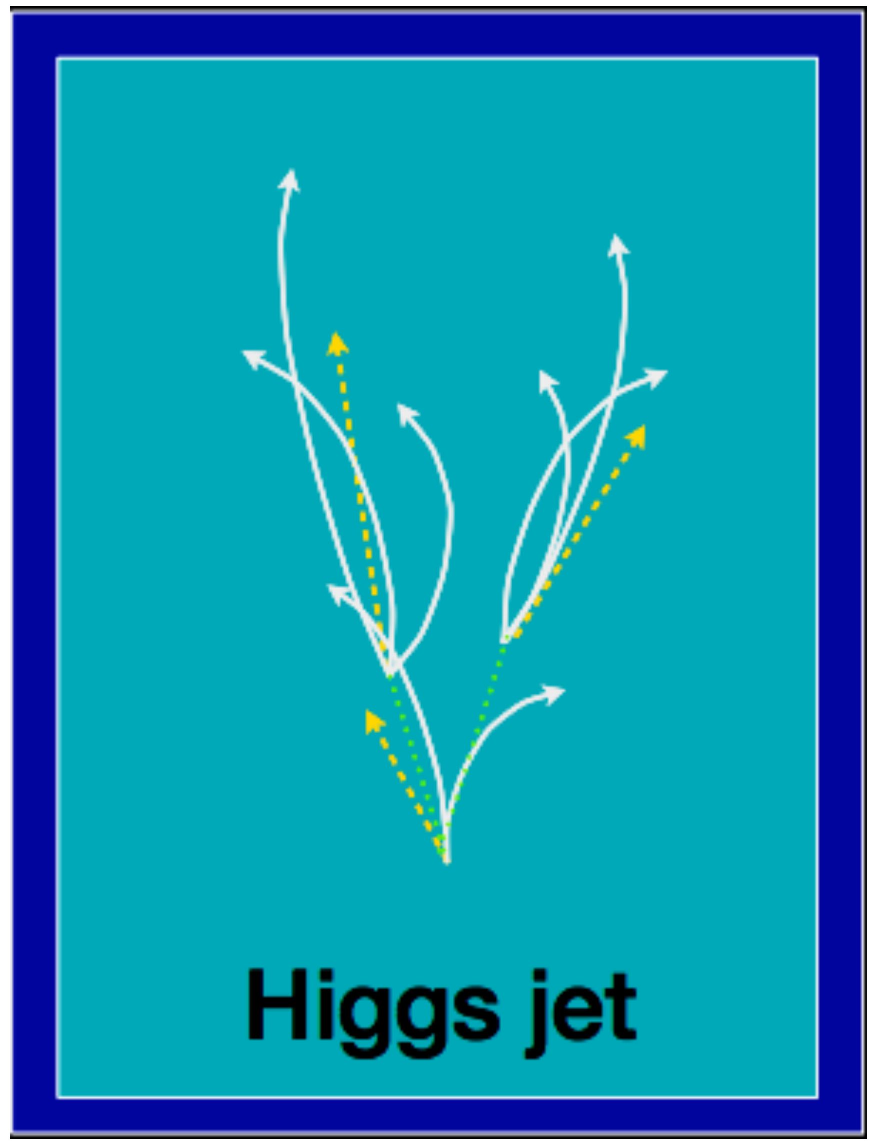

u,*d* or *s* jet

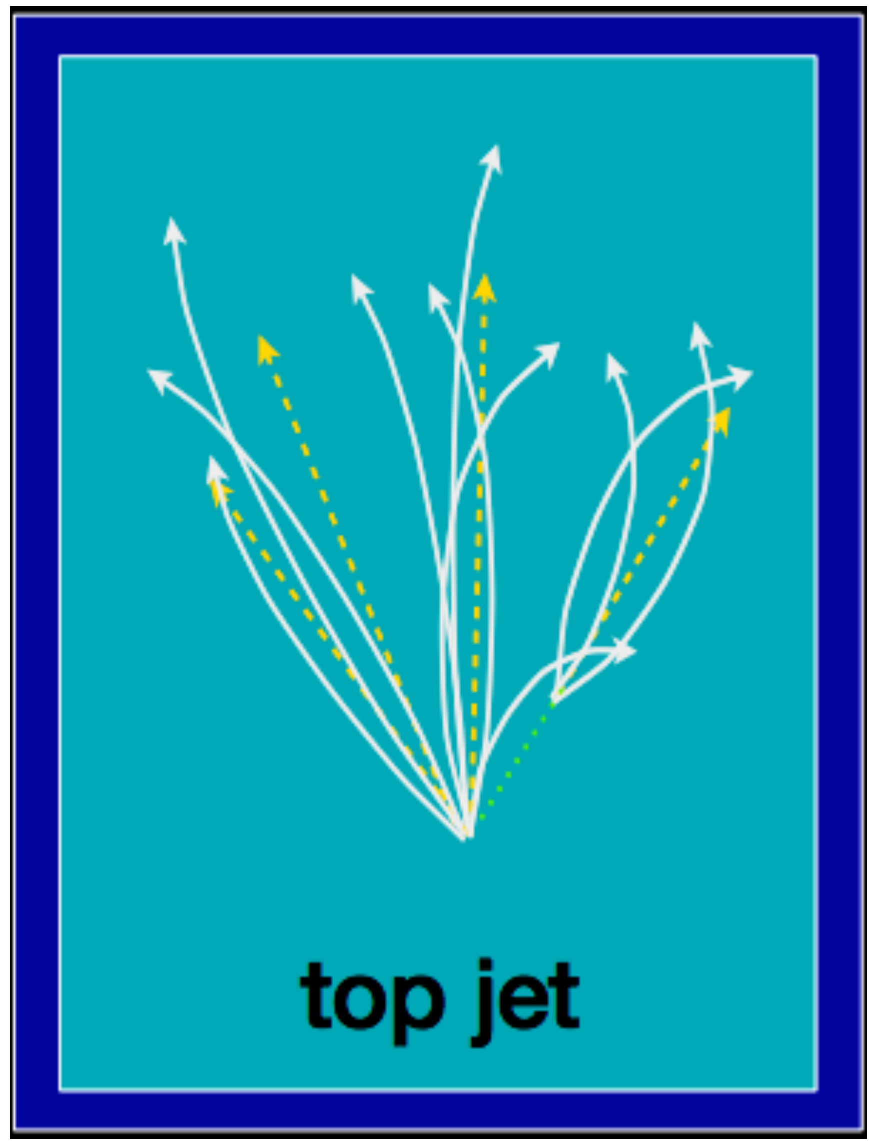

c or *b* jet


gluon jet


pileup jet


W or *Z* jet


Higgs jet

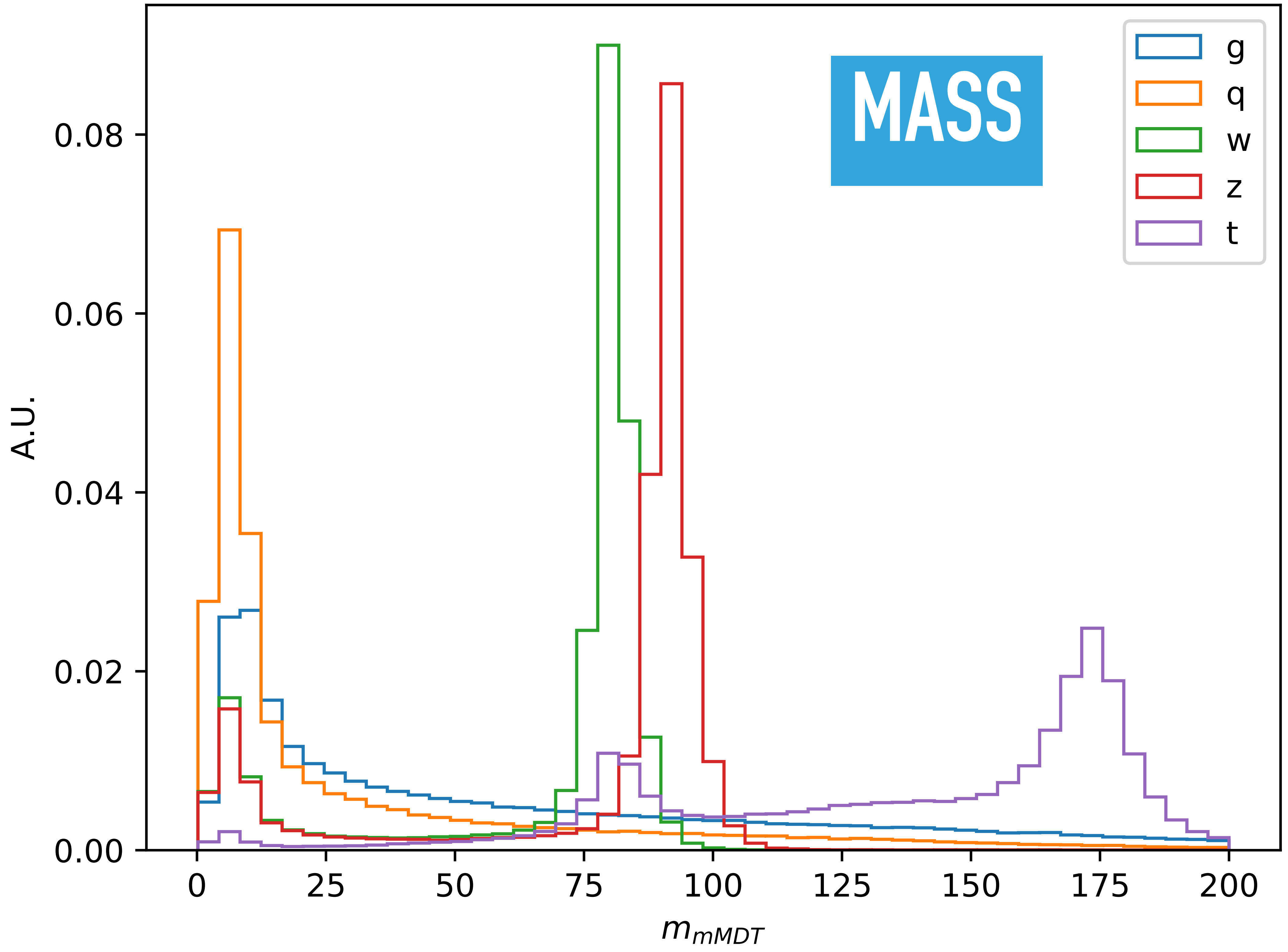

top jet

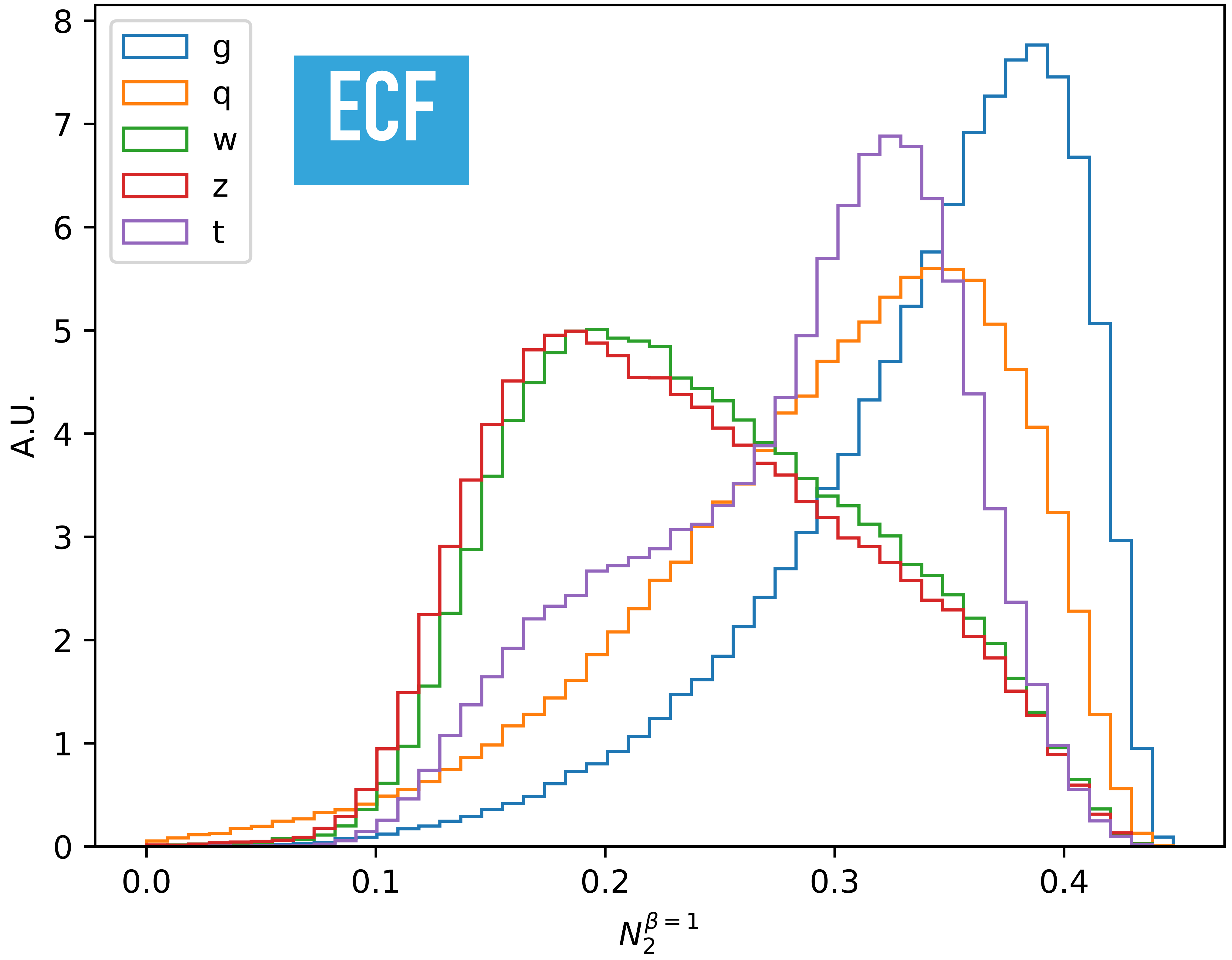
TABULAR DATA: JET SUBSTRUCTURE VARIABLES

- Tabular data: use physics knowledge to preprocess jet information into a set of high-level features
- Substructure variable:
  - jet mass
  - energy correlation functions, e.g. $N_2^{\beta=1} = {}_2e_3^{\beta=1}/({}_1e_3^{\beta=1})^2$

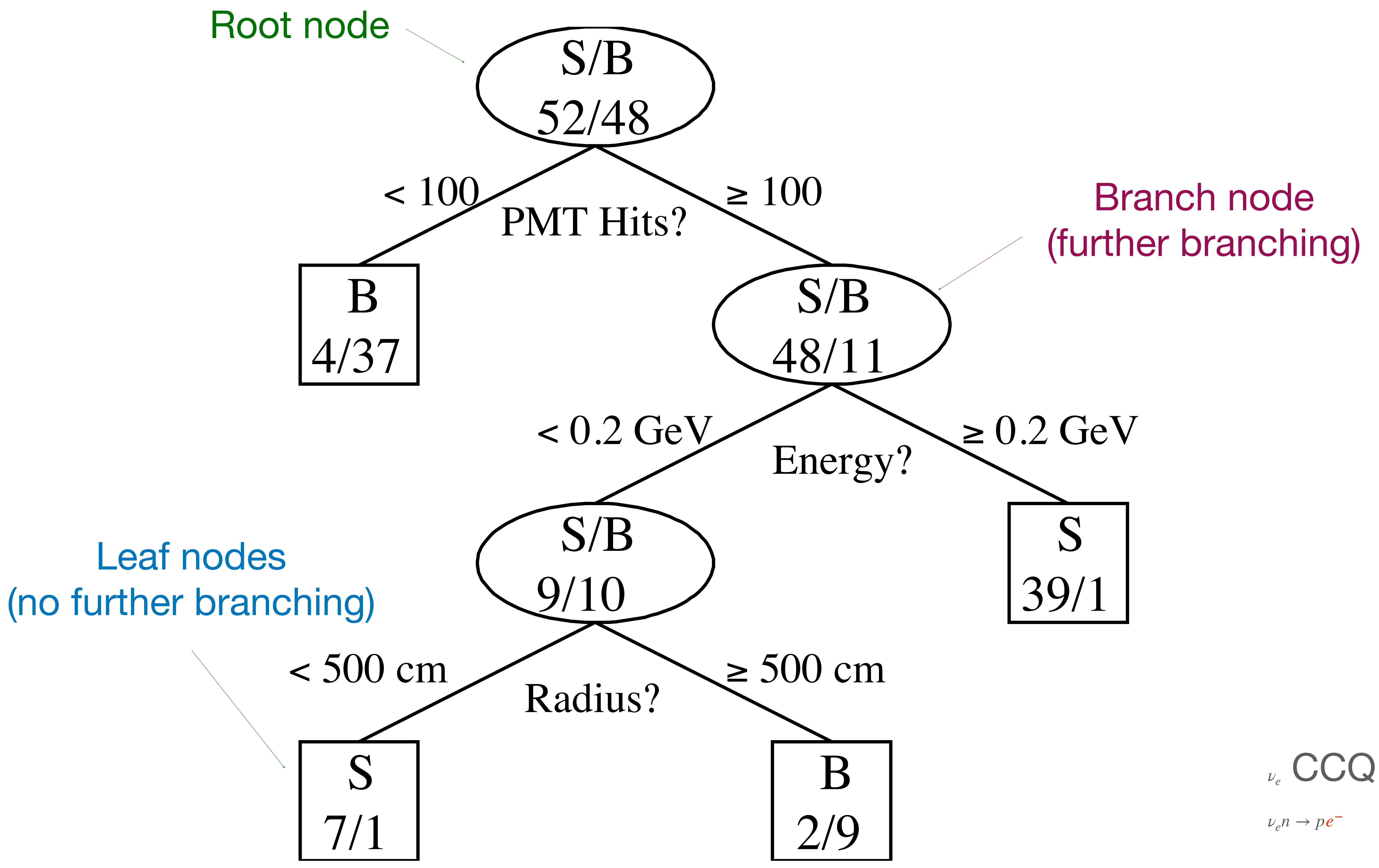
$$_1e_3^\beta = \sum_{1\leq i<j<k\leq n_J} z_i z_j z_k \min\{\Delta R_{ij}^\beta, \Delta R_{ik}^\beta, \Delta R_{jk}^\beta\}$$

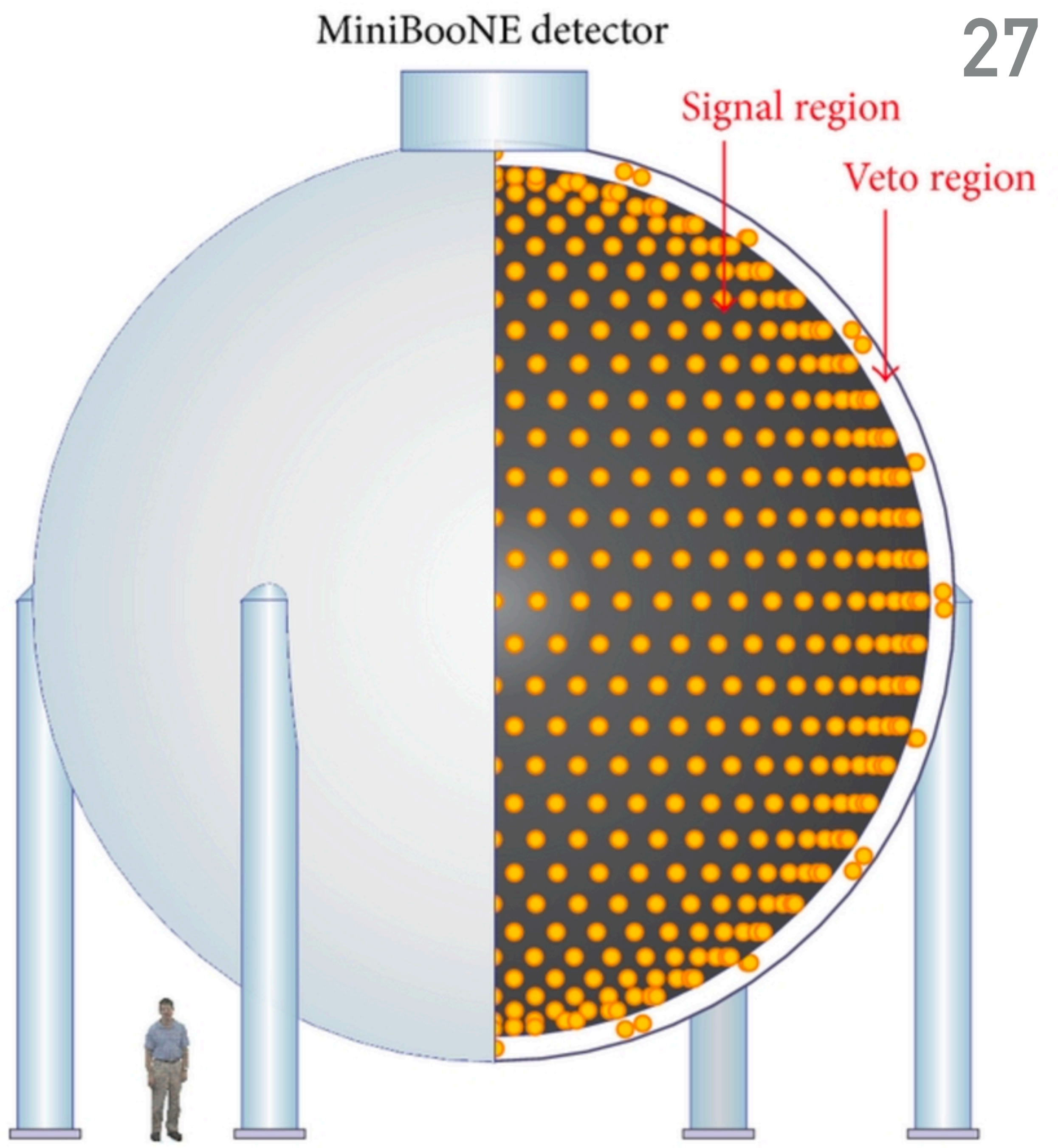
$$_2e_3^\beta = \sum_{1\leq i<j<k\leq n_J} z_i z_j z_k \min\{\Delta R_{ij}^\beta \Delta R_{ik}^\beta, \Delta R_{ij}^\beta \Delta R_{jk}^\beta, \Delta R_{ik}^\beta \Delta R_{jk}^\beta\}$$



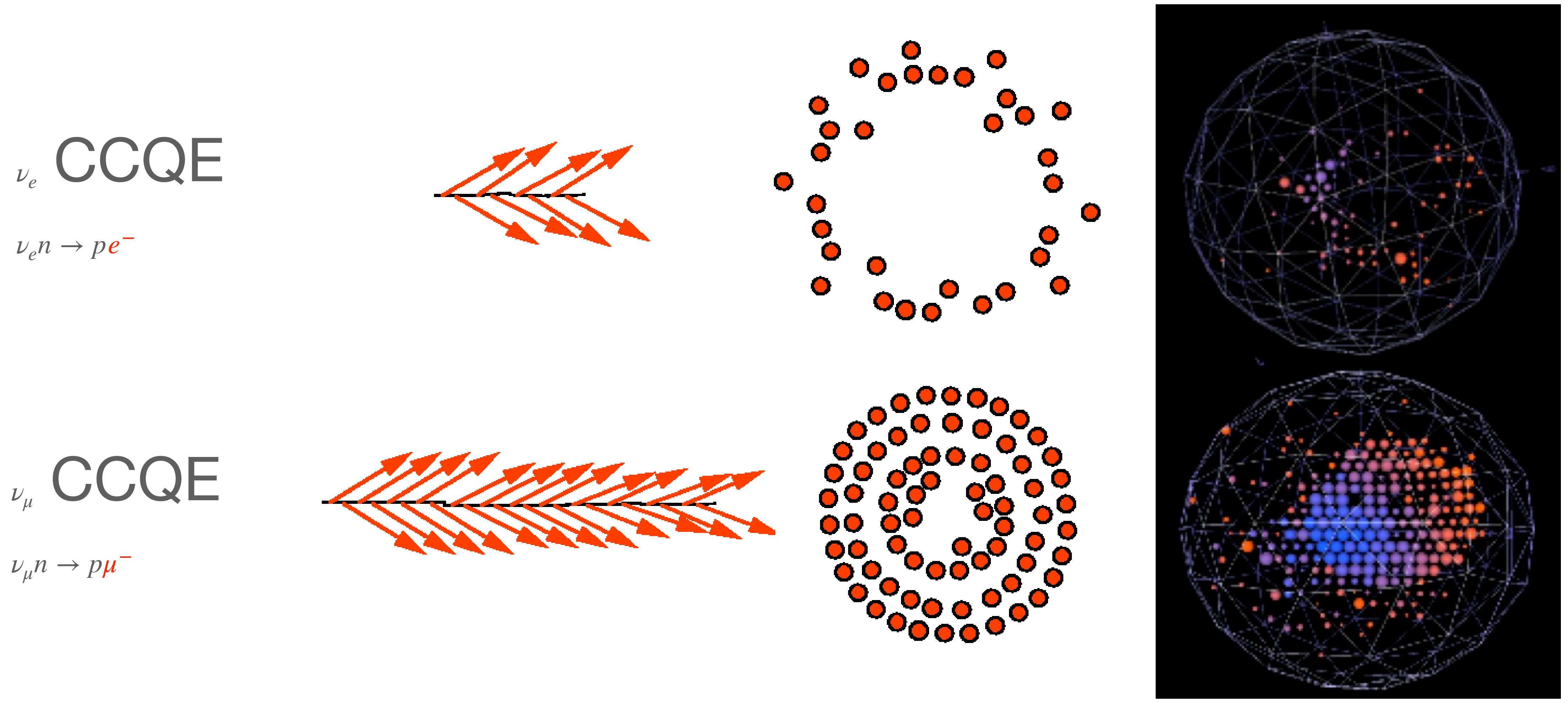


DECISION TREES





MiniBooNE: 1520 photomultiplier signals
Goal: separate ν_e and $\nu_\mu$events



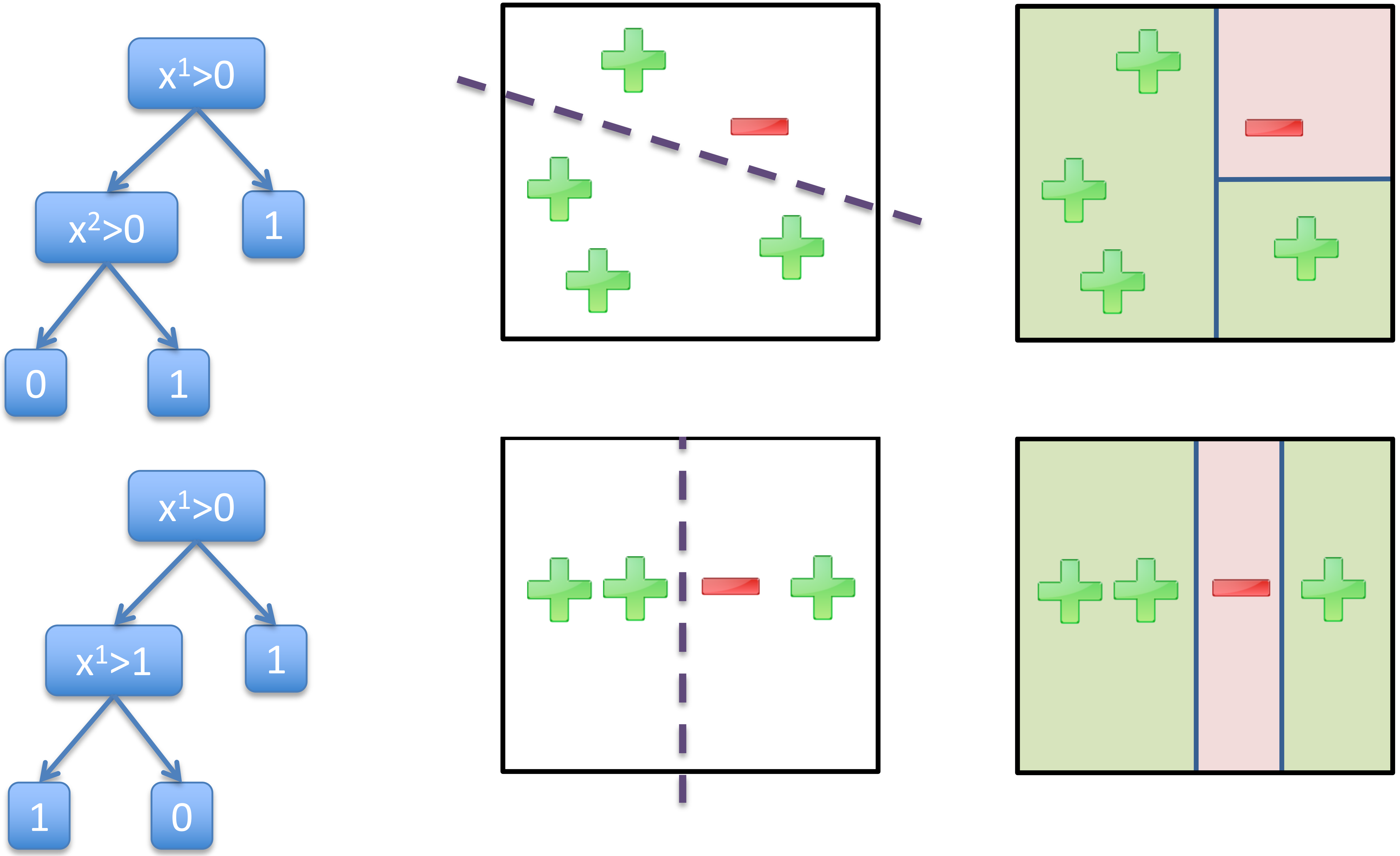
- Leaf nodes classify events as either signal (ν_e) or background (ν_μ)

DECISION TREES VS. LINEAR MODELS

- Decision trees are nonlinear models!
- Examples:

No linear model can achieve 0 error

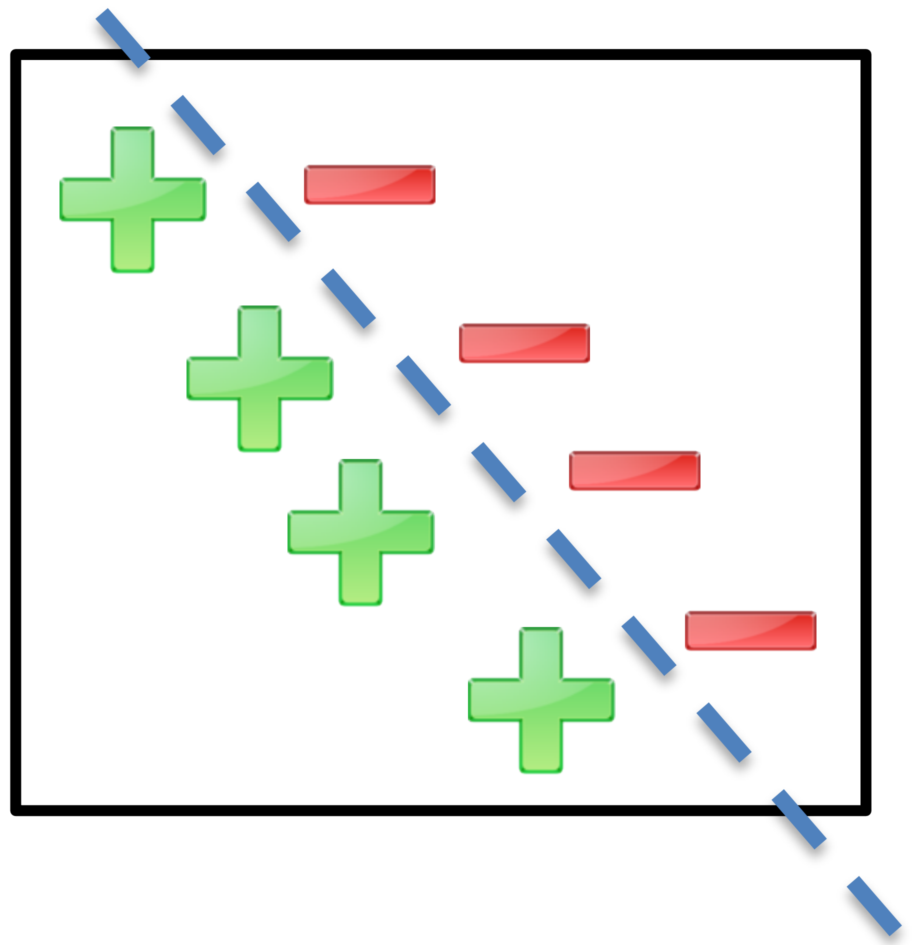
Simple decision tree can achieve 0 error



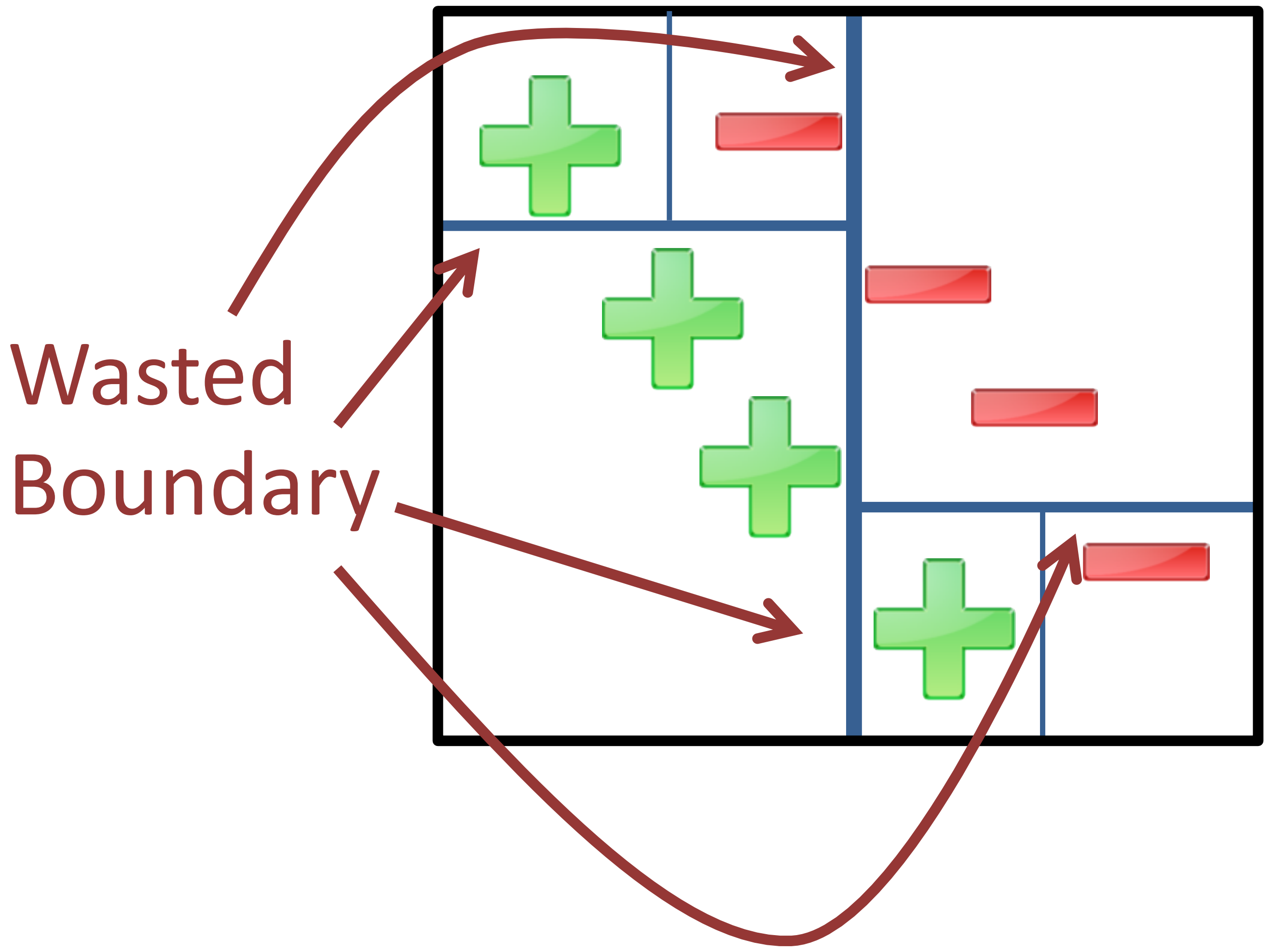
DECISION TREES VS. LINEAR MODELS

- Decision trees are axis-aligned!
- Example:

Simple linear SVM can easily find max margin

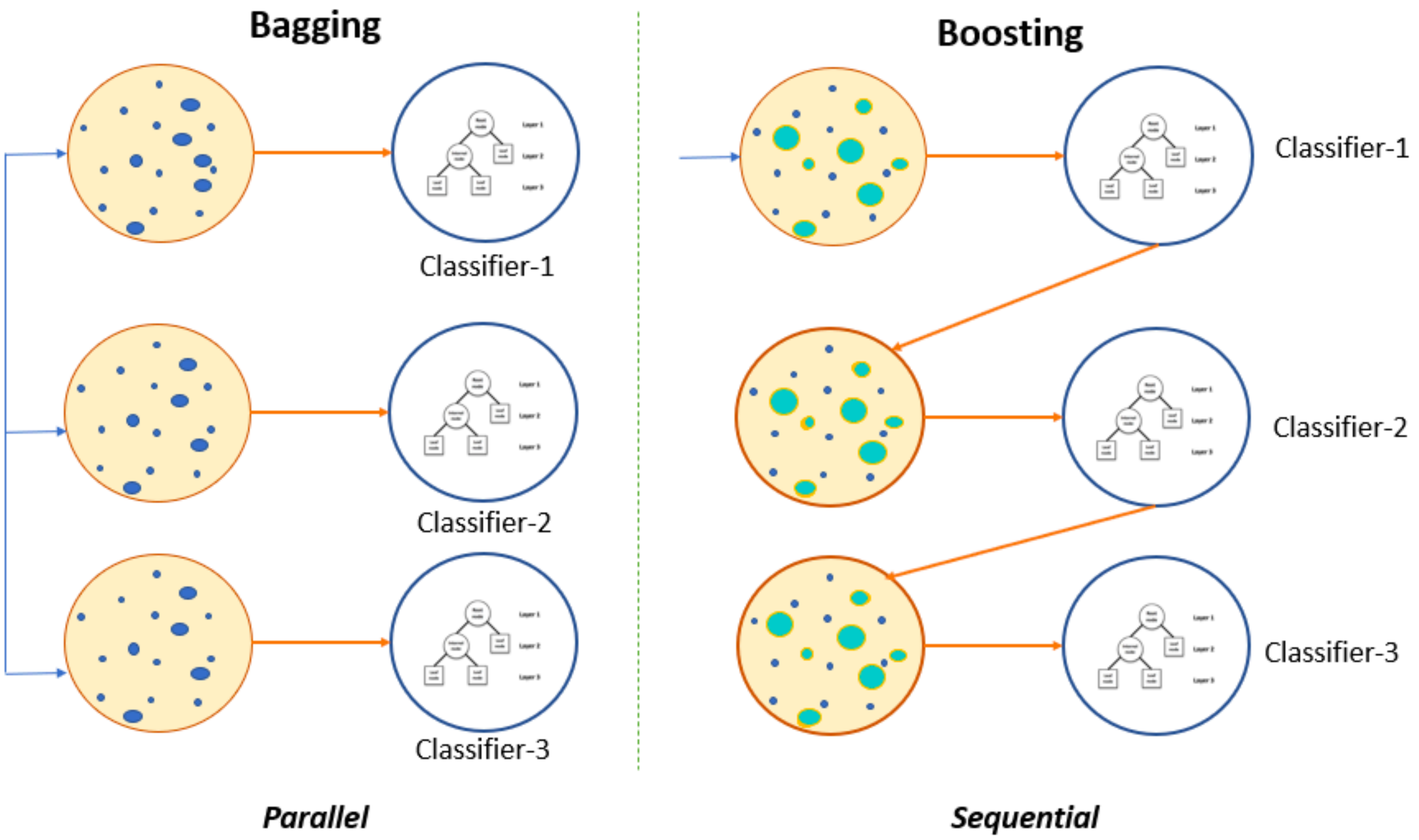


Decision trees require complex axis-aligned partitioning



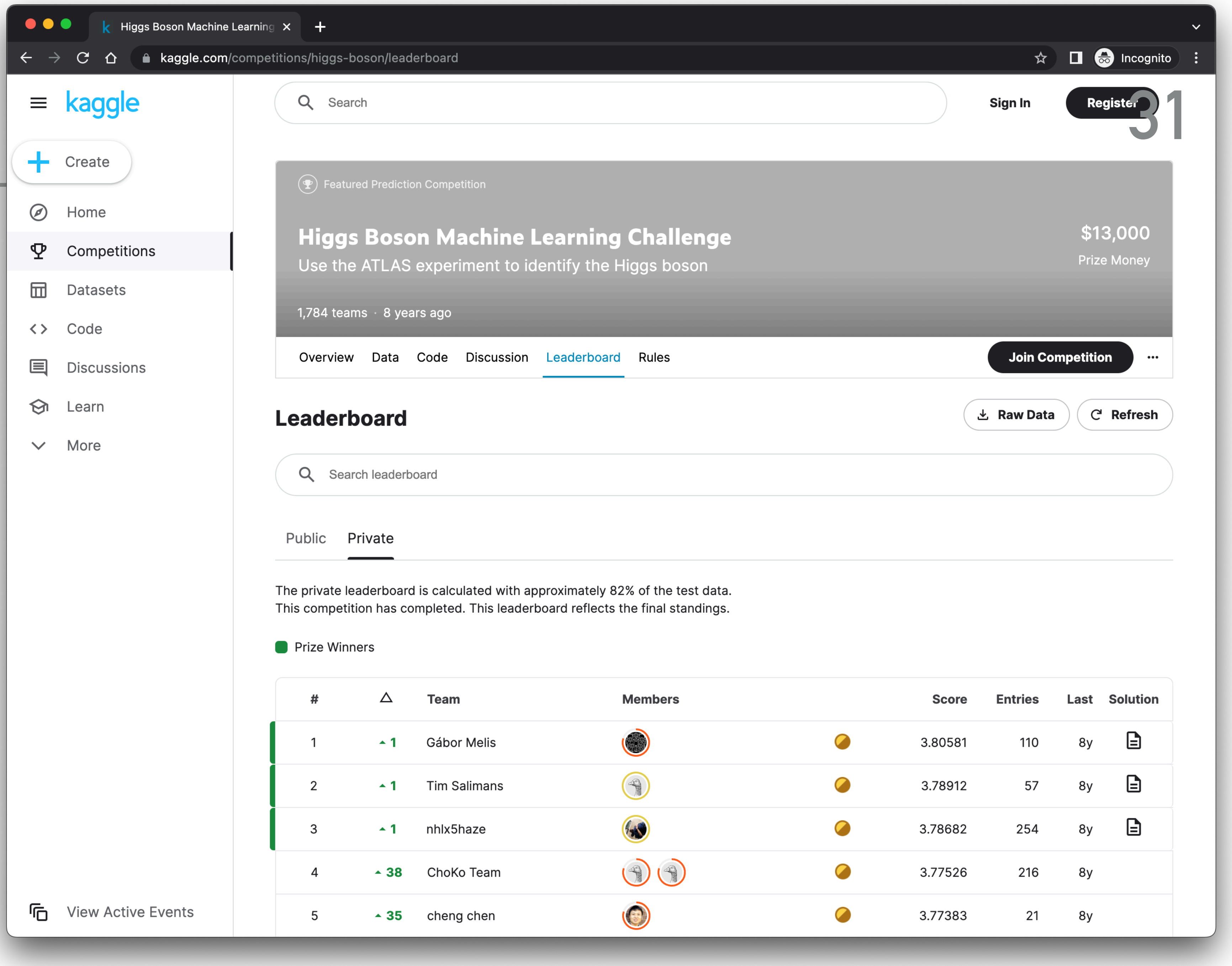
BAGGING VS. BOOSTING

- Bagging: reduce variance of weak learners
- Boosting: reduce bias of weak learners



BOOSTED DECISION TREES IN THE WILD

- 1st place in Kaggle Higgs Boson Machine Learning Challenge [kaggle.com/competitions/higgs-boson]
  - And many other uses at LHC, e.g. in Higgs boson discovery [10.1038/s41586-018-0361-2]
- Predicting critical temperature of a superconductor [10.1016/j.commatsci.2018.07.052]
- MiniBooNE neutrino event classification [10.1016/j.nima.2004.12.018]
- Observation of single top quark production at D0 [10.1103/PhysRevLett.103.092001]



Featured Prediction Competition

Higgs Boson Machine Learning Challenge

Use the ATLAS experiment to identify the Higgs boson

$13,000 Prize Money

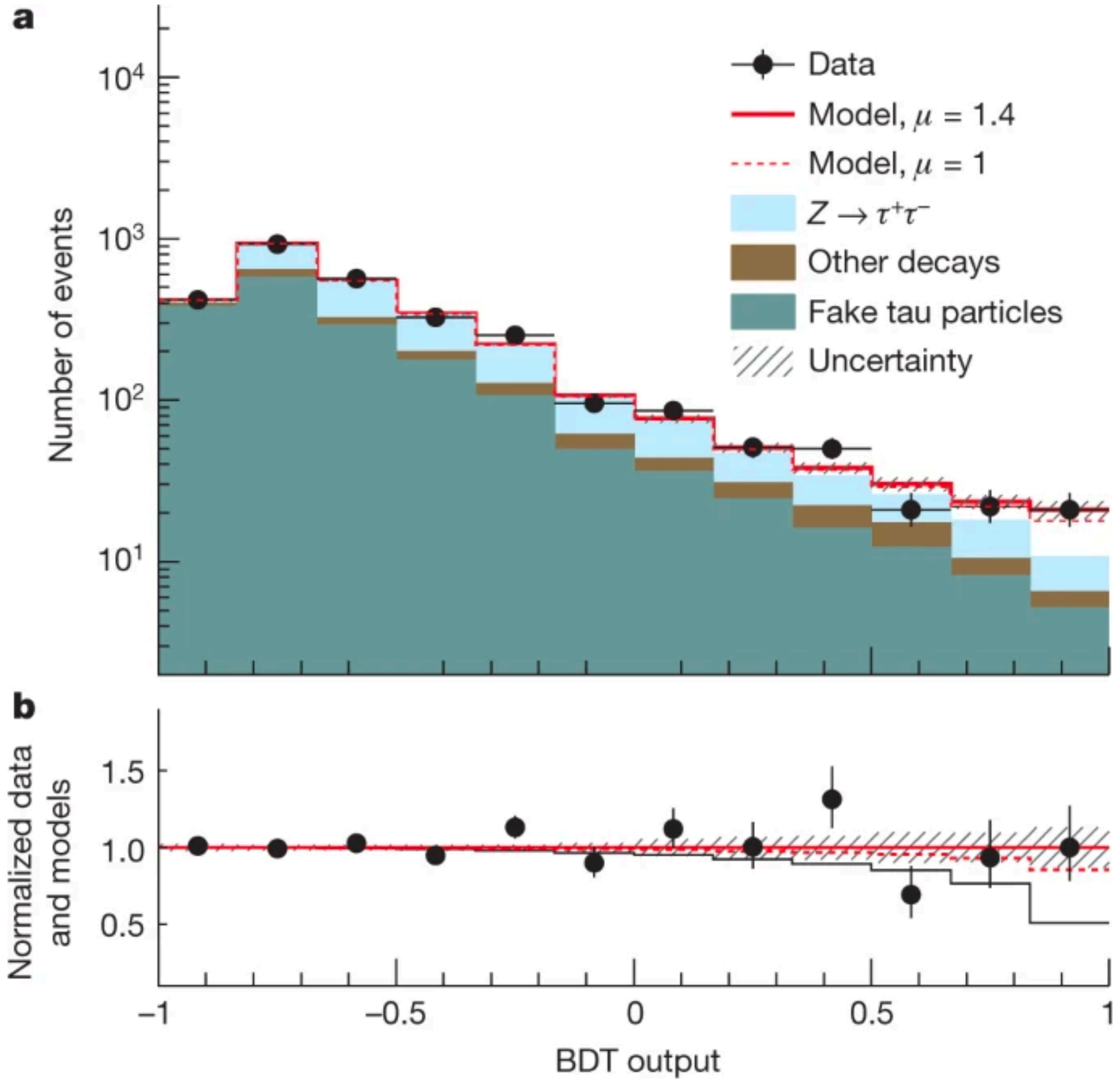
1,784 teams · 8 years ago

Leaderboard

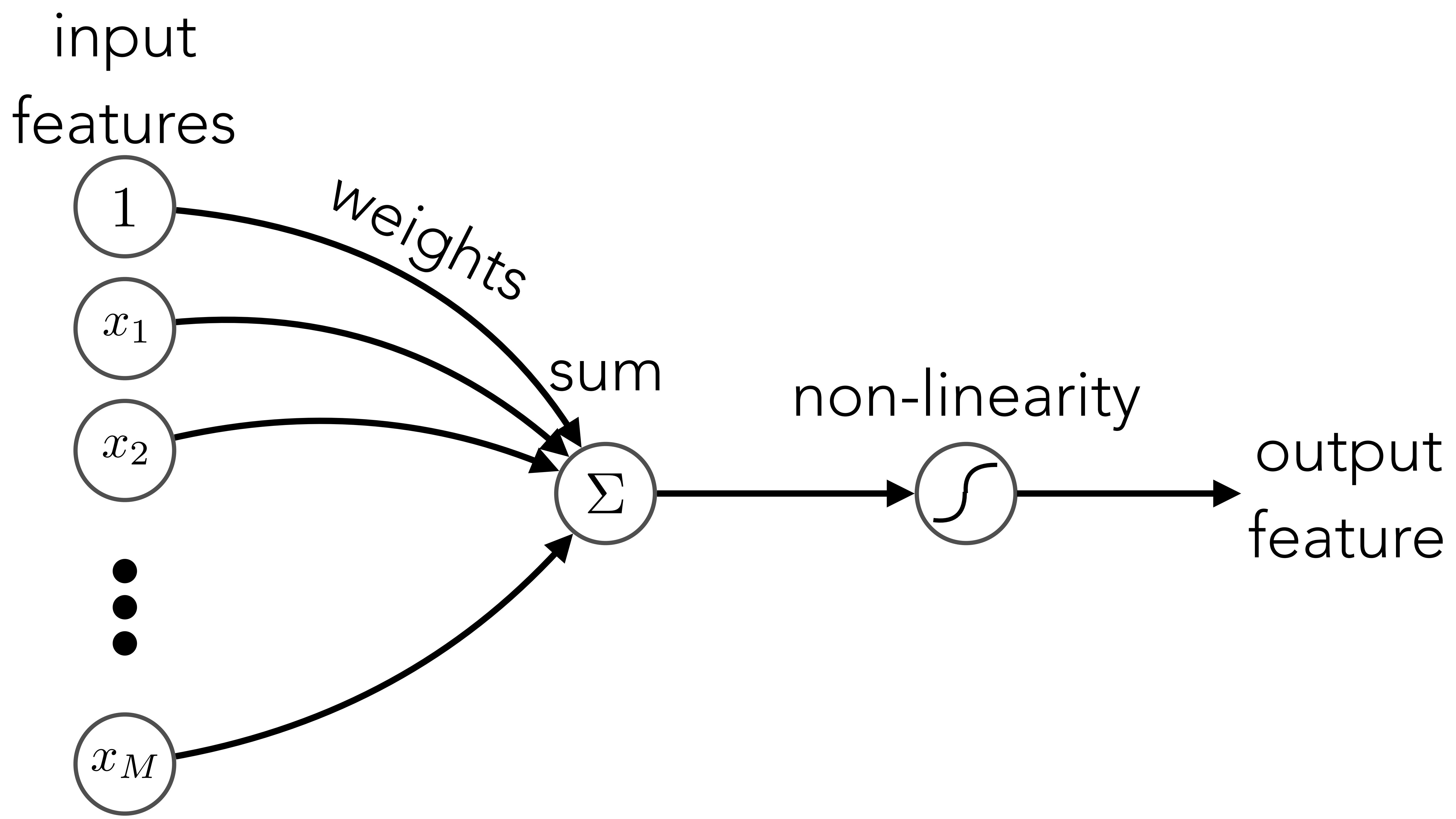
The private leaderboard is calculated with approximately 82% of the test data.
This competition has completed. This leaderboard reflects the final standings.

🟩 Prize Winners

#	△	Team	Members		Score	Entries	Last	Solution
1	▲ 1	Gábor Melis		●	3.80581	110	8y	🗎
2	▲ 1	Tim Salimans		●	3.78912	57	8y	🗎
3	▲ 1	nhlx5haze		●	3.78682	254	8y	🗎
4	▲ 38	ChoKo Team		●	3.77526	216	8y	
5	▲ 35	cheng chen		●	3.77383	21	8y	



ONE ARTIFICIAL NEURON



artificial neuron: *weighted sum and non-linearity*

bias

input features

$$s = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_M x_M = \mathbf{w}^\intercal \mathbf{x}$$

sum

weights

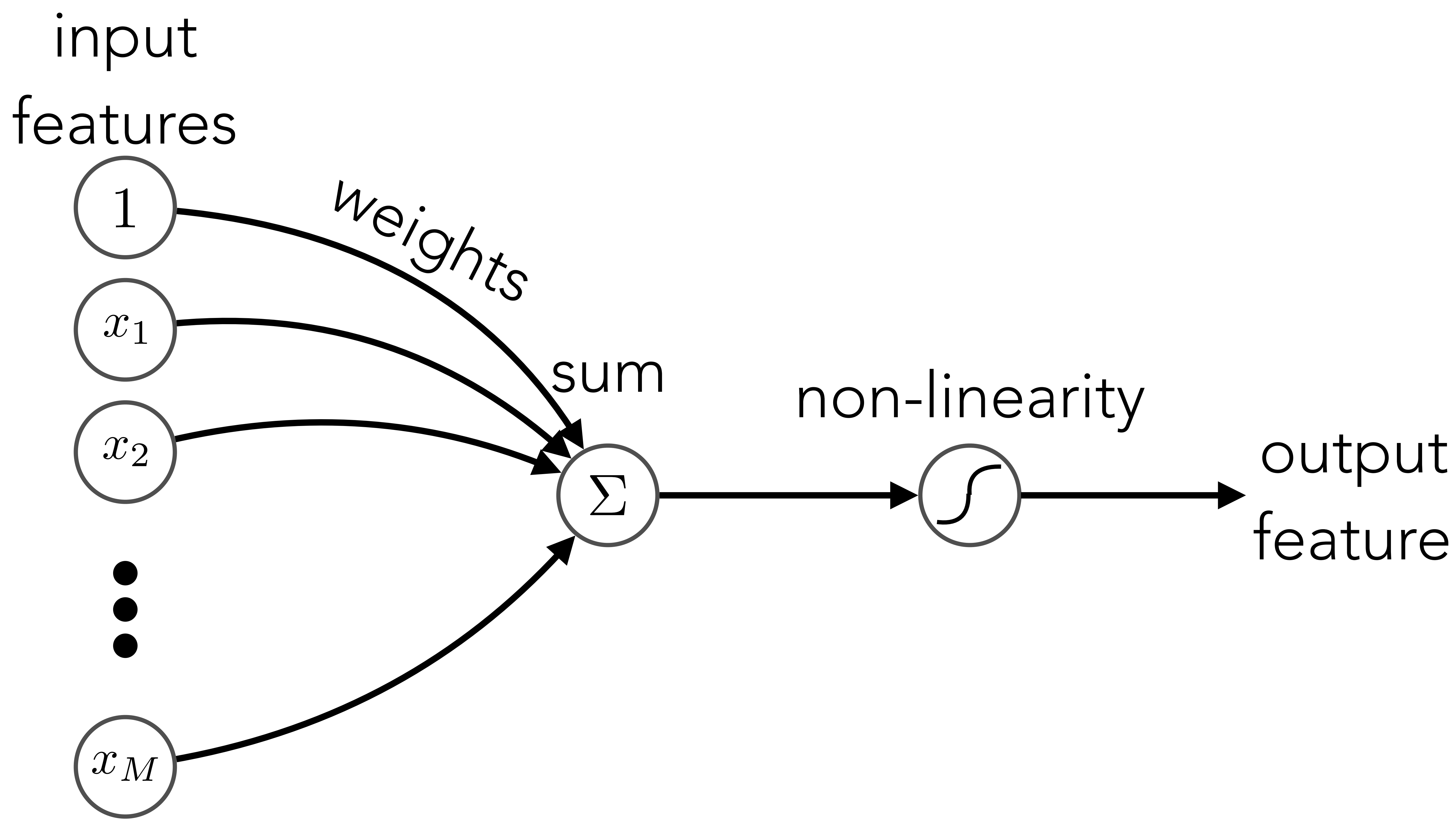
$$h = \sigma(s)$$

output feature

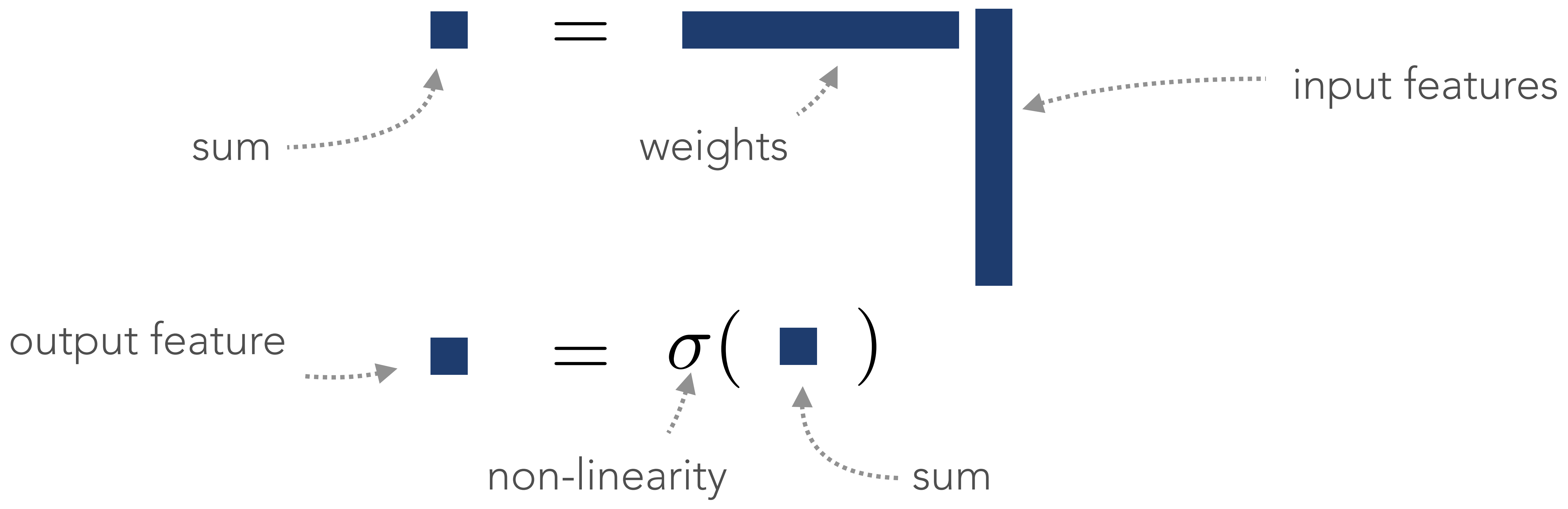
non-linearity

sum

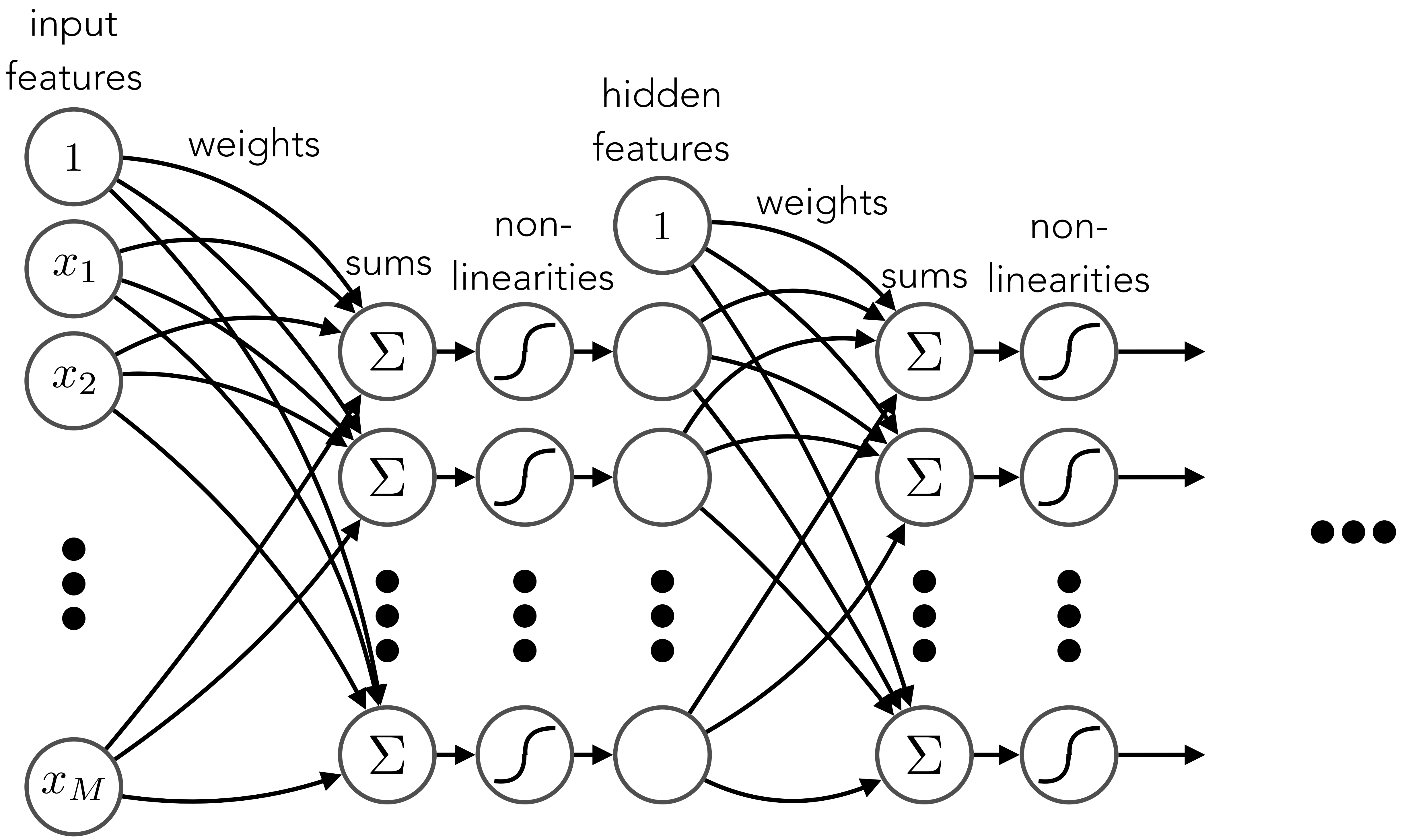
ONE ARTIFICIAL NEURON



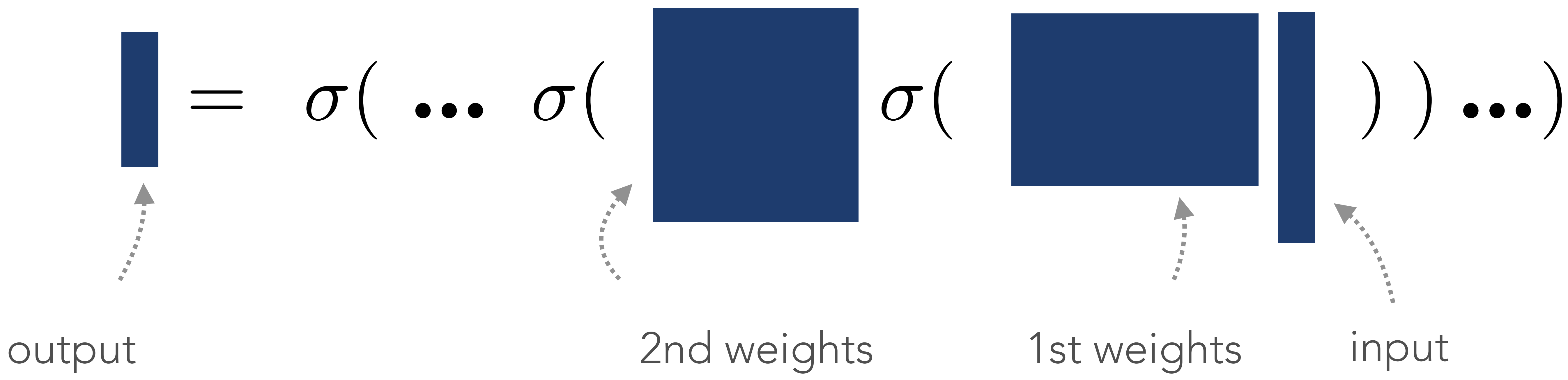
artificial neuron: *weighted sum and non-linearity*



LAYERS IN A NETWORK



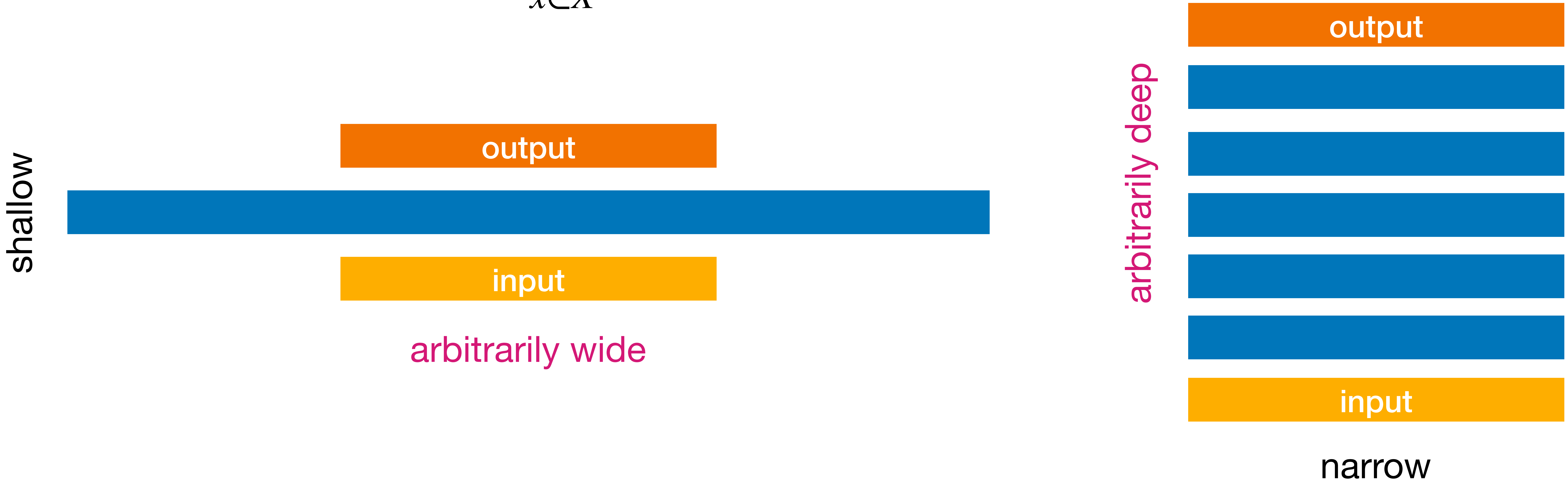
network: *sequence of parallelized weighted sums and non-linearities*



NNS ARE UNIVERSAL FUNCTION APPROXIMATORS

Universal approximation theorem (informal). Given a function $y = f(x)$ and an $\epsilon > 0$, there exists a deep network $y = f_w(x)$ (of arbitrary width or depth) such that:

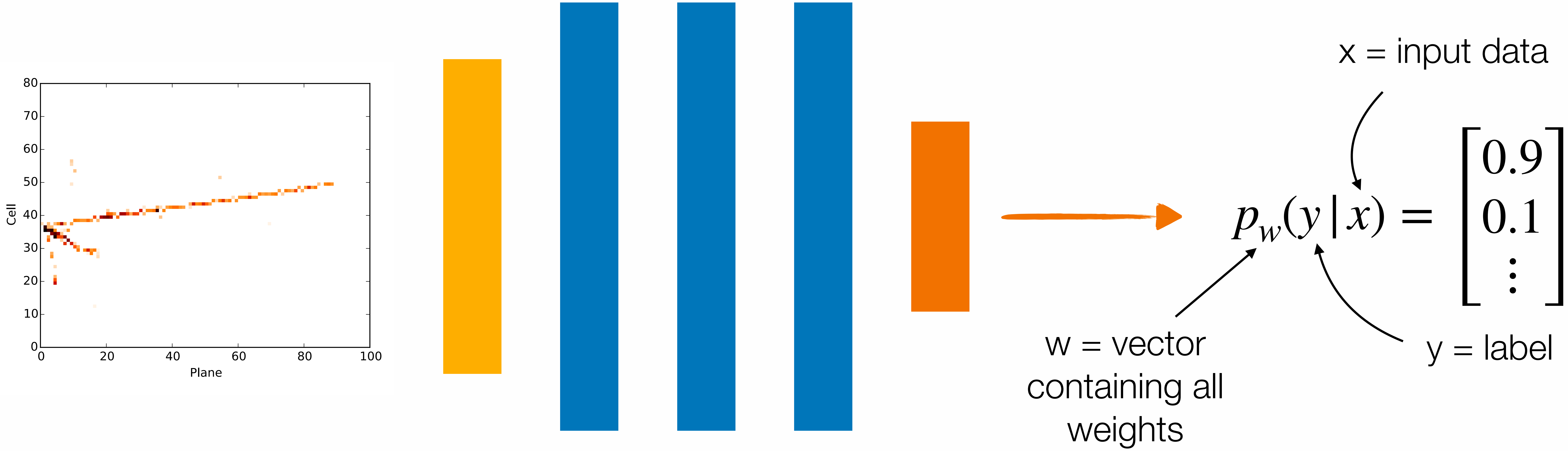
$$\sup_{x \in X} \|f(x) - f_w(x)\| < \epsilon$$



Note: This means that a network can *represent* any function, not that it can learn it! The "amount" of function a given network can represent is often called its expressive power.

TRAINING A NN

We train deep networks using Maximum Likelihood Estimation (MLE): The last layer of a DNN is a softmax that outputs probabilities over classes:



x = input data

$$p_w(y \mid x) = \begin{bmatrix} 0.9 \\ 0.1 \\ \vdots \end{bmatrix}$$

w = vector containing all weights

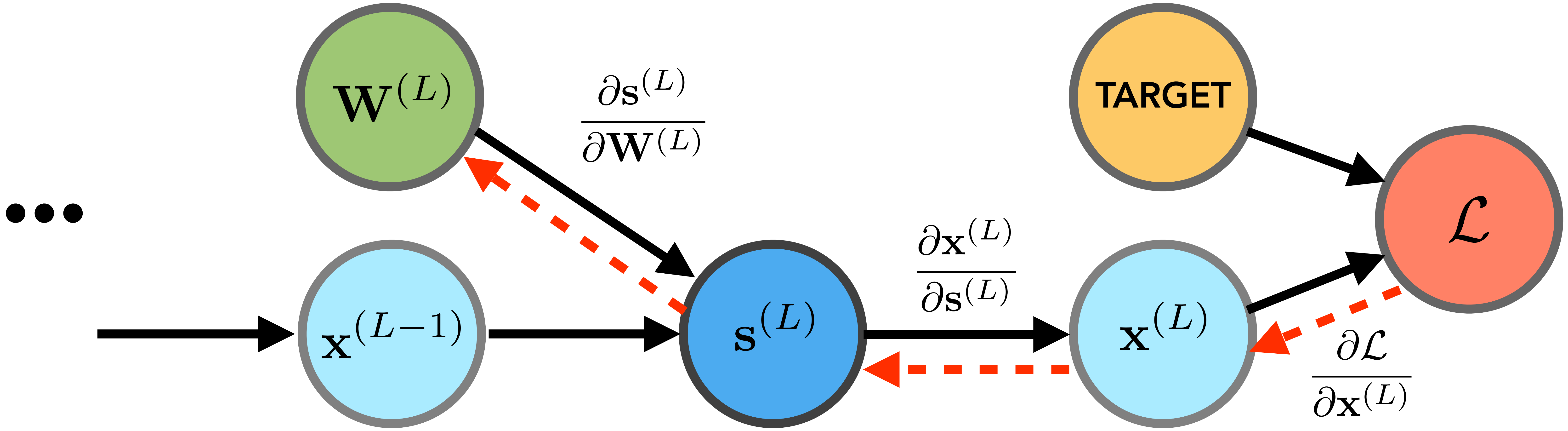
y = label

We train the weights w to maximize the log-likelihood of the data under our model:

$$L(w) = -\frac{1}{N}\sum_{i=1}^{N} \log p_w(y_i \mid x_i)$$

Negative log-likelihood loss (cross-entropy loss)

BACKPROPAGATION FOR NNS



...
$\mathbf{W}^{(L)}$
$\frac{\partial \mathbf{s}^{(L)}}{\partial \mathbf{W}^{(L)}}$
TARGET
$\mathcal{L}$
$\mathbf{x}^{(L-1)}$
$\mathbf{s}^{(L)}$
$\frac{\partial \mathbf{x}^{(L)}}{\partial \mathbf{s}^{(L)}}$
$\mathbf{x}^{(L)}$
$\frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(L)}}$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(L)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(L)}} \frac{\partial \mathbf{x}^{(L)}}{\partial \mathbf{s}^{(L)}} \frac{\partial \mathbf{s}^{(L)}}{\partial \mathbf{W}^{(L)}}$$

depends on the
form of the loss

derivative of the
non-linearity

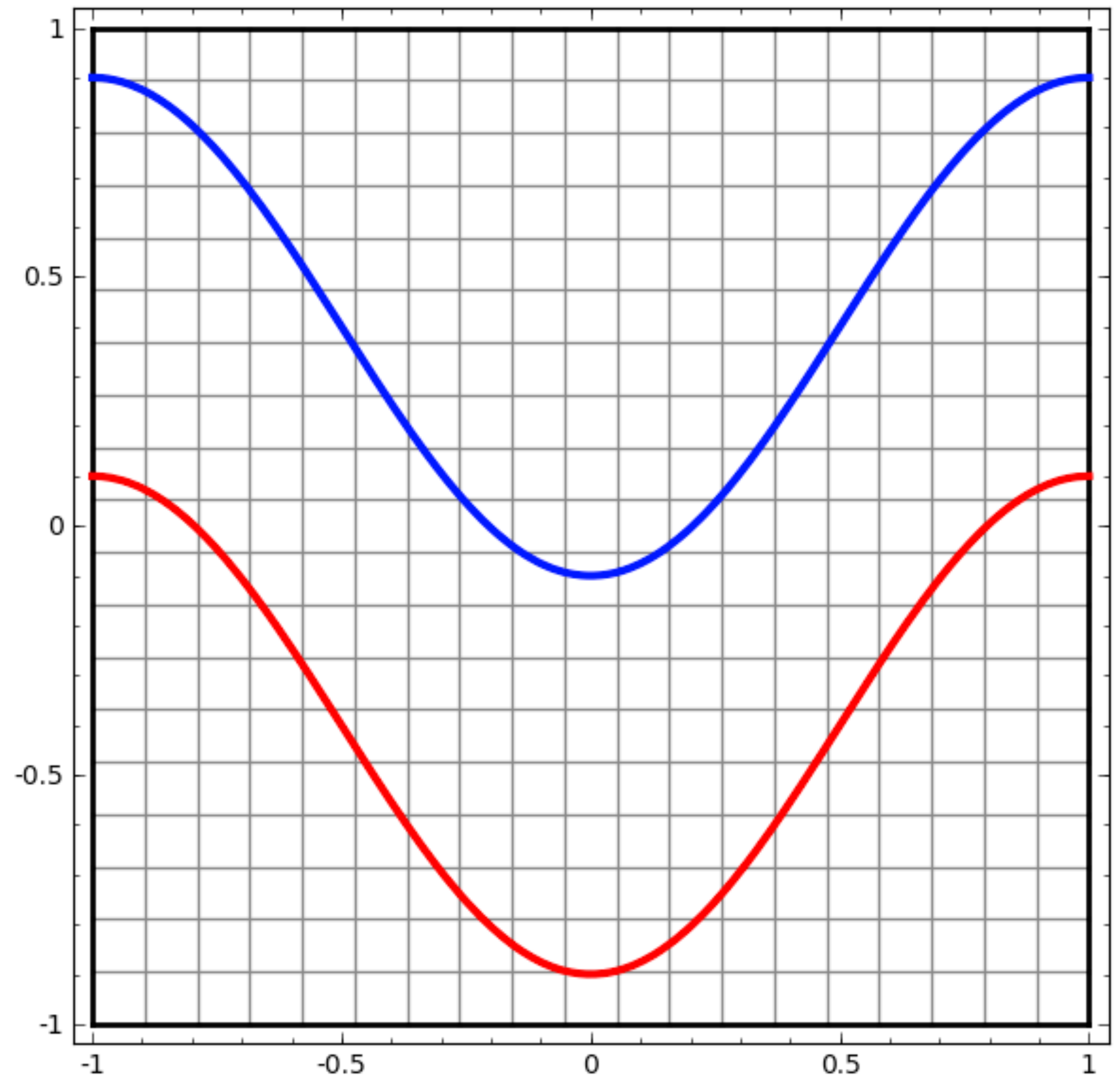
$$\frac{\partial}{\partial \mathbf{W}^{(L)}} (\mathbf{W}^{(L)\intercal} \mathbf{x}^{(L-1)})$$

$$= \mathbf{x}^{(L-1)\intercal}$$

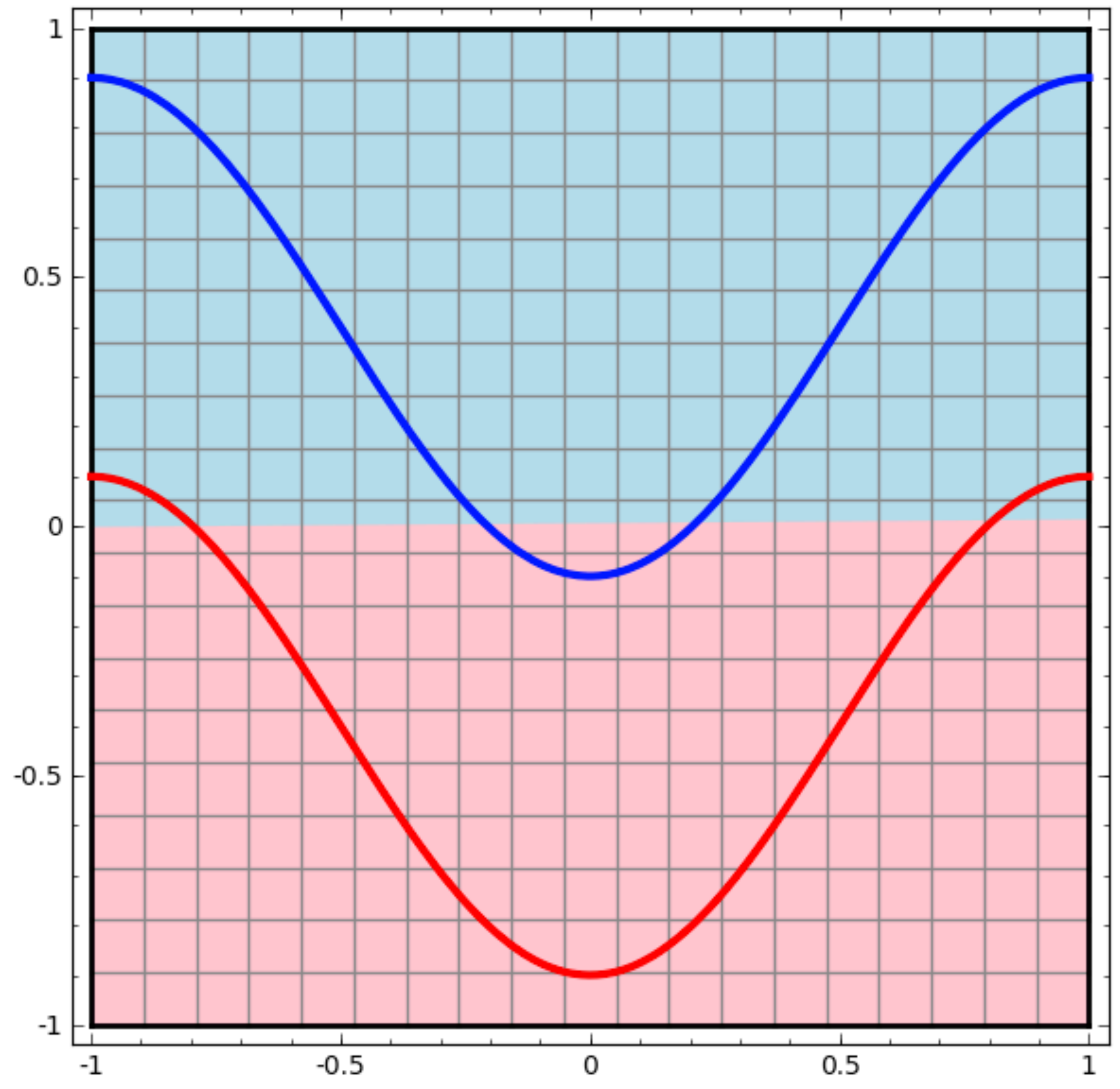
note $\nabla_{\mathbf{W}^{(L)}} \mathcal{L} \equiv \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(L)}}$ is notational convention

LINEAR MODELS & EMBEDDINGS

Data

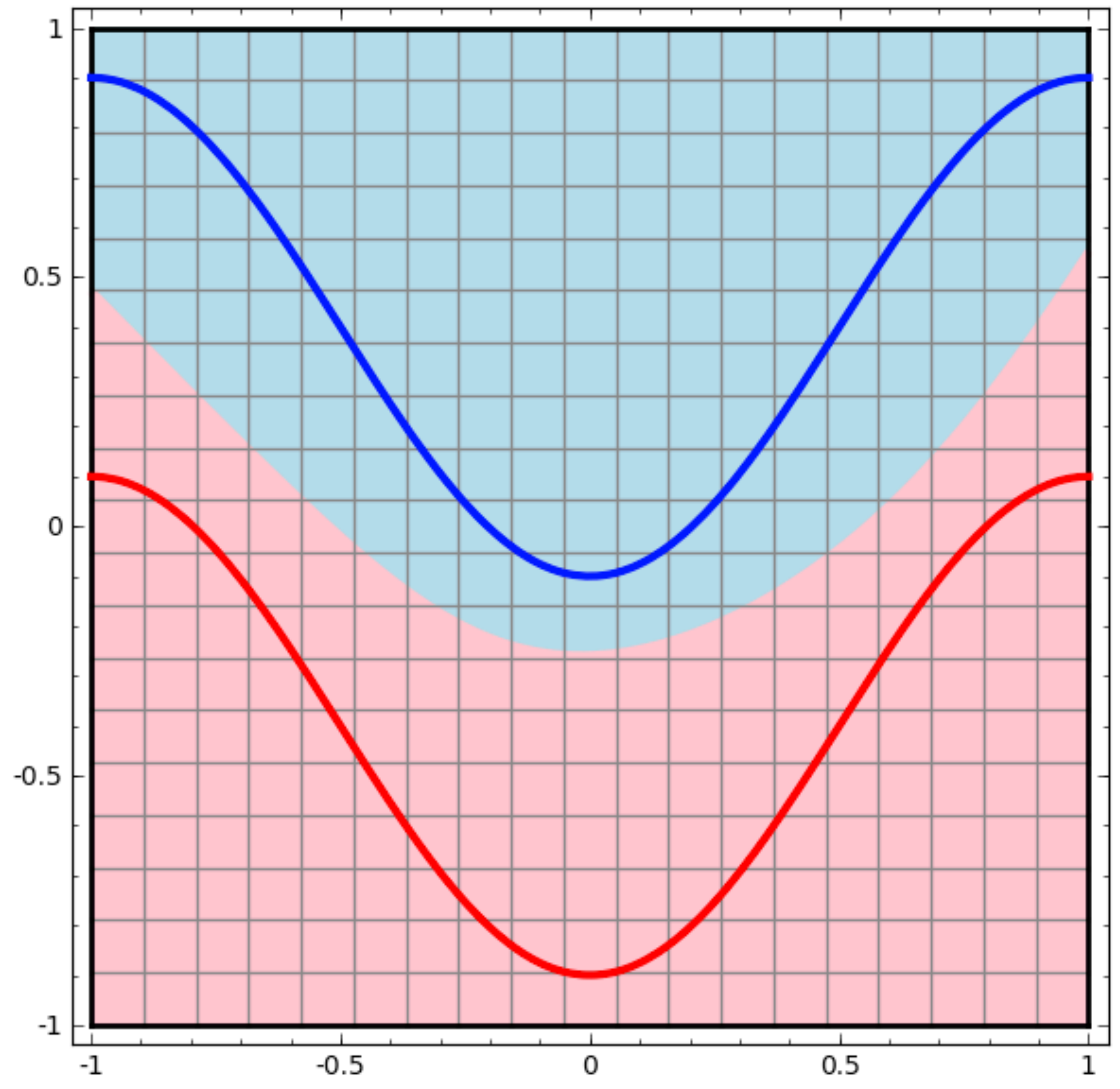


Linear classifier



$$y = \mathrm{softmax}(w^\intercal x)$$

Embedding + Linear classifier



$$y = \mathrm{softmax}(w^\intercal \phi(x))$$

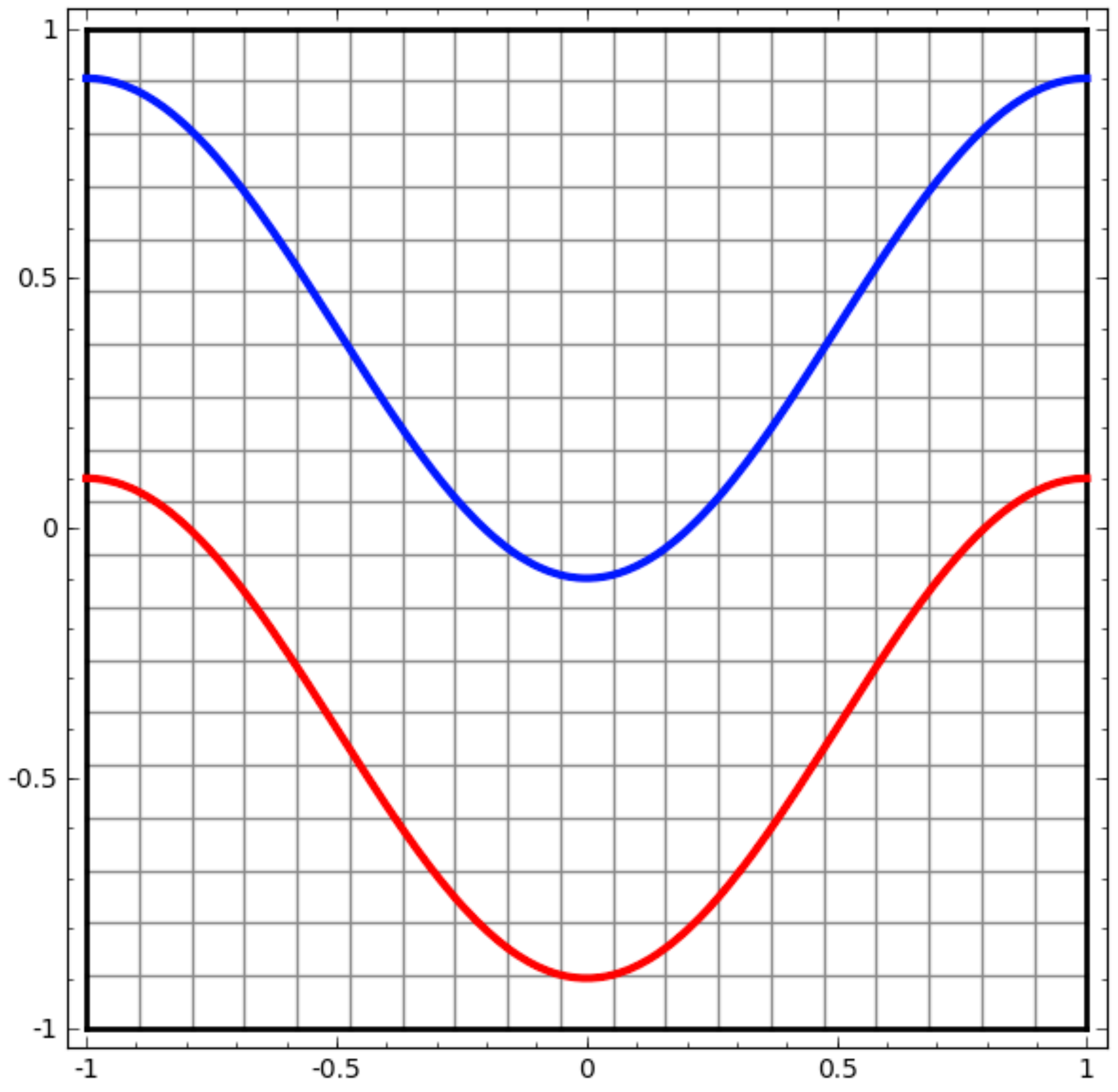
We have seen the polynomial embedding:

$$\phi(x) = (1, x, x^2, \ldots, x^n)$$

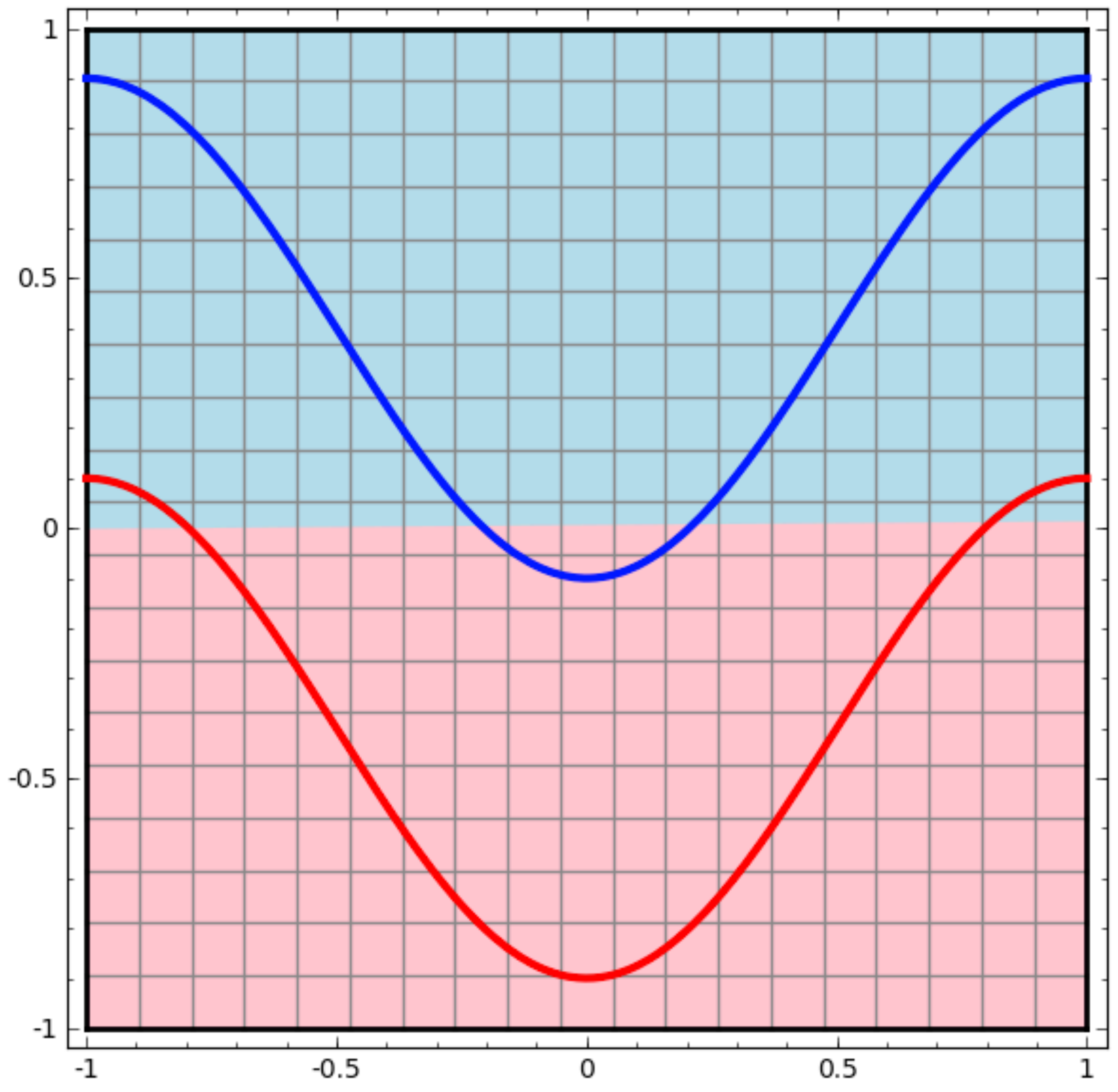
colah.github.io/posts/2014-03-NN-Manifolds-Topology

NEURAL NETWORKS & TOPOLOGY

Data

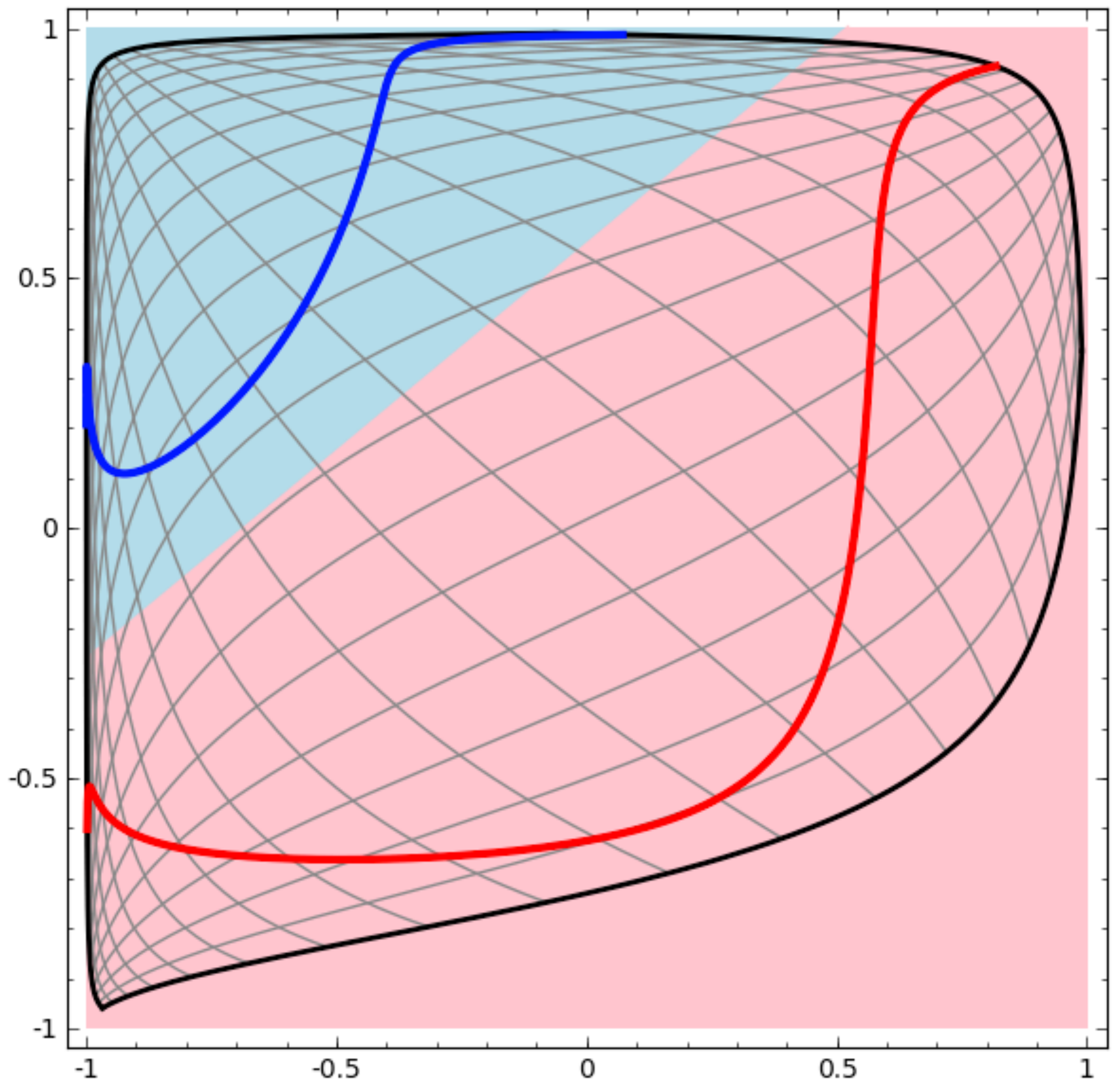


Linear classifier



$$y = \mathrm{softmax}(w^\intercal x)$$

2-layer network

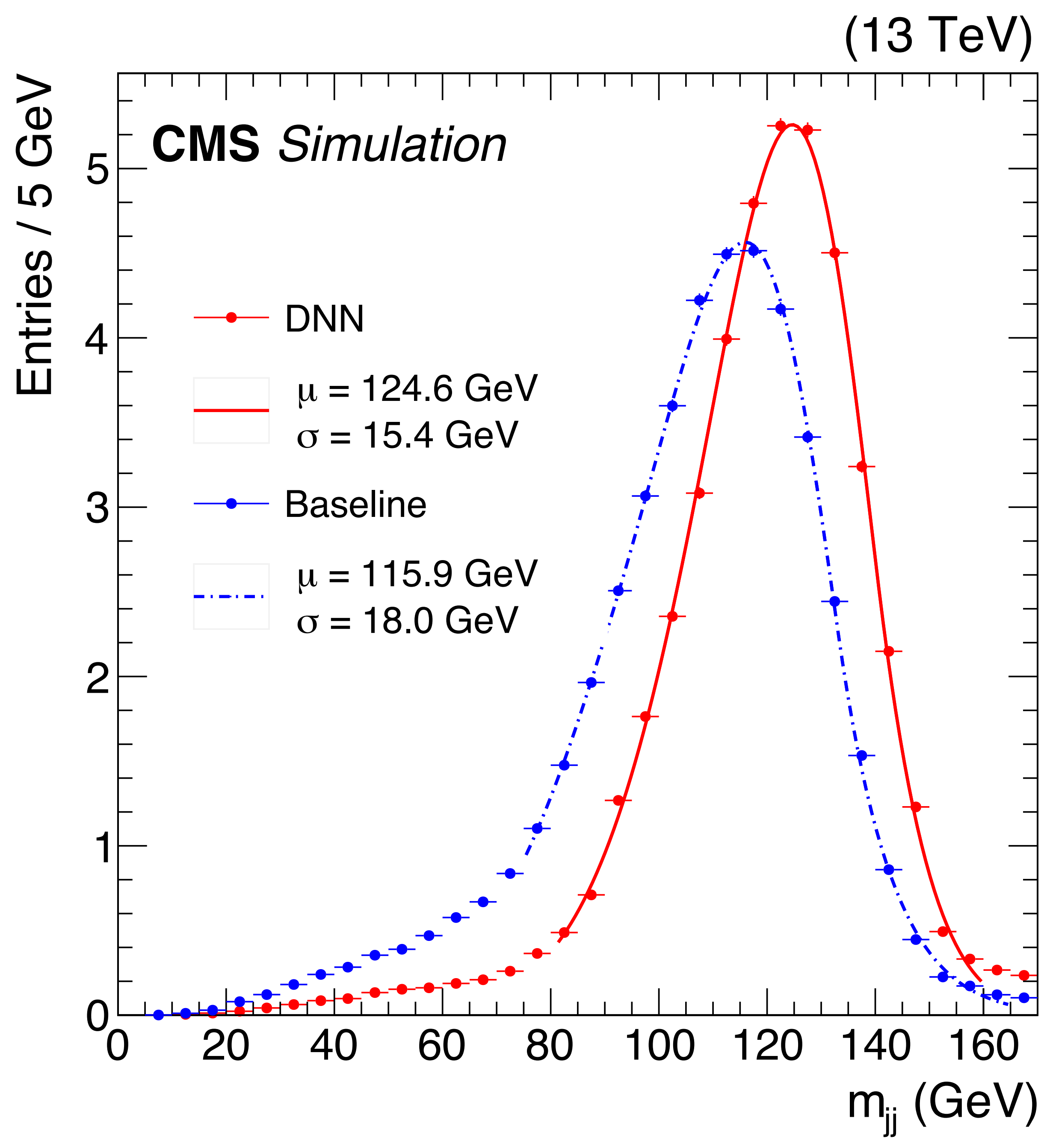


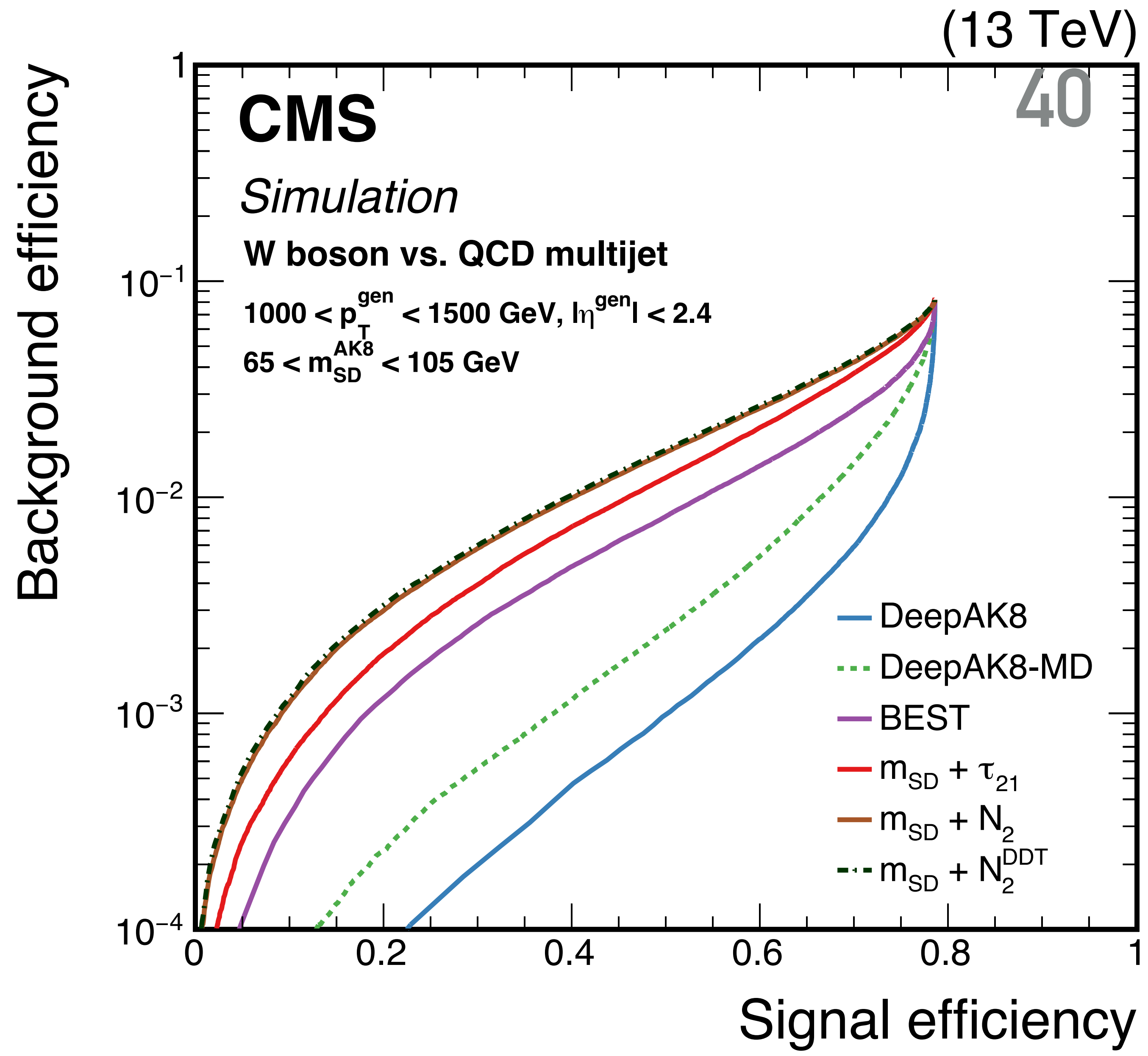
$$y = \sigma(W_2 \sigma(W_1 x))$$

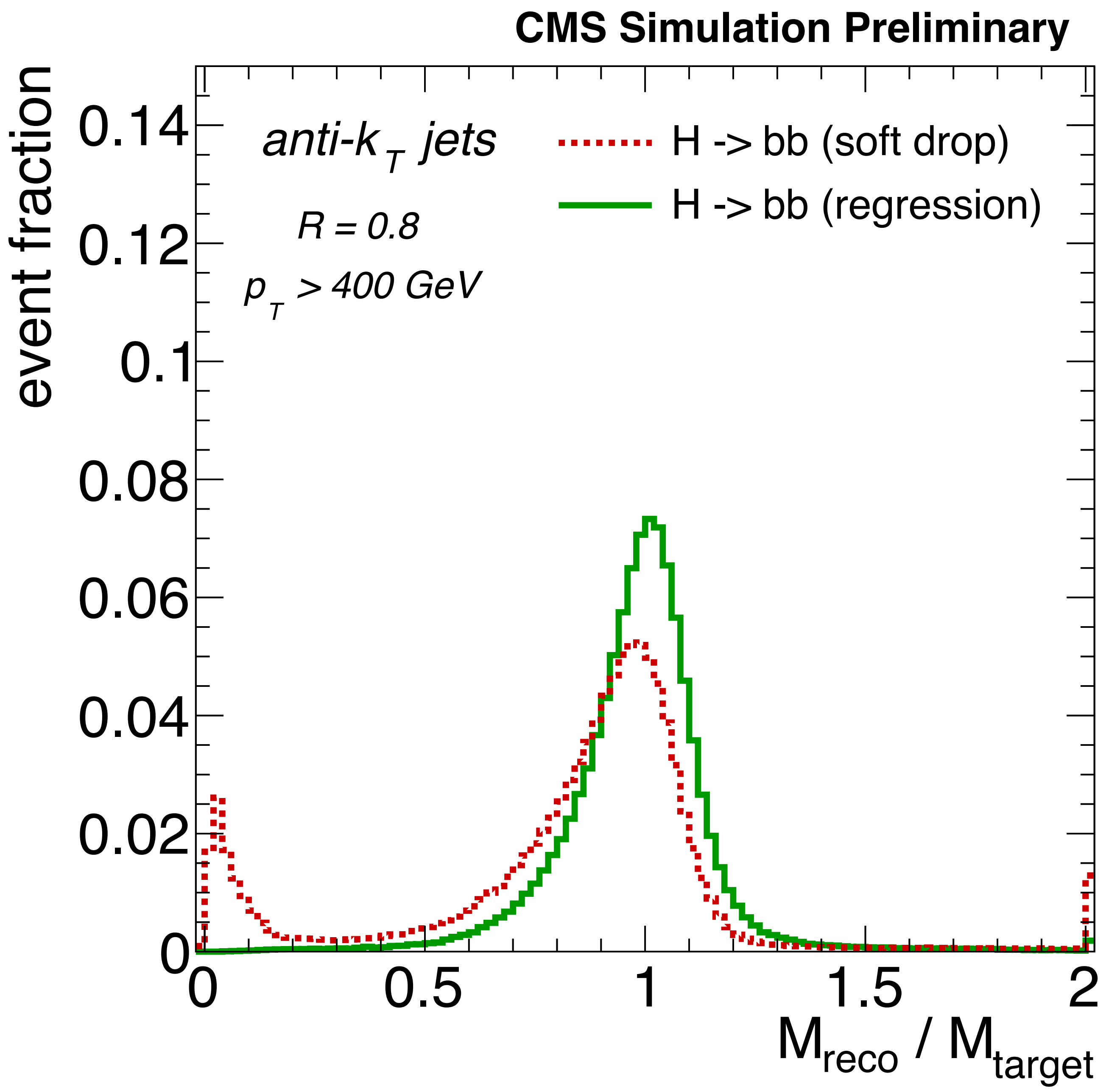
colah.github.io/posts/2014-03-NN-Manifolds-Topology

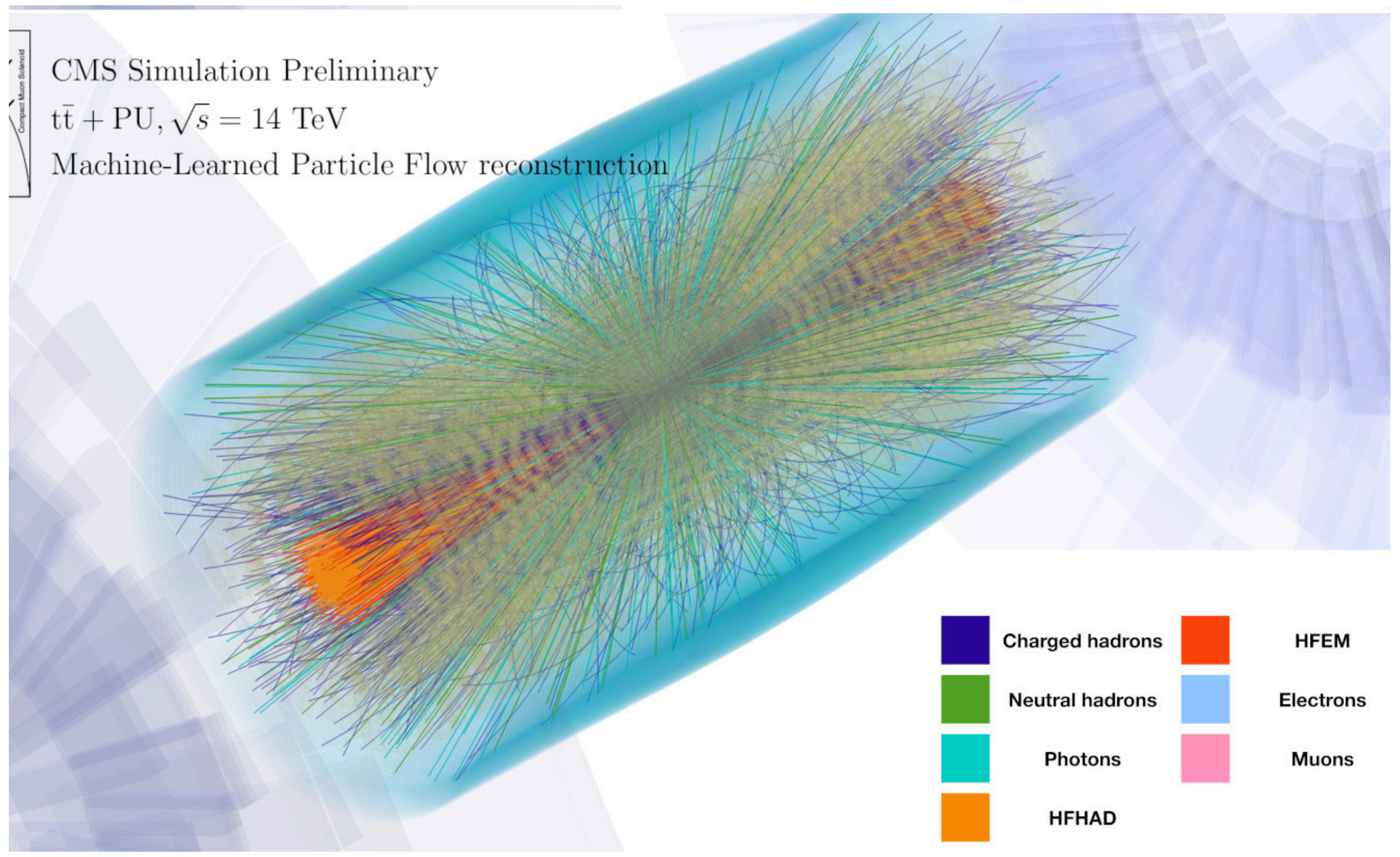
NNS IN THE WILD

- B-jet energy regression [arXiv:1912.06046]
- Jet classification [arXiv:2004.08262]
- Jet mass regression [https://cds.cern.ch/record/2777006]
- Tracking
- Clustering
- Particle-flow reconstruction [arXiv:2203.00330]
- Anomaly detection
- Fast simulation
- Trigger applications
- Background modeling



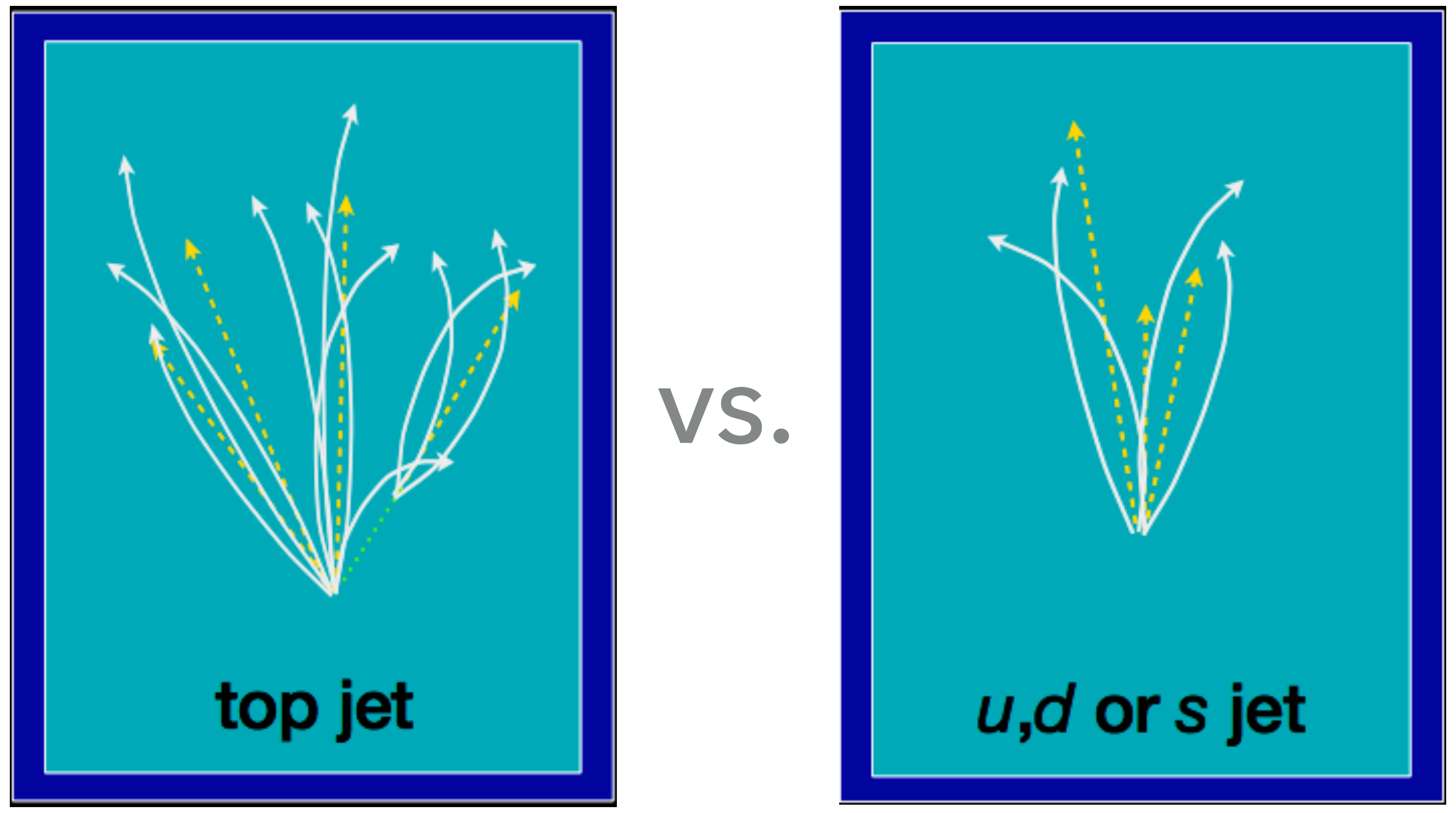


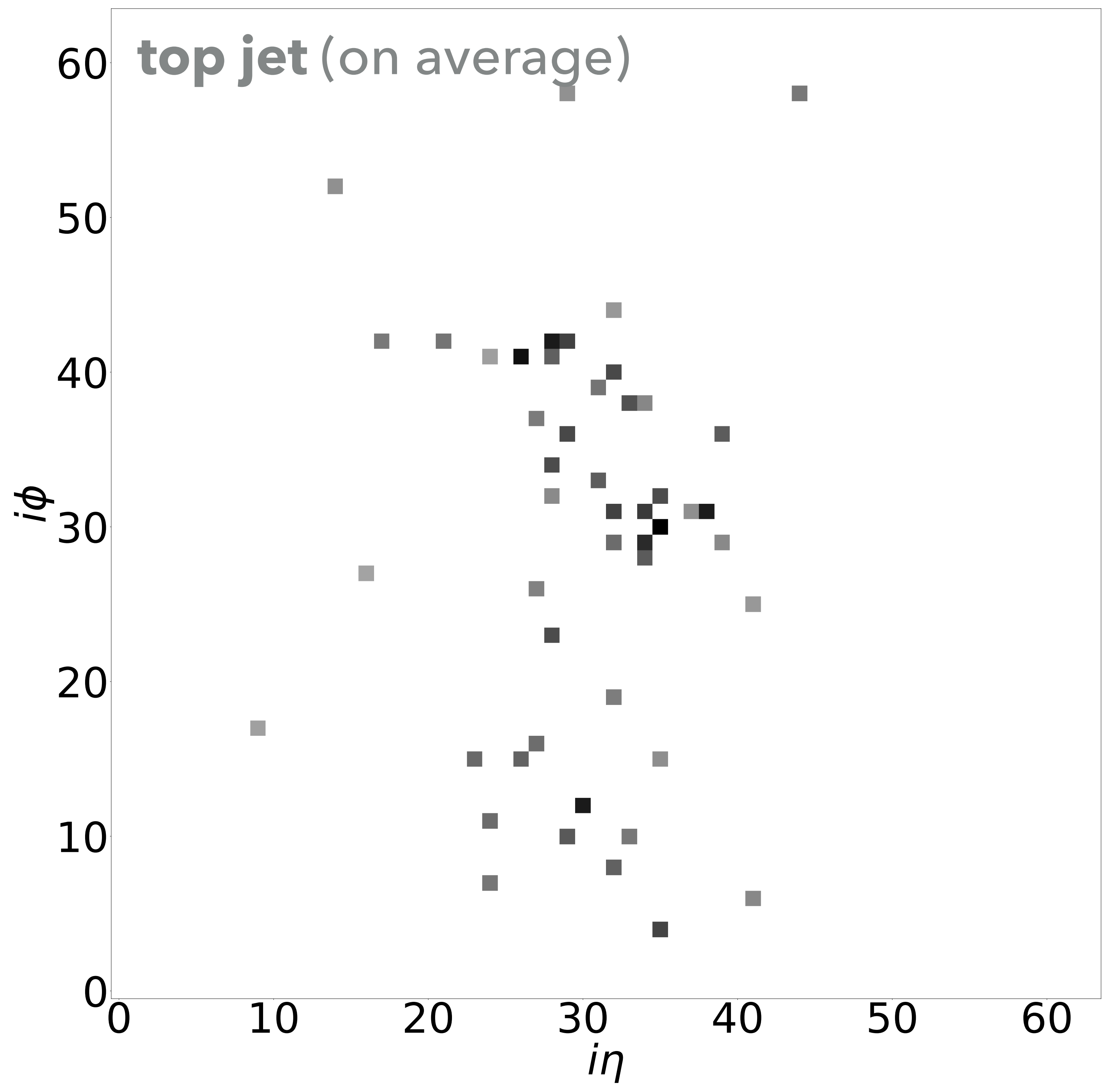


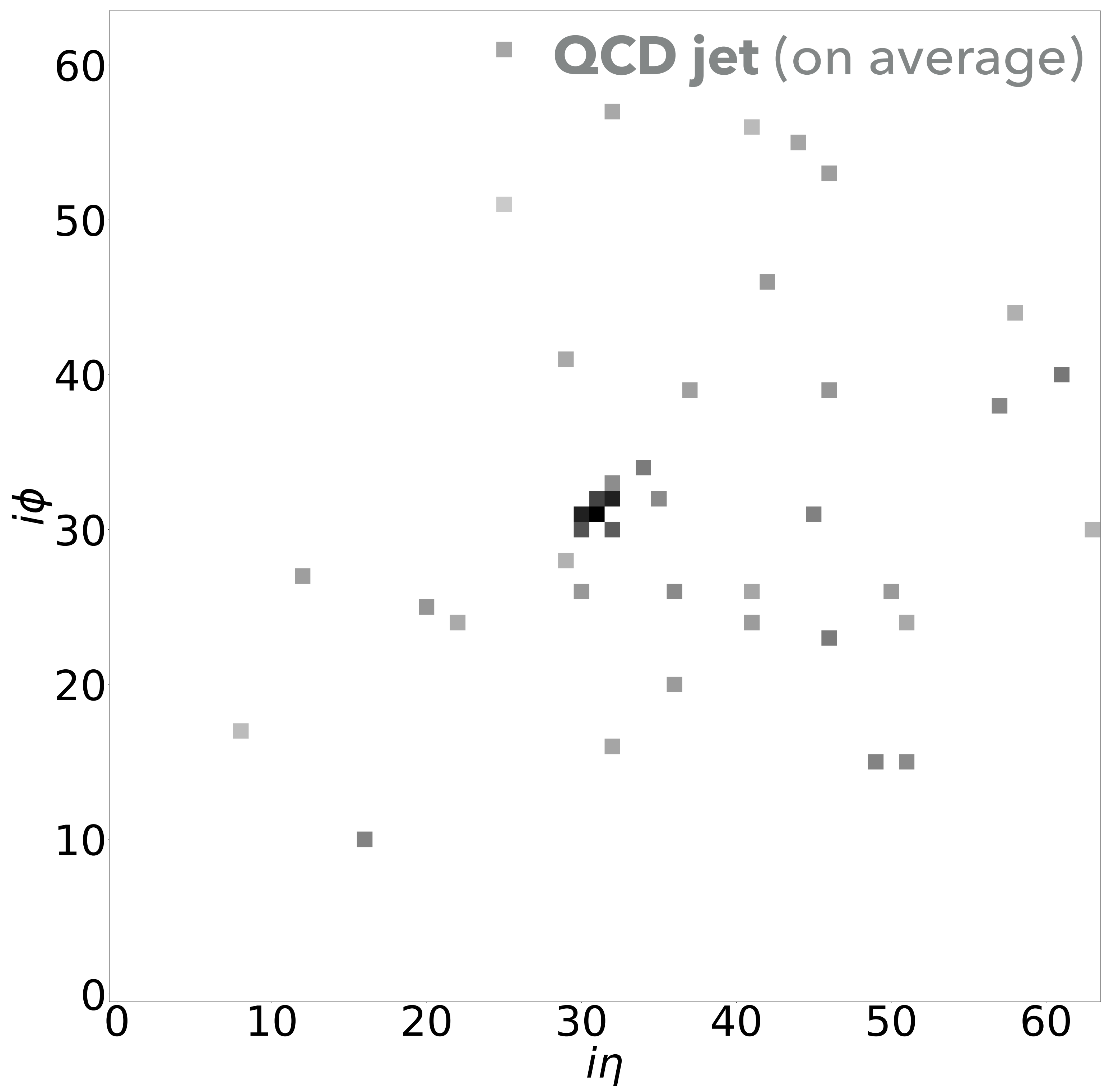

CMS Simulation Preliminary
$t\bar{t}$ + PU, $\sqrt{s}$ = 14 TeV
Machine-Learned Particle Flow reconstruction
Charged hadrons
Neutral hadrons
Photons
HFHAD
HFEM
Electrons
Muons

JET IMAGES

- Jet images = pixelated versions of calorimeter hits in 2D (η, ϕ)
- Much lower level



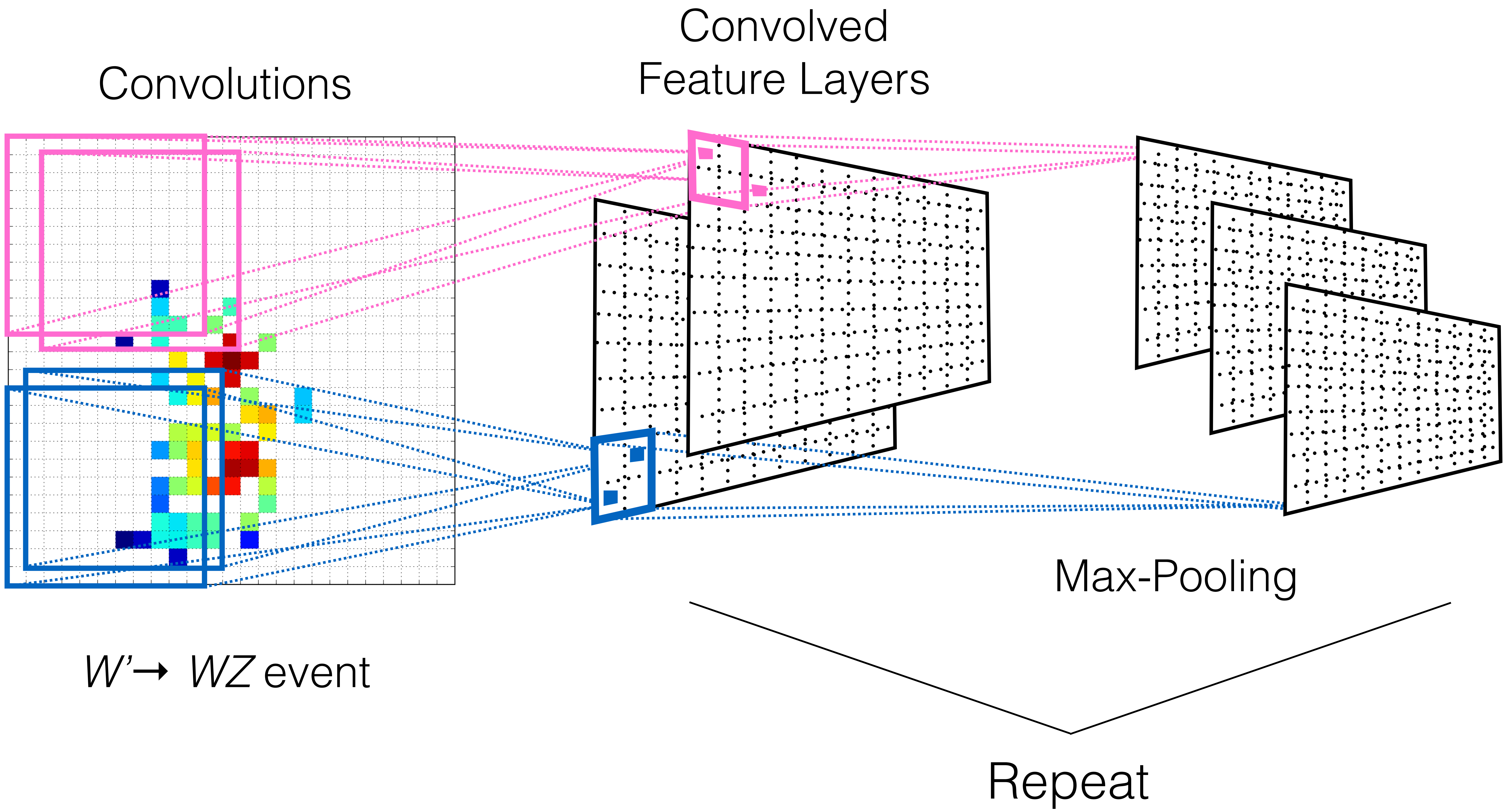

top jet (on average)


QCD jet (on average)

CONVOLUTIONAL NEURAL NETWORKS

arXiv:1511.05190

- Natural to apply 2D CNN

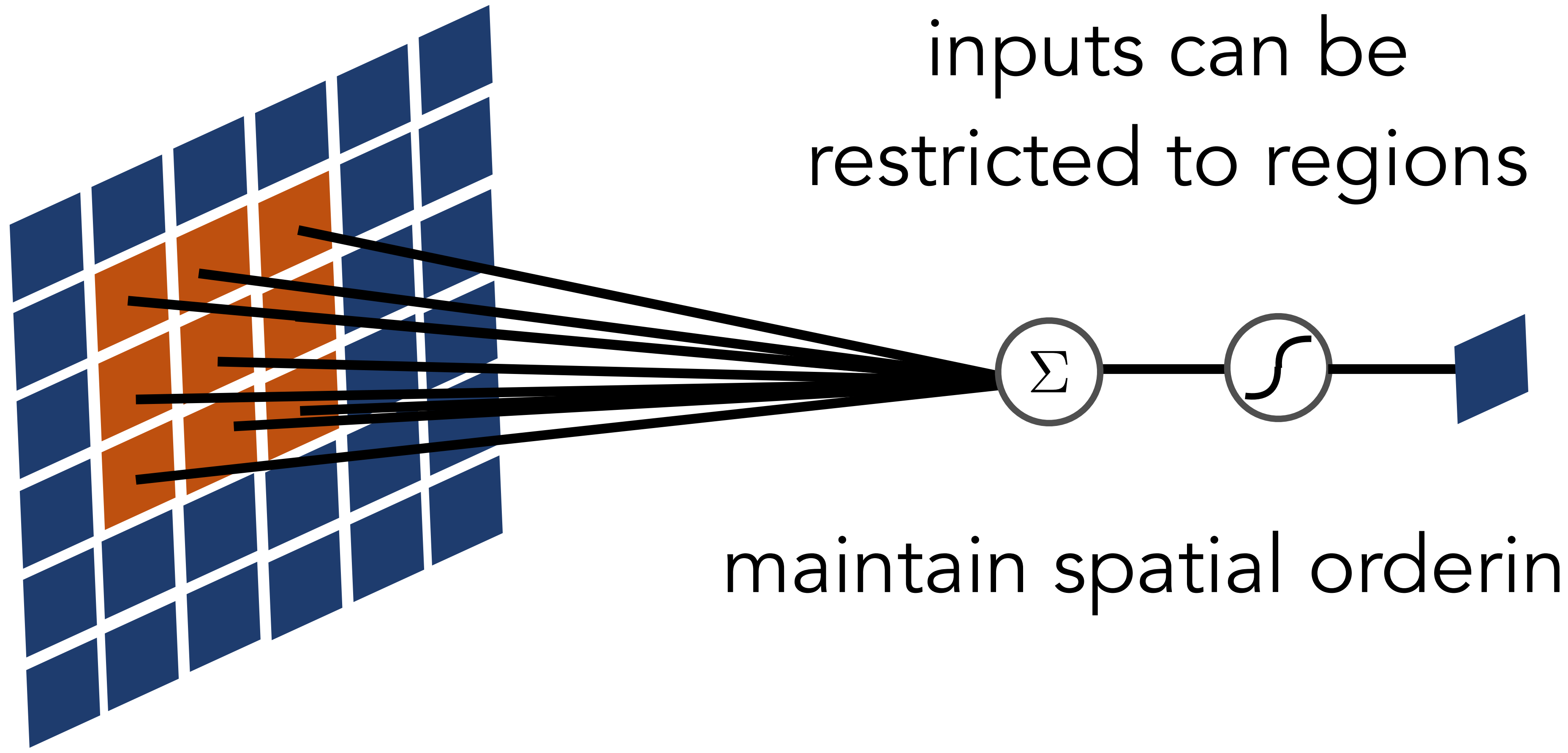


LOCALITY AND TRANSLATION INVARIANCE

Locality and **translation invariance** as *inductive biases*

locality

nearby areas tend to contain stronger patterns

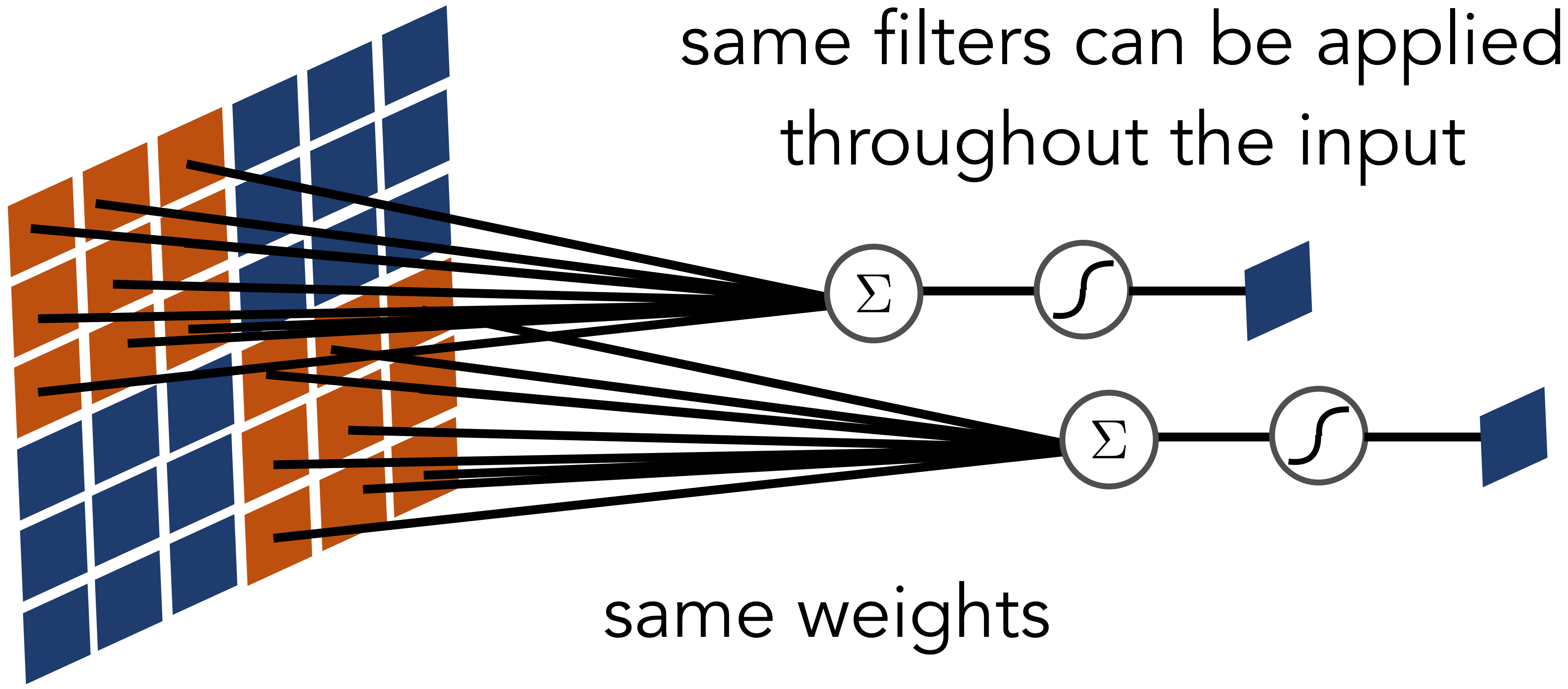


inputs can be restricted to regions

maintain spatial ordering

translation invariance

relative positions are relevant



same filters can be applied throughout the input

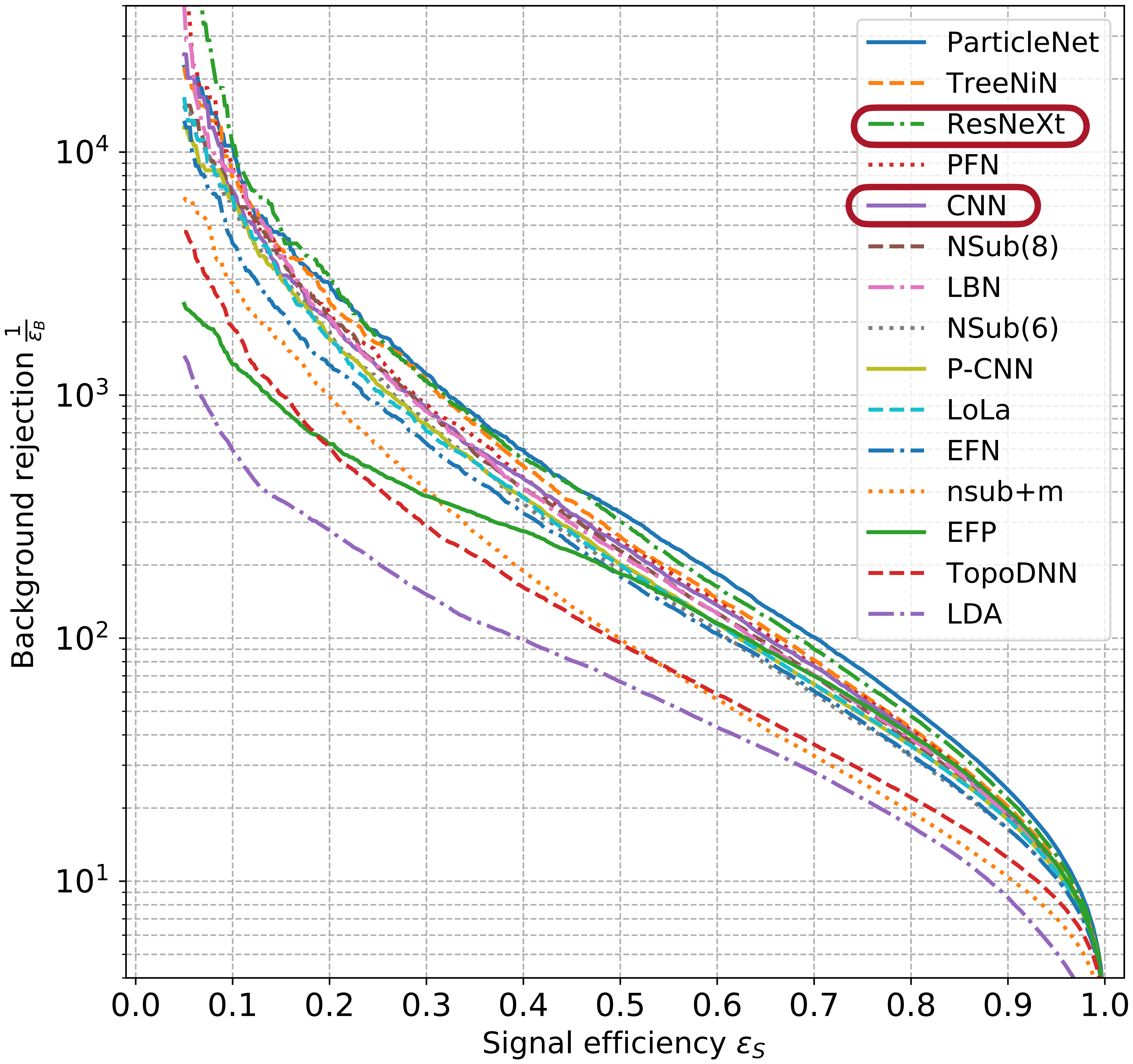
same weights

CNN PERFORMANCE

arXiv:1902.09914

- CNNs among the best performing algorithms

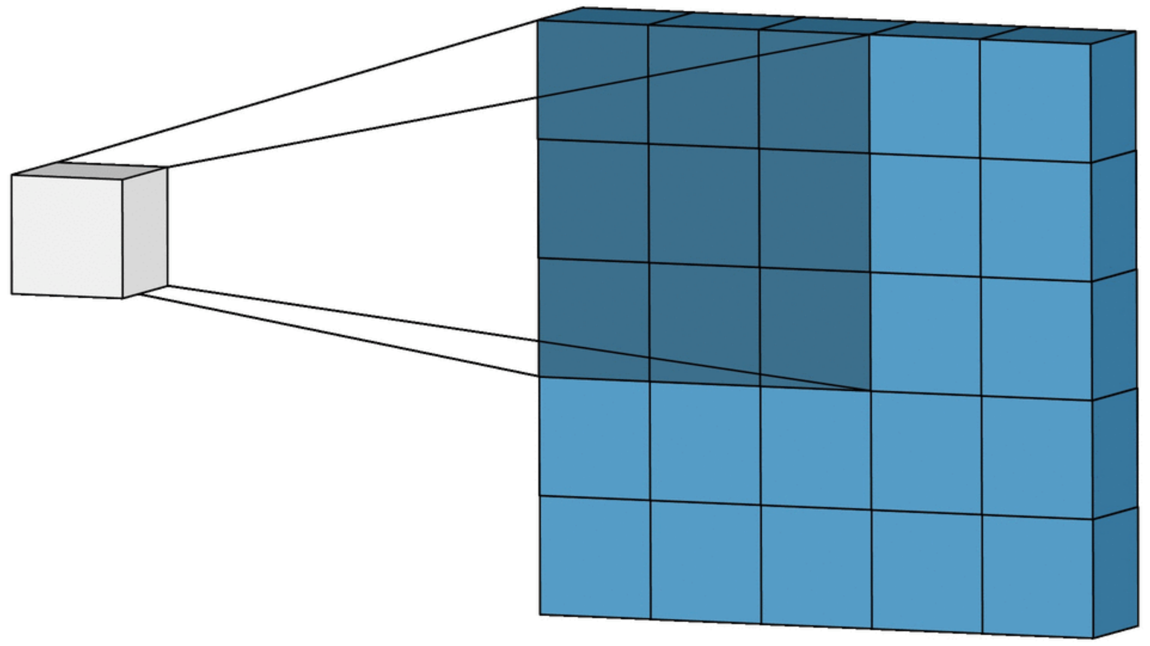
	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$)			#Param
			single	mean	median	
CNN [16]	0.981	0.930	914±14	995±15	975±18	610k
ResNeXt [31]	0.984	0.936	1122±47	1270±28	1286±31	1.46M
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8	59k
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13	57k
Multi-body N-subjettiness 8 [24]	0.981	0.929	867±15	918±20	926±18	58k
TreeNiN [43]	0.982	0.933	1025±11	1202±23	1188±24	34k
P-CNN	0.980	0.930	732±24	845±13	834±14	348k
ParticleNet [47]	0.985	0.938	1298±46	1412±45	1393±41	498k
LBN [19]	0.981	0.931	836±17	859±67	966±20	705k
LoLa [22]	0.980	0.929	722±17	768±11	765±11	127k
LDA [54]	0.955	0.892	151±0.4	151.5±0.5	151.7±0.4	184k
Energy Flow Polynomials [21]	0.980	0.932	384			1k
Energy Flow Network [23]	0.979	0.927	633±31	729±13	726±11	82k
Particle Flow Network [23]	0.982	0.932	891±18	1063±21	1052±29	82k
GoaT	0.985	0.939	1368±140		1549±208	35k



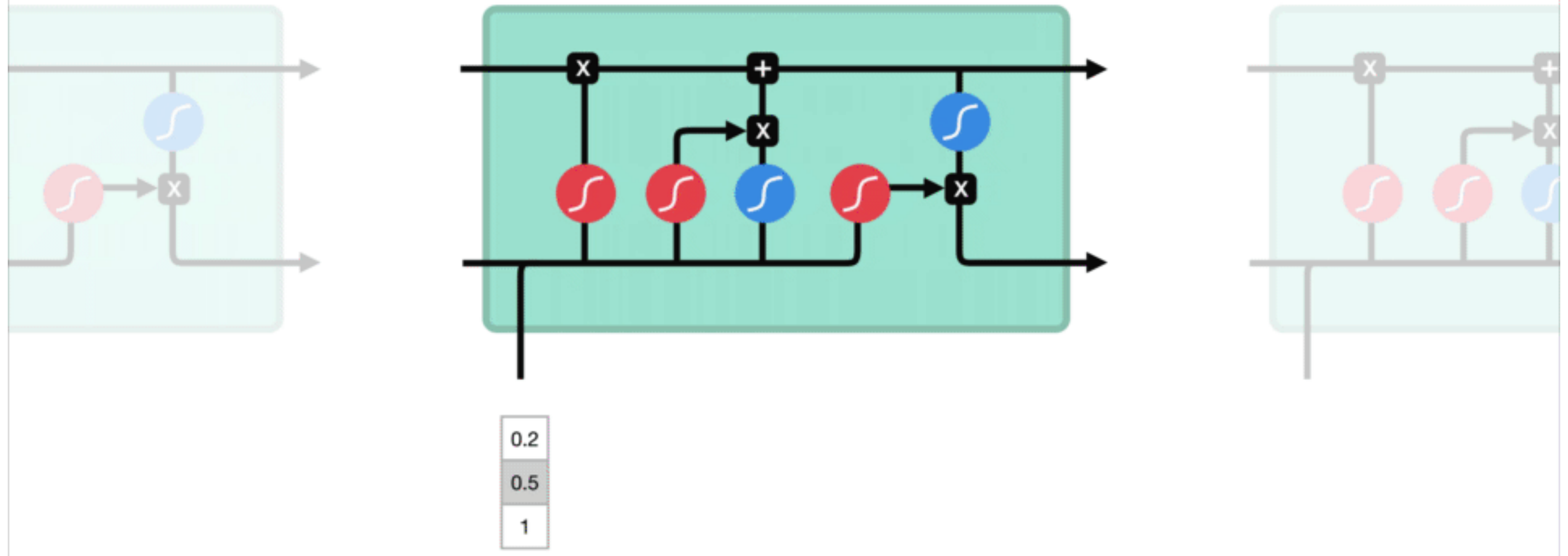
INNOVATING WITH NEW REPRESENTATIONS

arXiv:2007.13681
arXiv:2012.01249

- In deep learning, tailoring algorithms to the structure (and symmetries) of the data has led to groundbreaking performance
  - CNNs for images

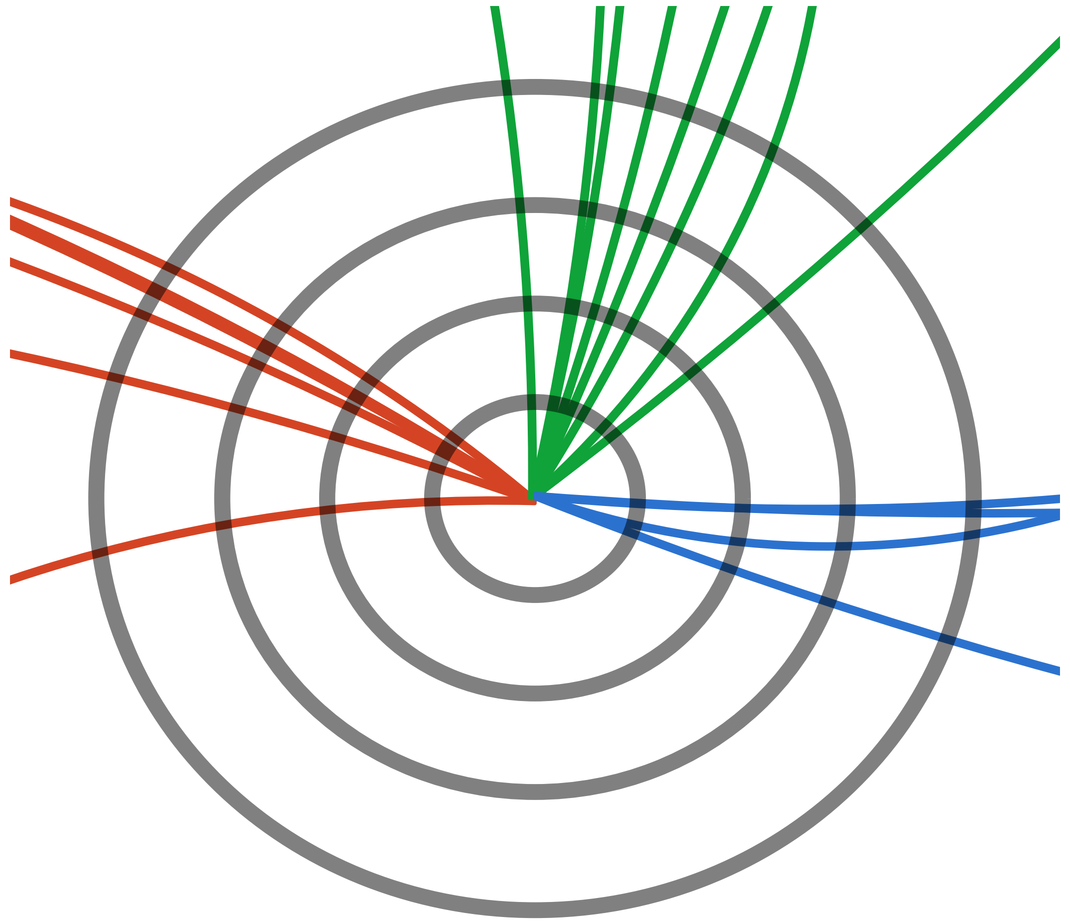


  - RNNs for language processing



- What about high energy physics data like jets?

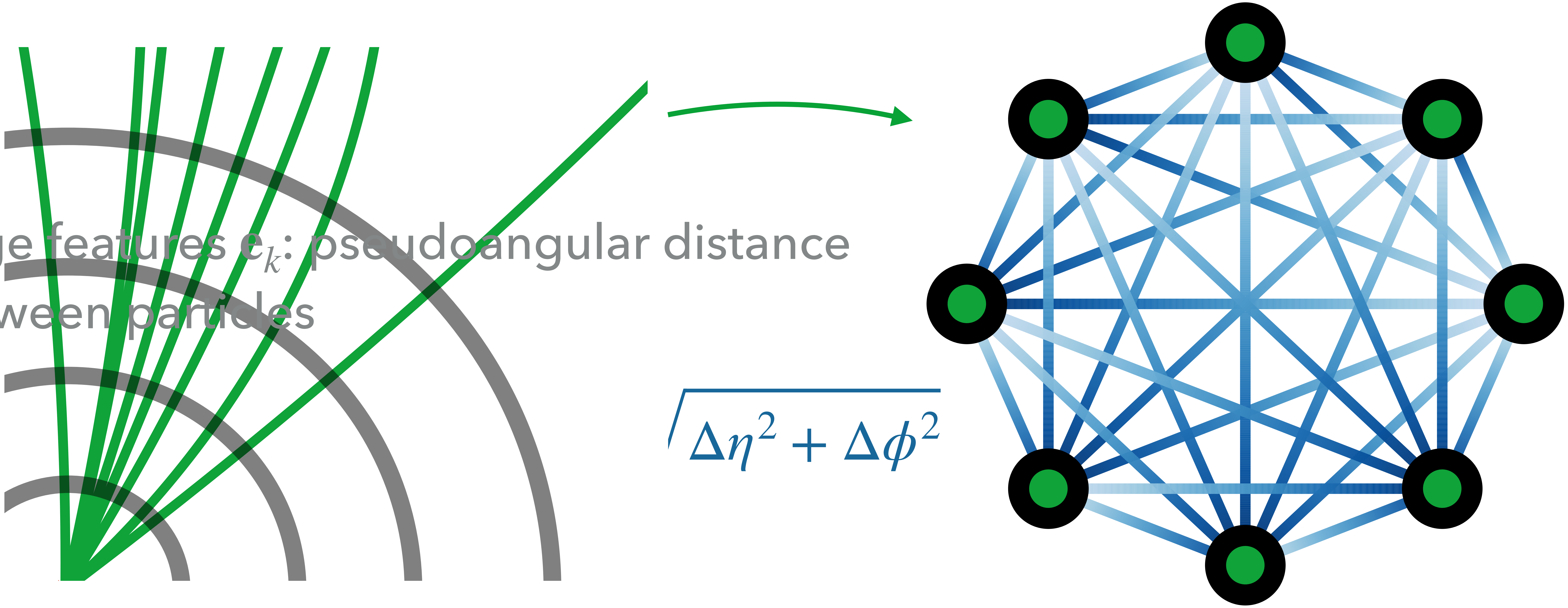
  - Distributed unevenly in space
  - Sparse
  - Variable size
  - No defined order
  - Interconnections
    → Graphs



NODE, EDGE, GRAPH FEATURES IN HEP (E.G. JET)

- Node features $\mathbf{v}_i$: particle 4-momentum

$$p = [E, p_x, p_y, p_z] \equiv [p_\mathrm{T}, \eta, \phi, m]$$



- Edge features $\mathbf{e}_k$: pseudoangular distance between particles

$$\sqrt{\Delta\eta^2 + \Delta\phi^2}$$

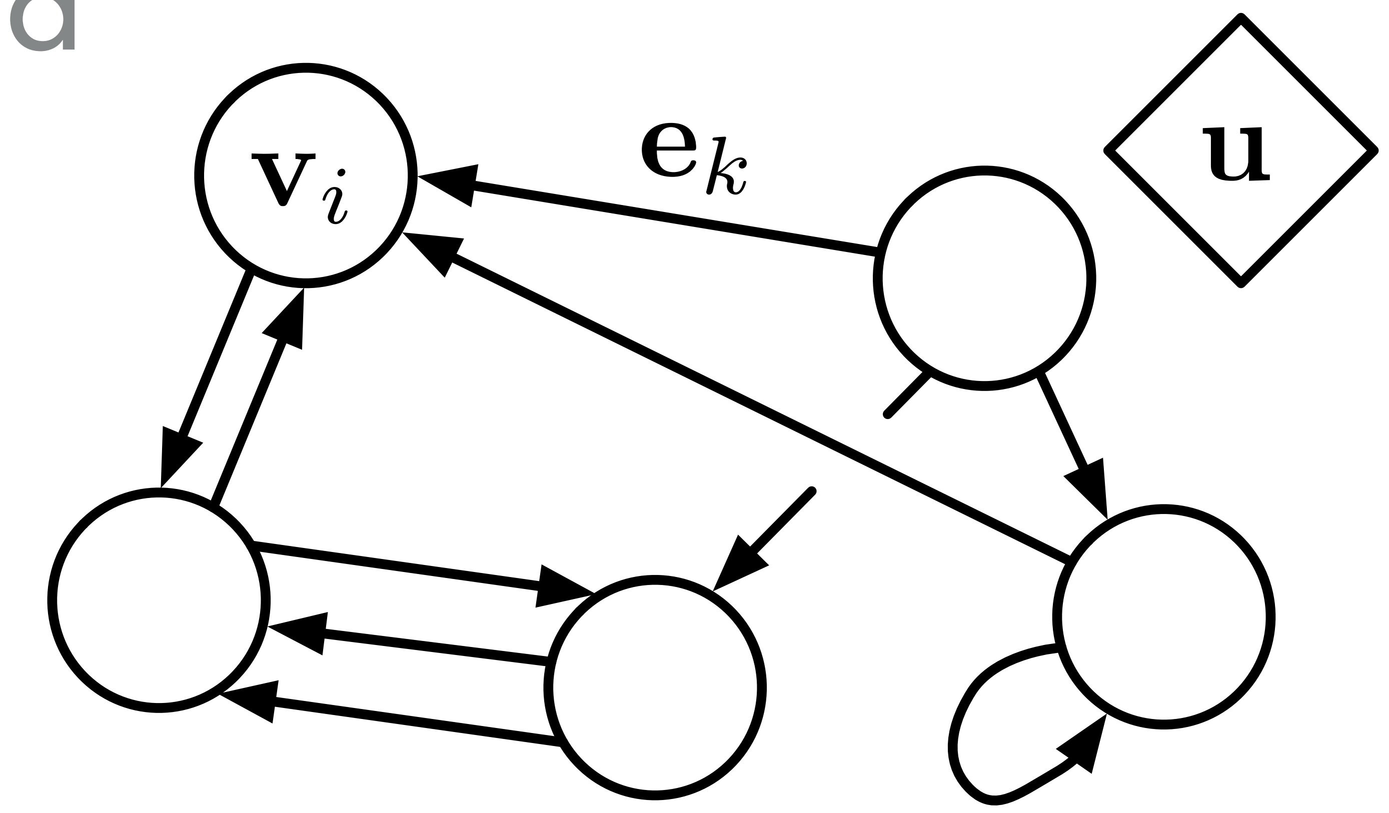
- Graph (global) features $\mathbf{u}$: jet mass

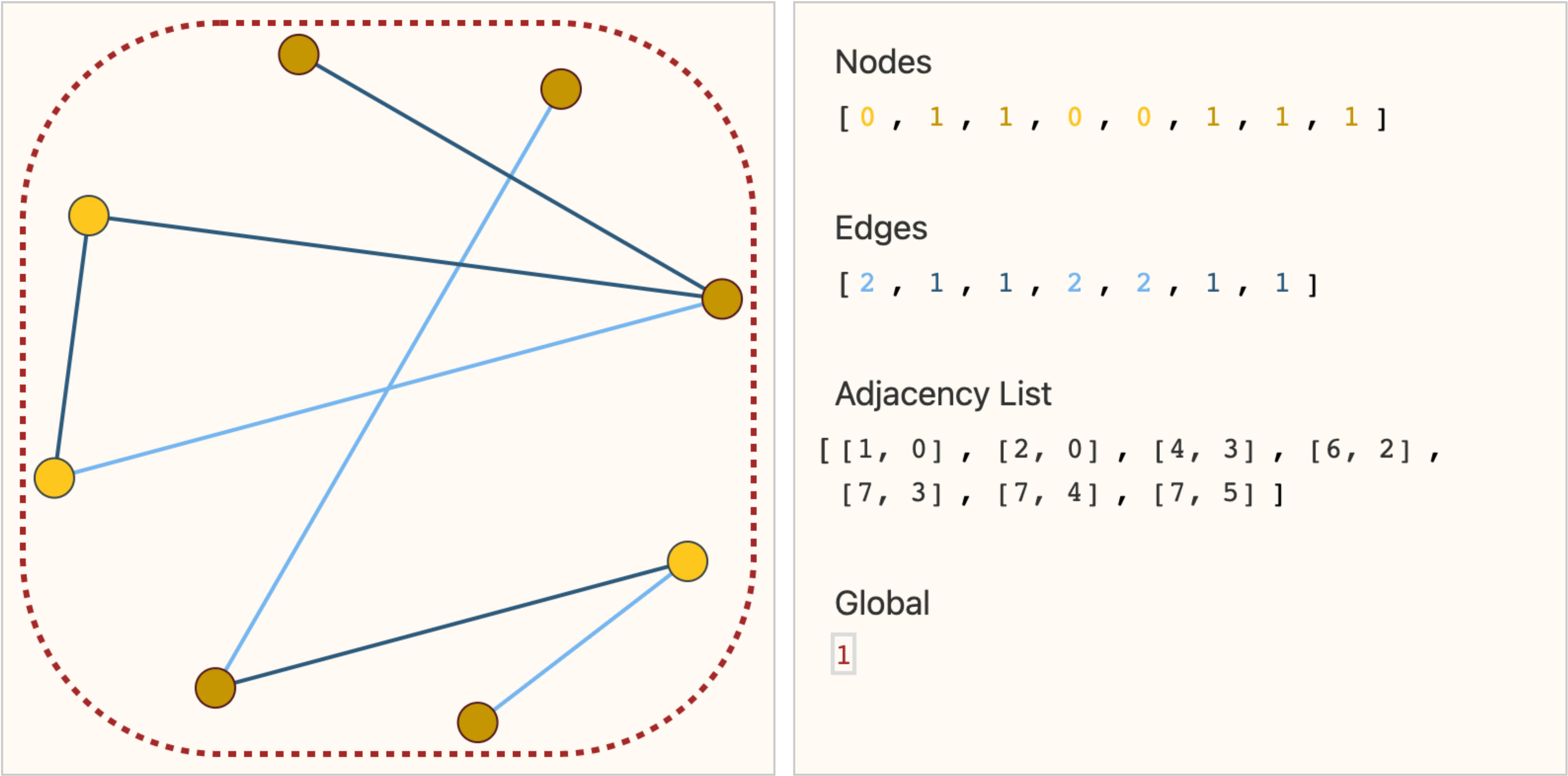
$$m = \sqrt{\sum_{i \in \mathrm{jet}} E_i^2 - p_{x,i}^2 - p_{y,i}^2 - p_{z,i}^2}$$

FORMALIZING A GRAPH

https://distill.pub/2021/gnn-intro/

- Features: triplet of global features, node features, and edge features: $(\mathbf{u}, V, E)$
- Graph connectivity: adjacency matrix $A = \{a_{ij} = 1 \text{ if } i \text{ is connected to } j\}$
  - Sparse representation:
    "receiver" indices $\boldsymbol{r}$ and "sender" indices $\boldsymbol{s}$
    e.g. kth edge connects node s$_k$ to node r$_k$



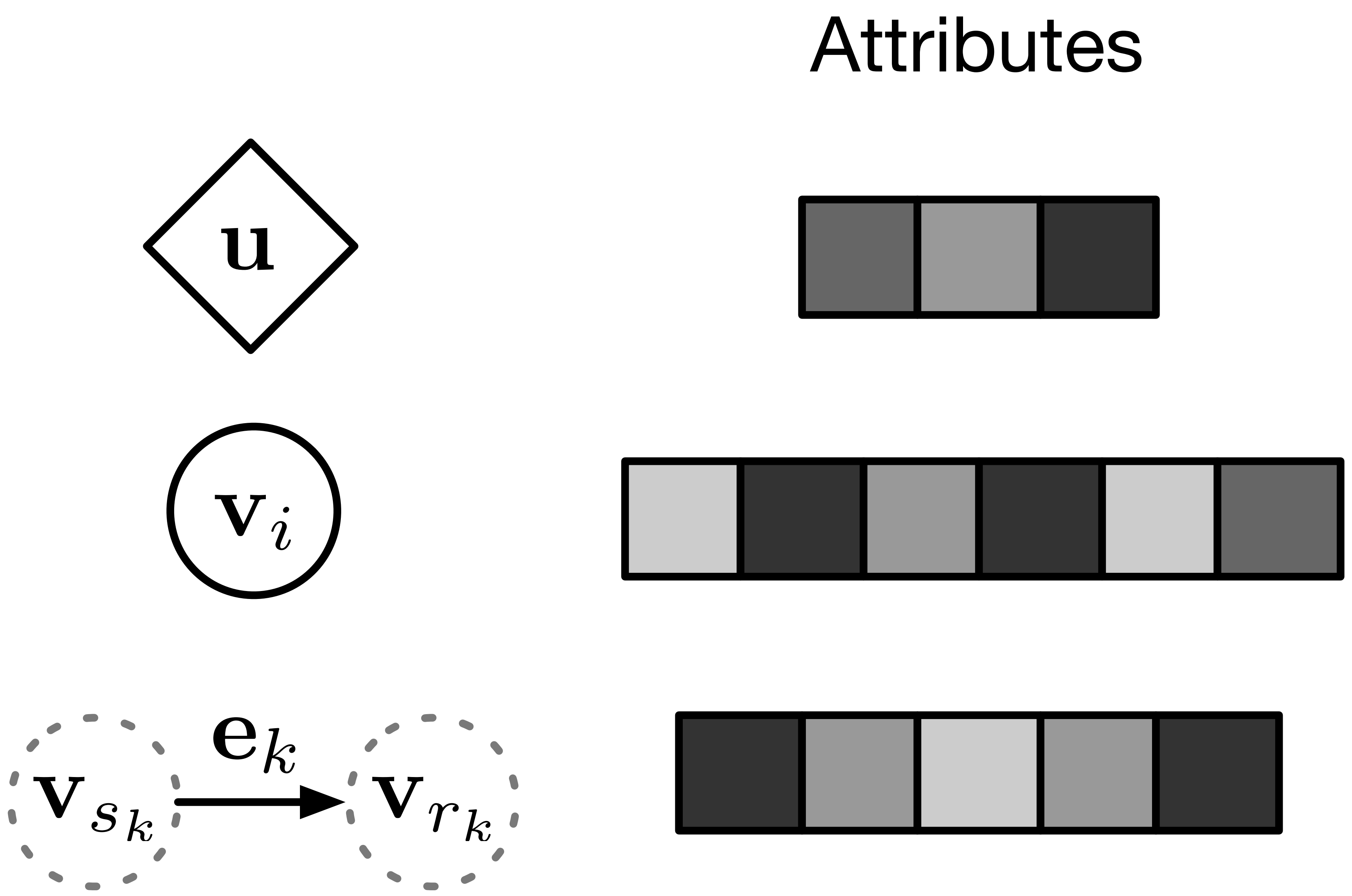


Nodes
```
[ 0 , 1 , 1 , 0 , 0 , 1 , 1 , 1 ]
```

Edges
```
[ 2 , 1 , 1 , 2 , 2 , 1 , 1 ]
```

Adjacency List
```
[ [1, 0] , [2, 0] , [4, 3] , [6, 2] ,
  [7, 3] , [7, 4] , [7, 5] ]
```

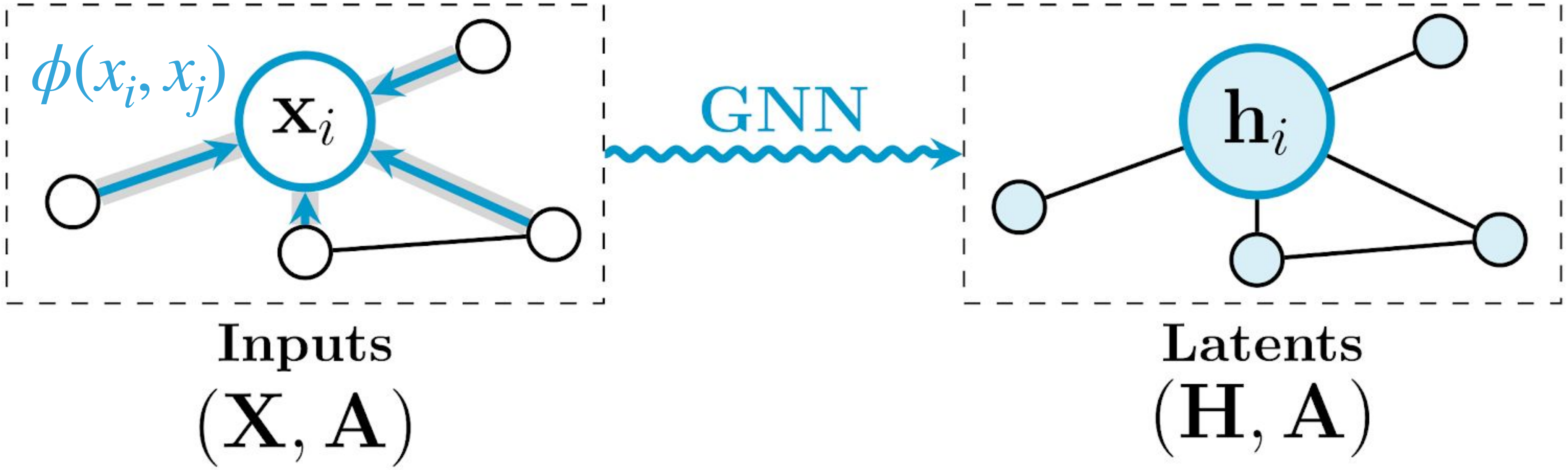
Global
```
1
```



GNN'S MAIN INGREDIENT: MESSAGE PASSING

- For all neighbors j of node i compute a "message" via a NN: $\phi(x_i, x_j)$

- Update the node features by summing all messages:
  $$h_i = \sum_j \phi(x_i, x_j)$$

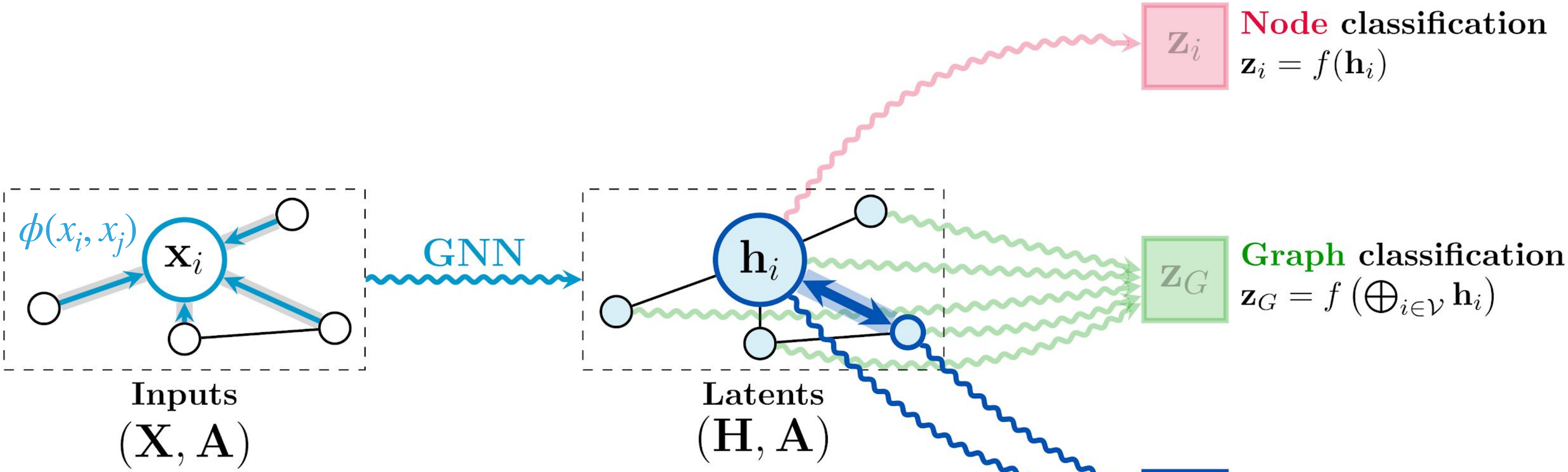


"message passing"

HOW TO USE GNNS IN HEP

Source: https://youtu.be/uF53xsT7mjc

- Node-level tasks
  - Identify "pileup" particles
- Graph-level tasks
  - **Jet tagging**

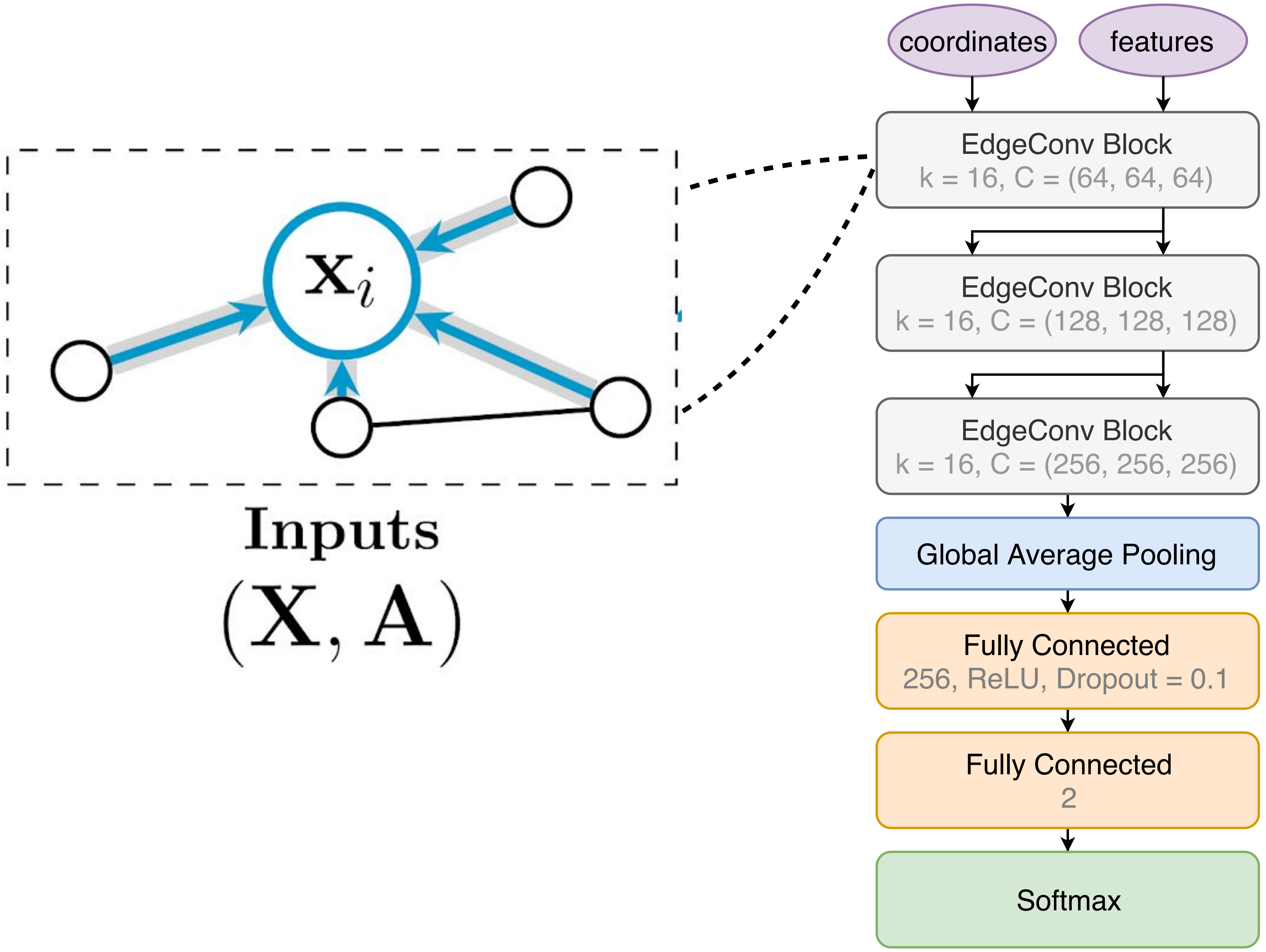

$\phi(x_i, x_j)$
$\mathbf{x}_i$
GNN
$\mathbf{h}_i$
Inputs
$(\mathbf{X}, \mathbf{A})$
Latents
$(\mathbf{H}, \mathbf{A})$
$\mathbf{z}_i$
Node classification
$\mathbf{z}_i = f(\mathbf{h}_i)$
$\mathbf{z}_G$
Graph classification
$\mathbf{z}_G = f\left(\bigoplus_{i \in \mathcal{V}} \mathbf{h}_i\right)$
$\mathbf{z}_{ij}$
Link prediction
$\mathbf{z}_{ij} = f(\mathbf{h}_i, \mathbf{h}_j, \mathbf{e}_{ij})$

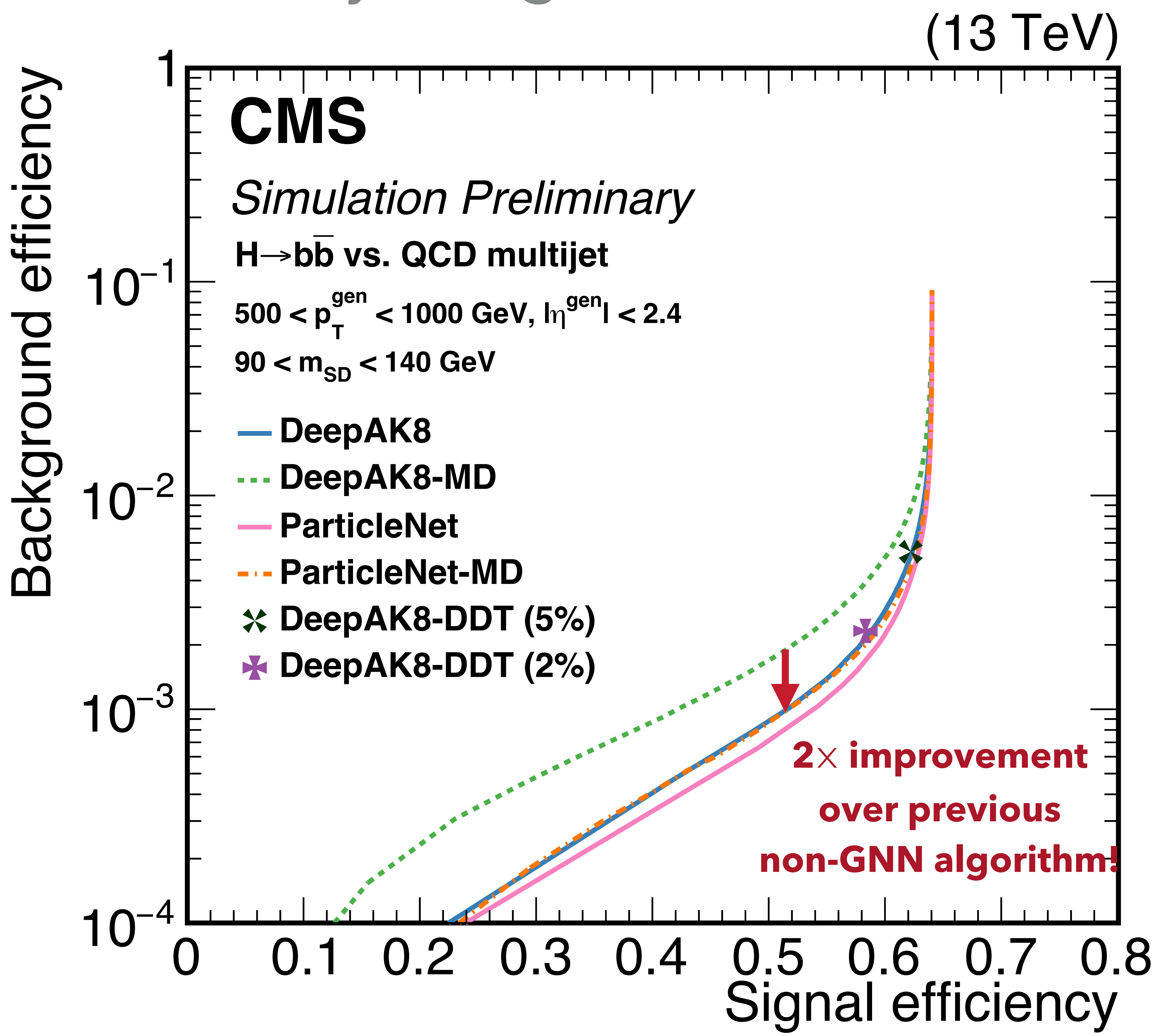
- Edge-level tasks
  - Identify good track doublets

PARTICLENET: GNN FOR TAGGING H(BB) IN CMS

arXiv:1902.08570
CMS-DP-2020-002

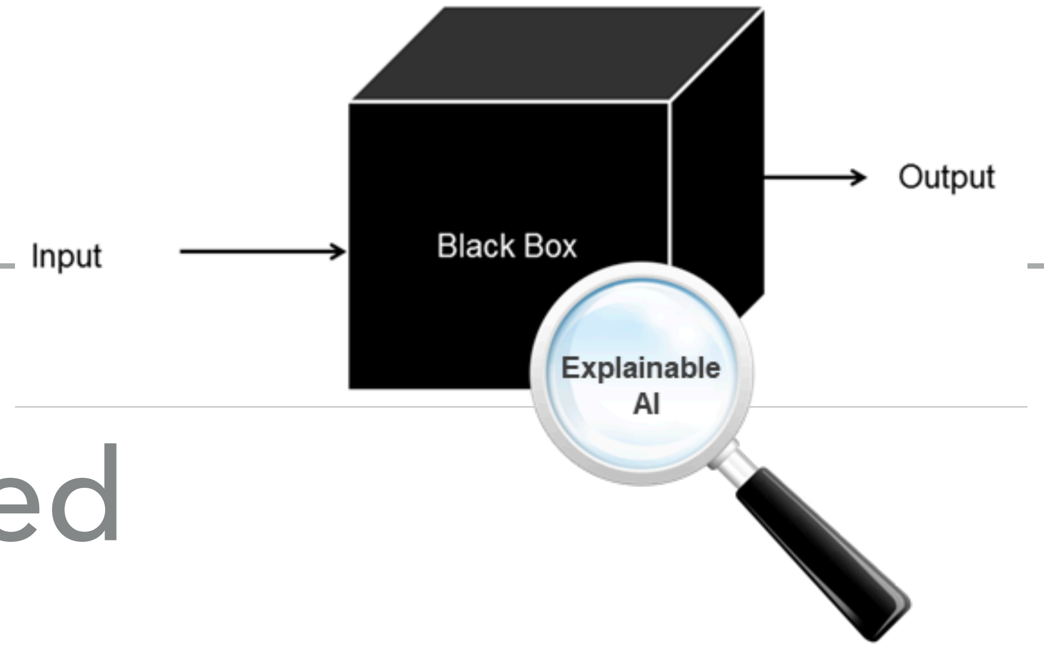
- ParticleNet, using “dynamic edge convolutions:” graph is constructed based on “closeness” in an abstract “latent” space
- Identifies H(bb) with an efficiency of ~50% while rejecting 99.9% of background





2× improvement over previous non-GNN algorithm!

WHAT IS PARTICLENET LEARNING?



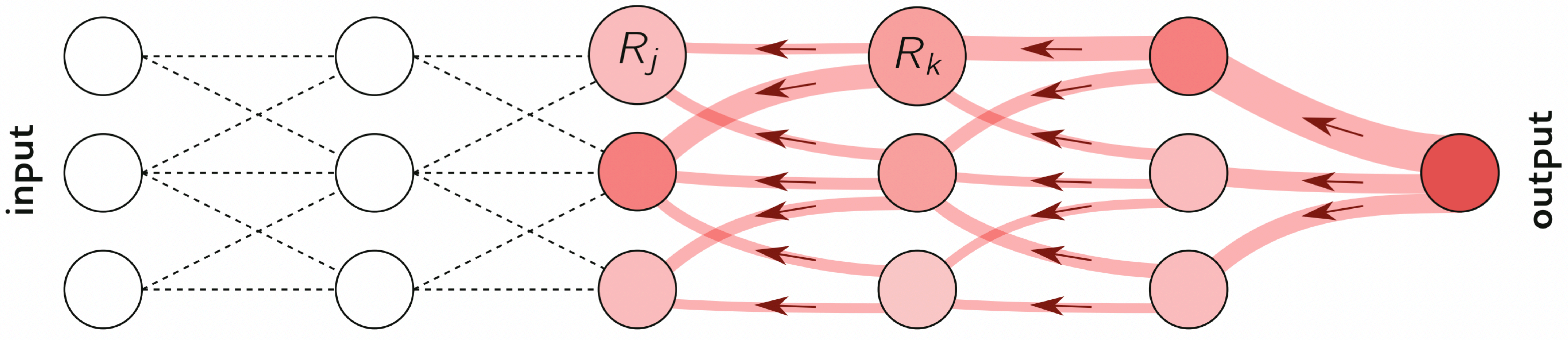
- Explainable AI (XAI) refers to the set of techniques employed to provide explanations for ML model predictions

- Layerwise relevance propagation (LRP) [1] computes relevance (R) scores for each neuron in a ML model

- Neuron's R score is a measure of its contribution to the model's output,

$$R_j^{(l)} = \sum_k \frac{z_{jk}}{\sum_m z_{mk}} R_k^{(l+1)} \text{ with } z_{jk} = x_j^{(l)} w_{jk}^{(l+1)}$$

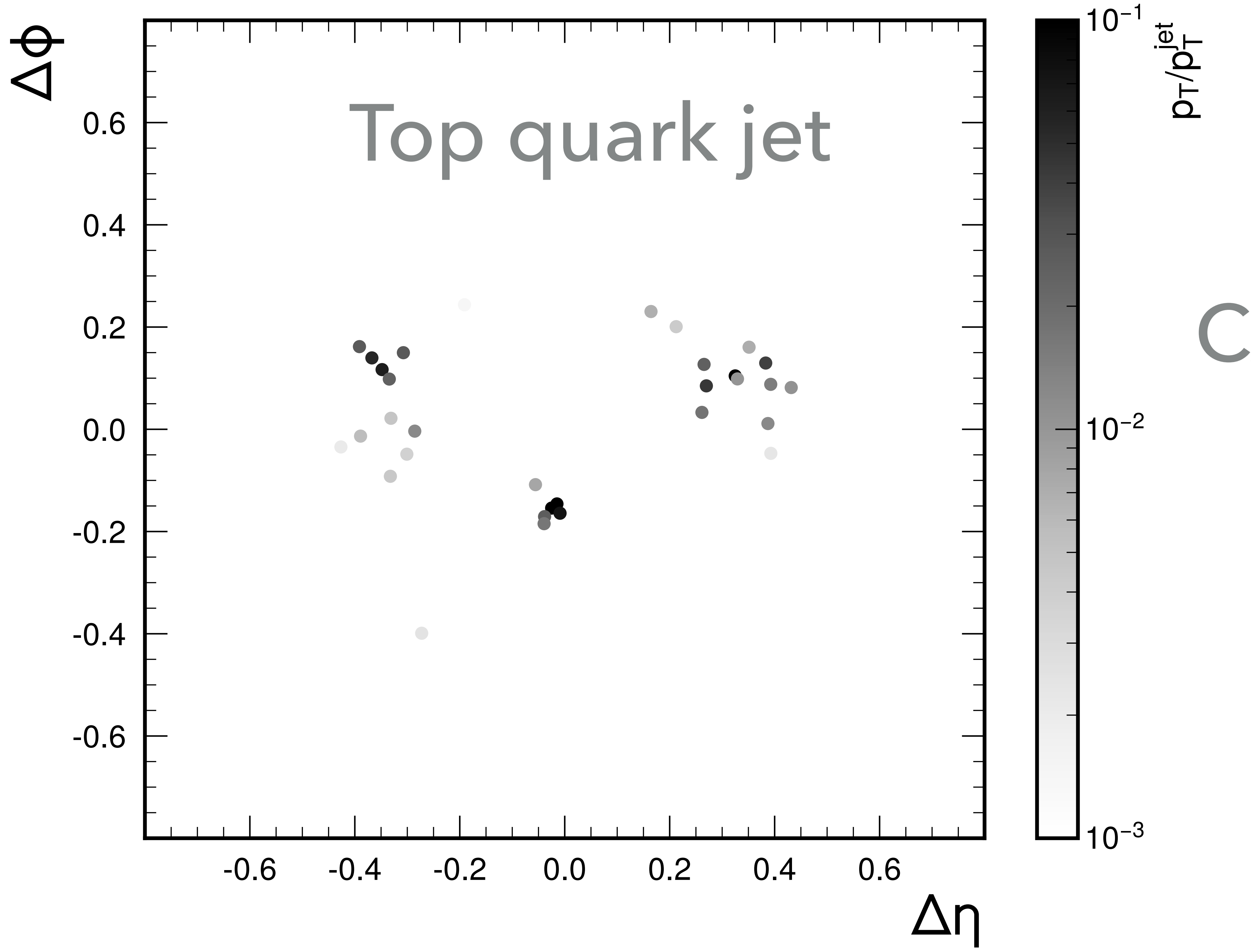
- Flow of R scores for a multilayer perceptron (MLP)



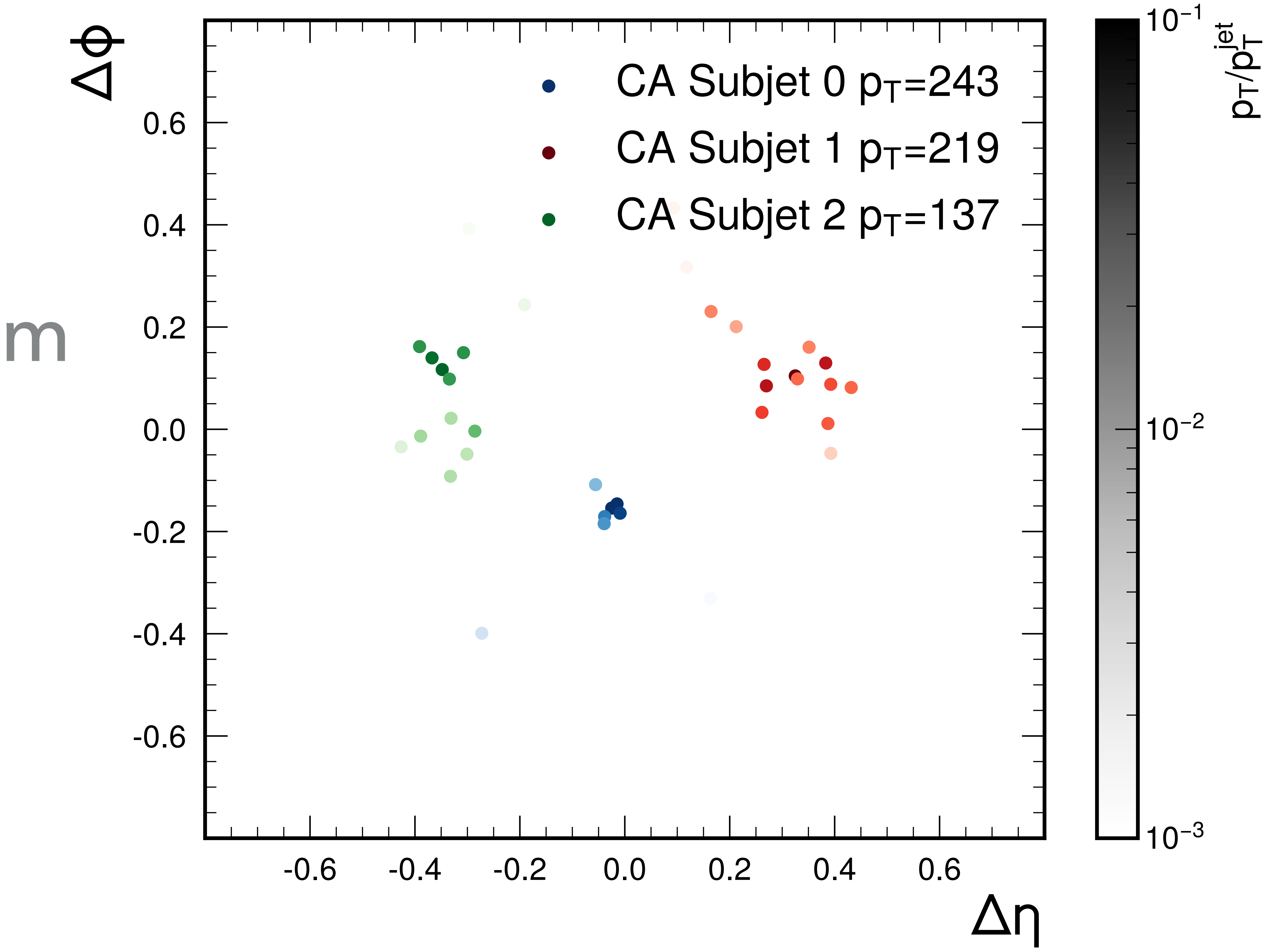
[1] https://doi.org/10.1007/978-3-030-28954-6_10

IS PARTICLENET USING TRADITIONAL JET SUBSTRUCTURE?

- Is the model learning to connect particles from different ***subjets*** more often for top quark jets than for QCD jets?
- Use CA algorithm to decluster each jet into exactly 3 subjets



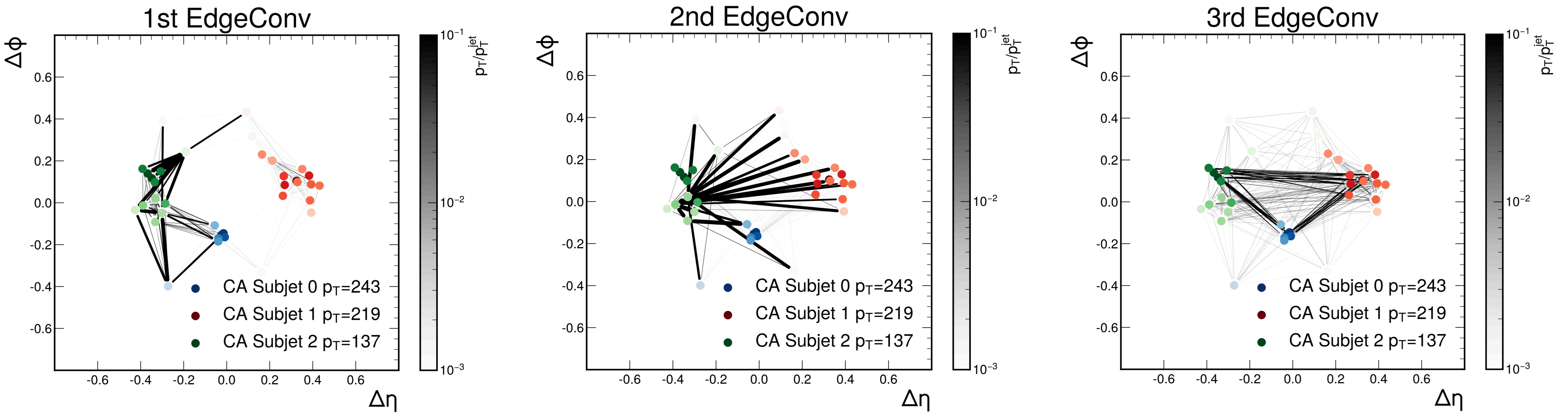
CA algorithm →



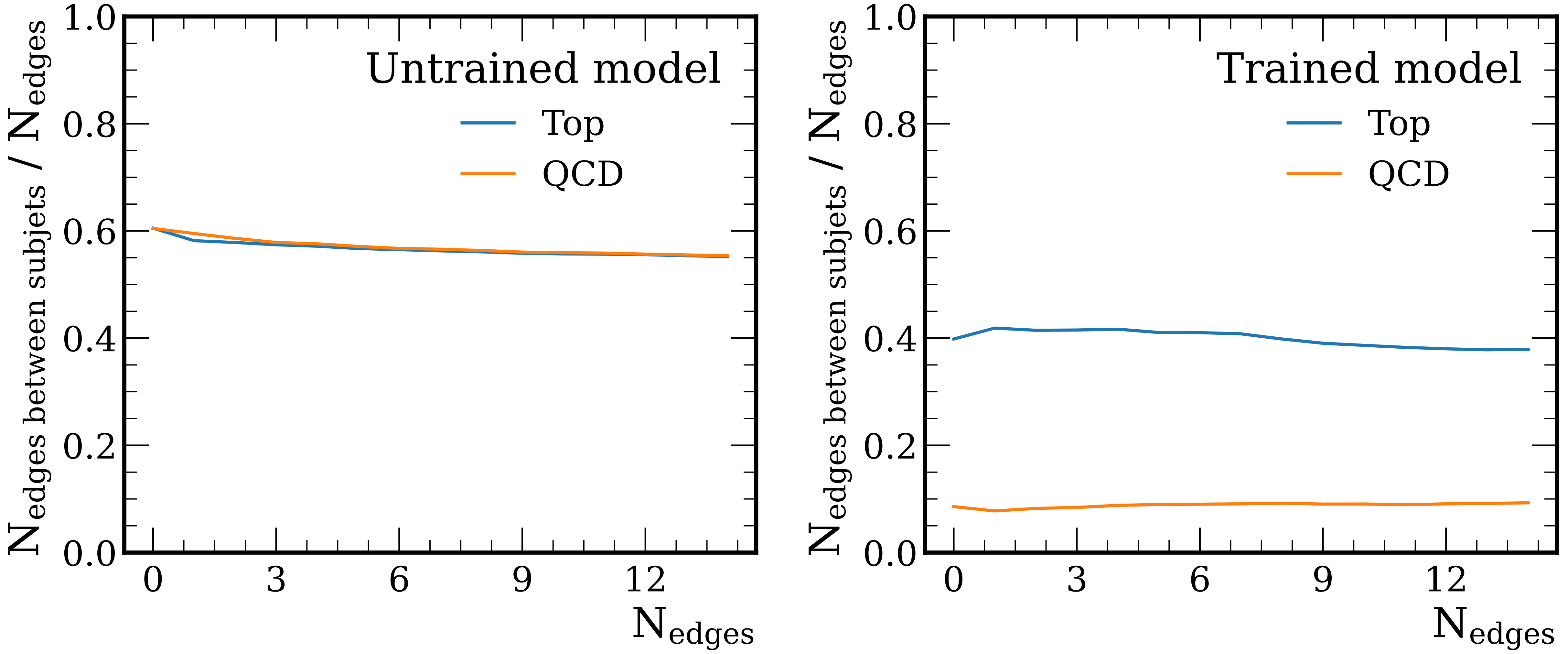
LRP FOR PARTICLENET

arXiv.2211.09912

- For a top quark jet sample, relevant edges connect different subjets

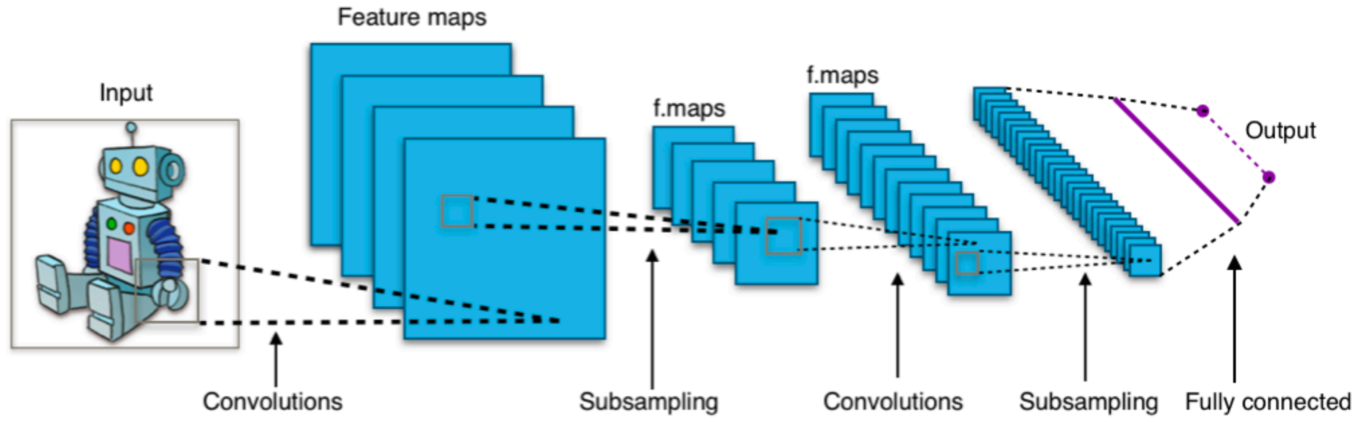


- Trained model connects different subjets more often for top quarks than for QCD



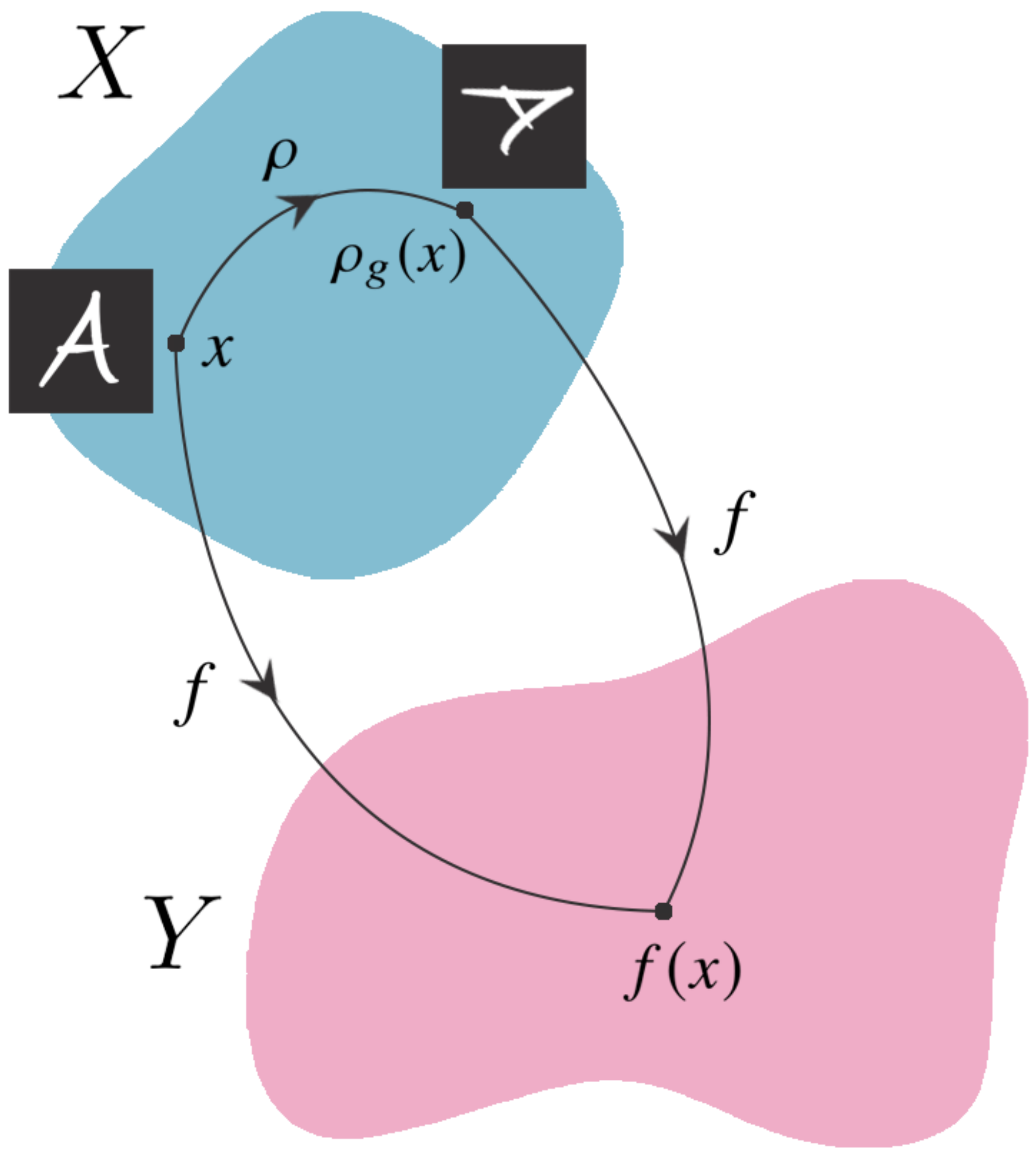
INDUCTIVE BIAS & EQUIVARIANCE

- Symmetry-equivariant networks
  - More economical (fewer, but more expressive parameters), interpretable, and trainable



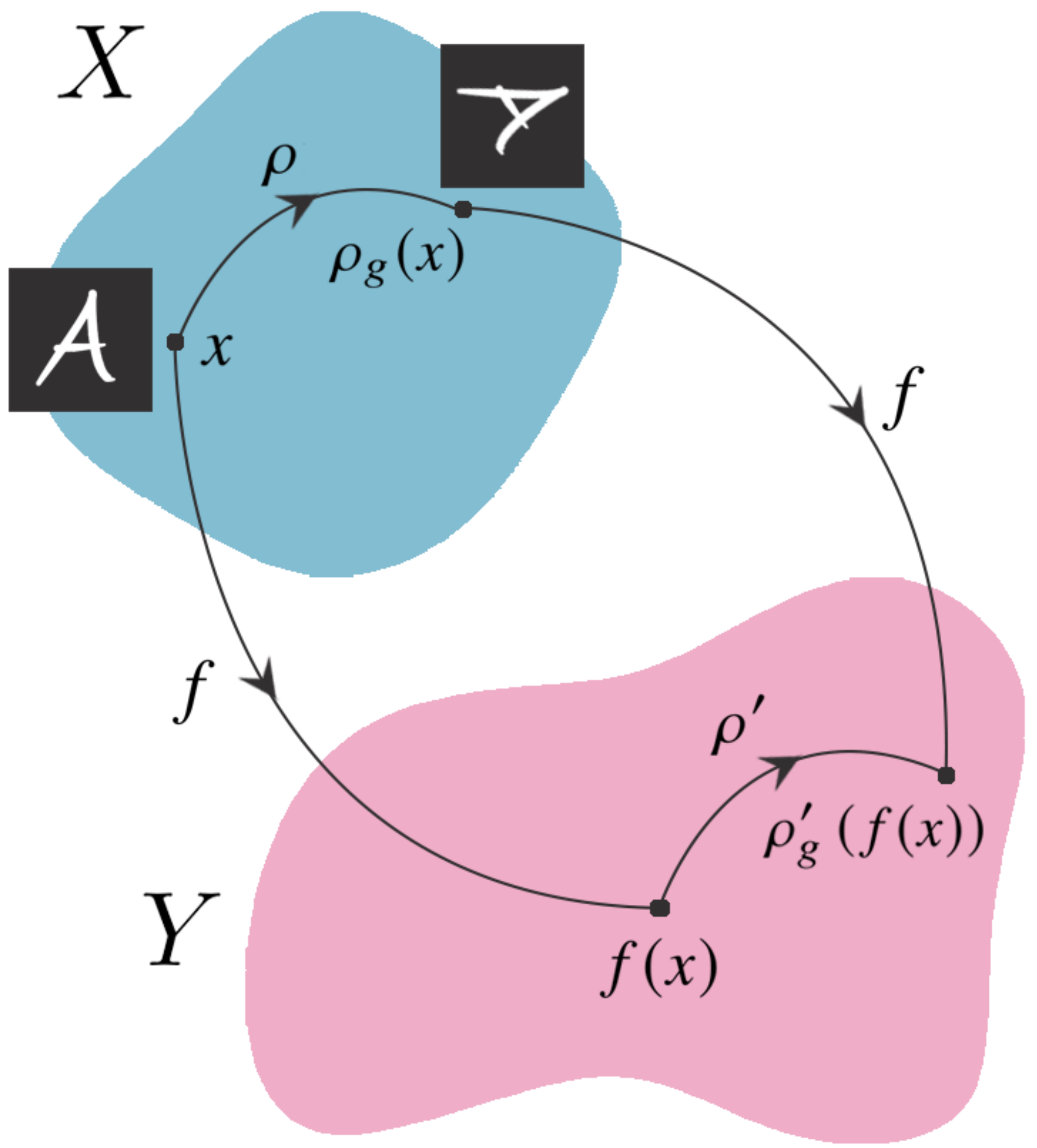
Invariance

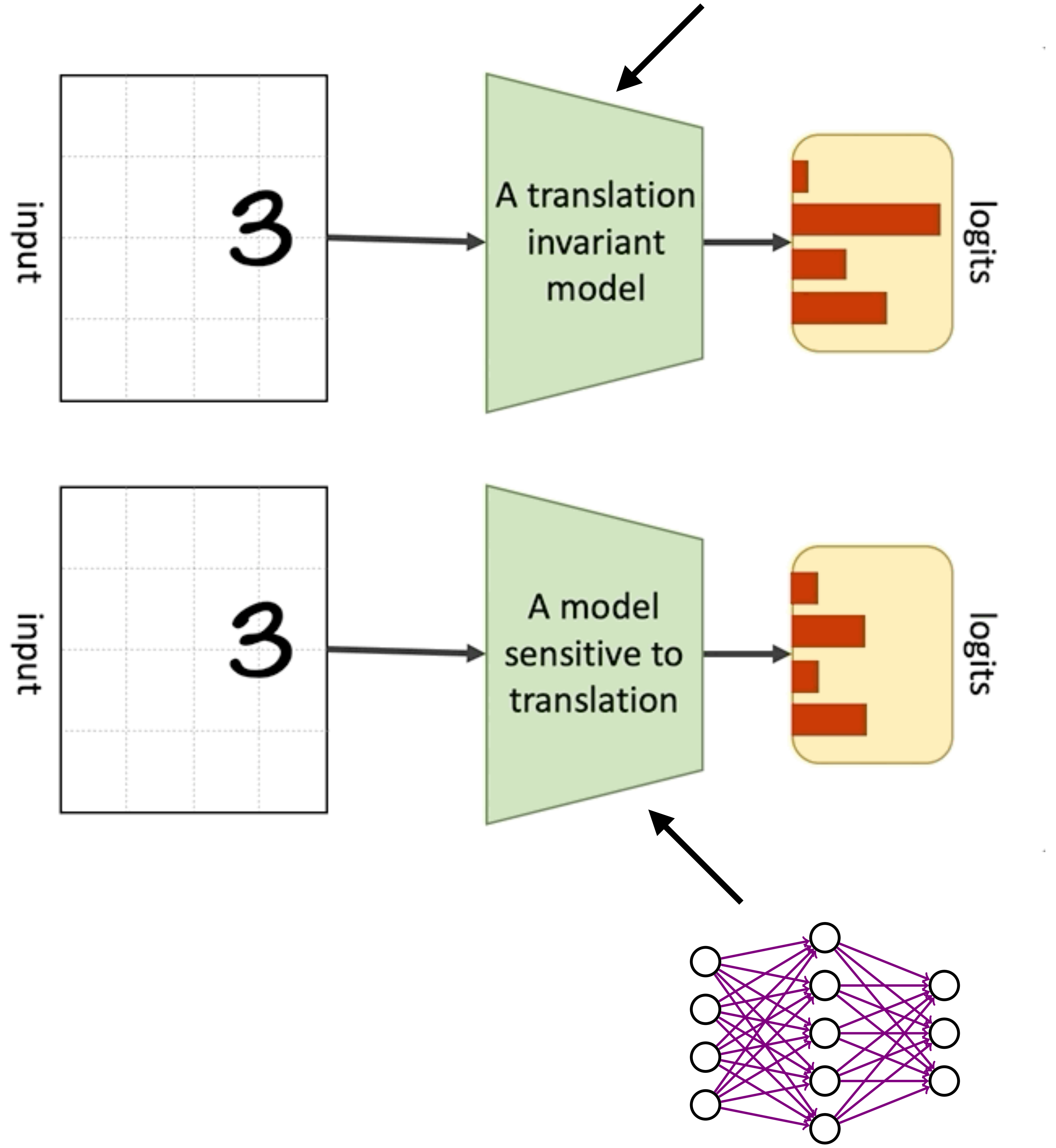
$$f(\rho_g(x)) = f(x)$$



Equivariance

$$f(\rho_g(x)) = \rho'_g\,(f(x))$$



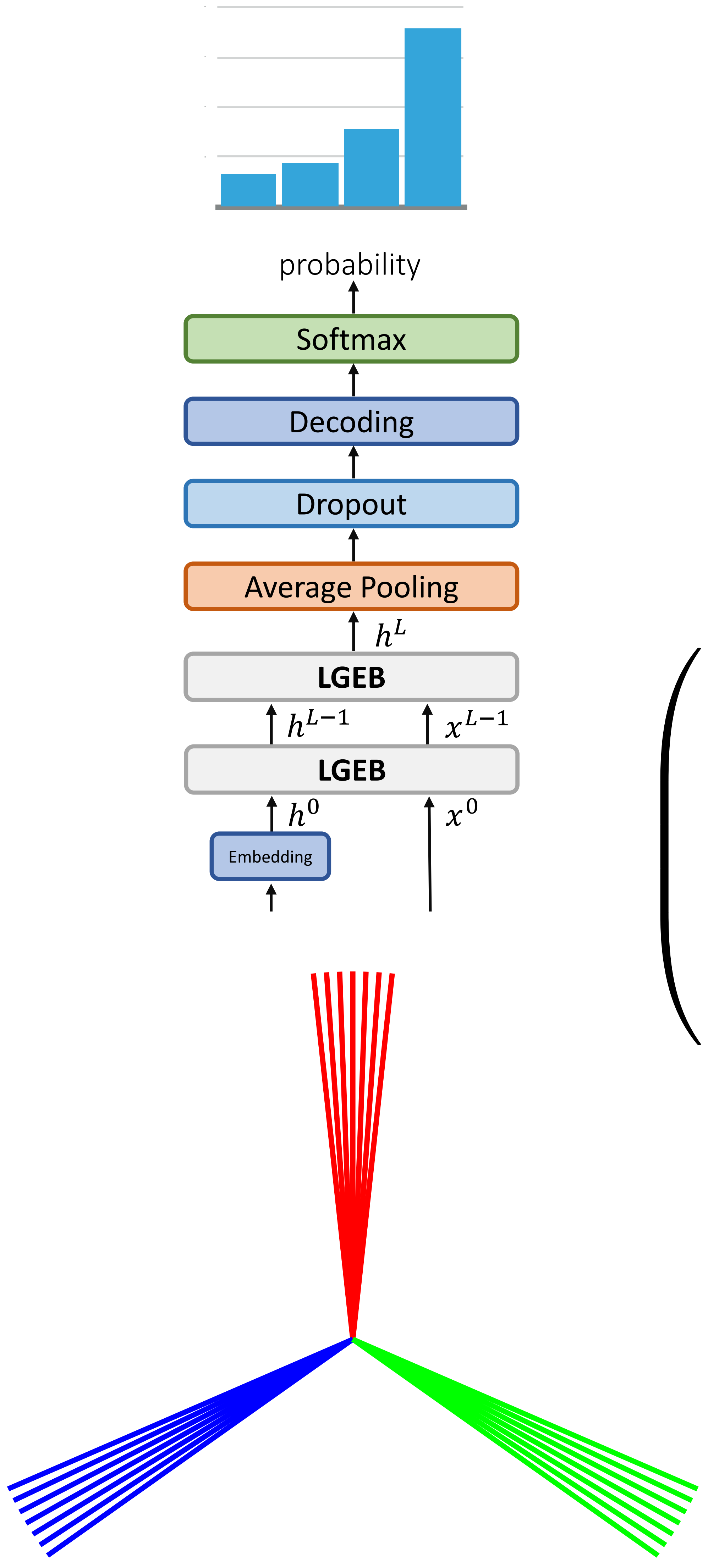


A translation invariant model

A model sensitive to translation

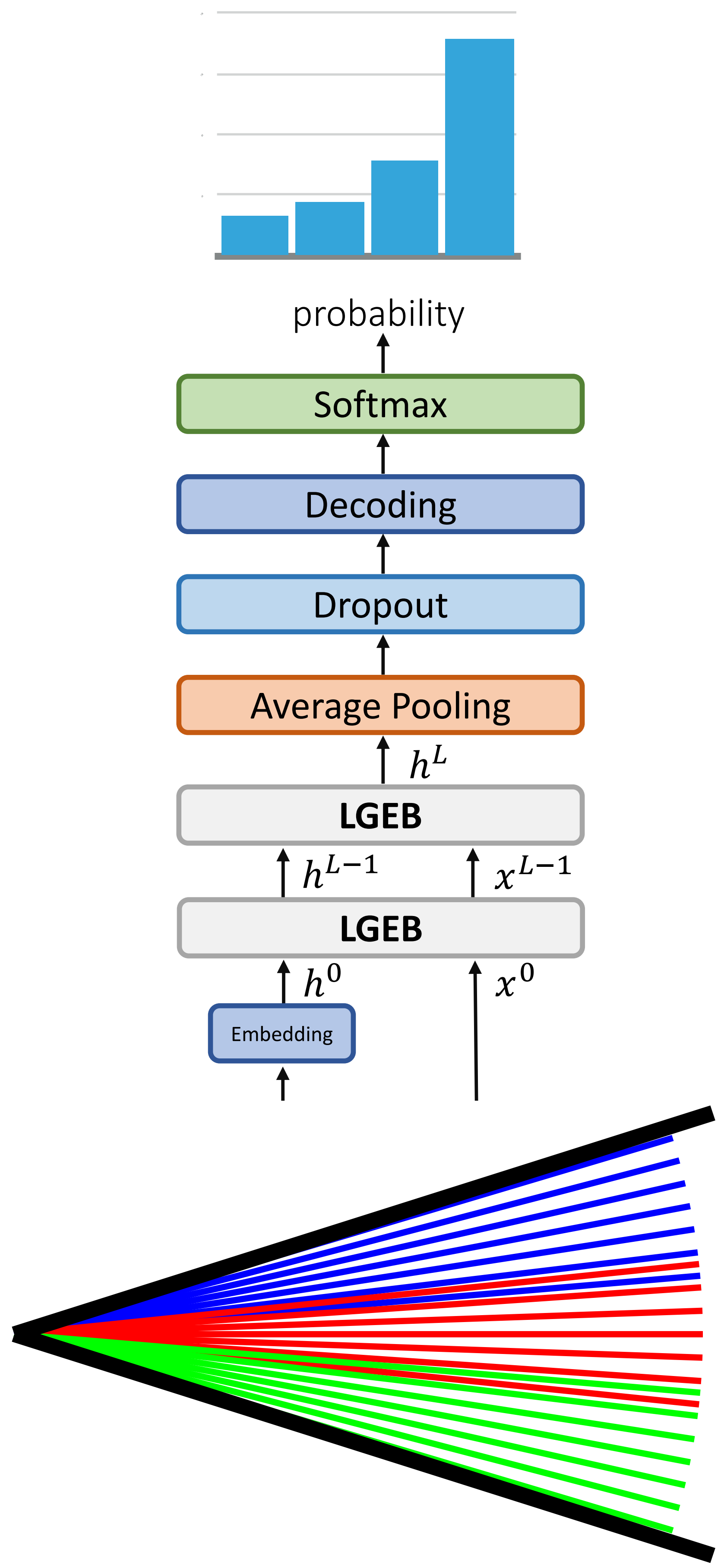
HOW DO WE ENFORCE LORENTZ SYMMETRY?

- Lorentz-invariant networks:
  - Boosting all particles into a new frame should give the same result
- Lorentz-equivariant networks:
  - Boosting all particles into a new frame should give an output that transforms the same way



$$\begin{pmatrix} \gamma & -\gamma\beta & 0 & 0 \\ -\gamma\beta & \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Lorentz boost →

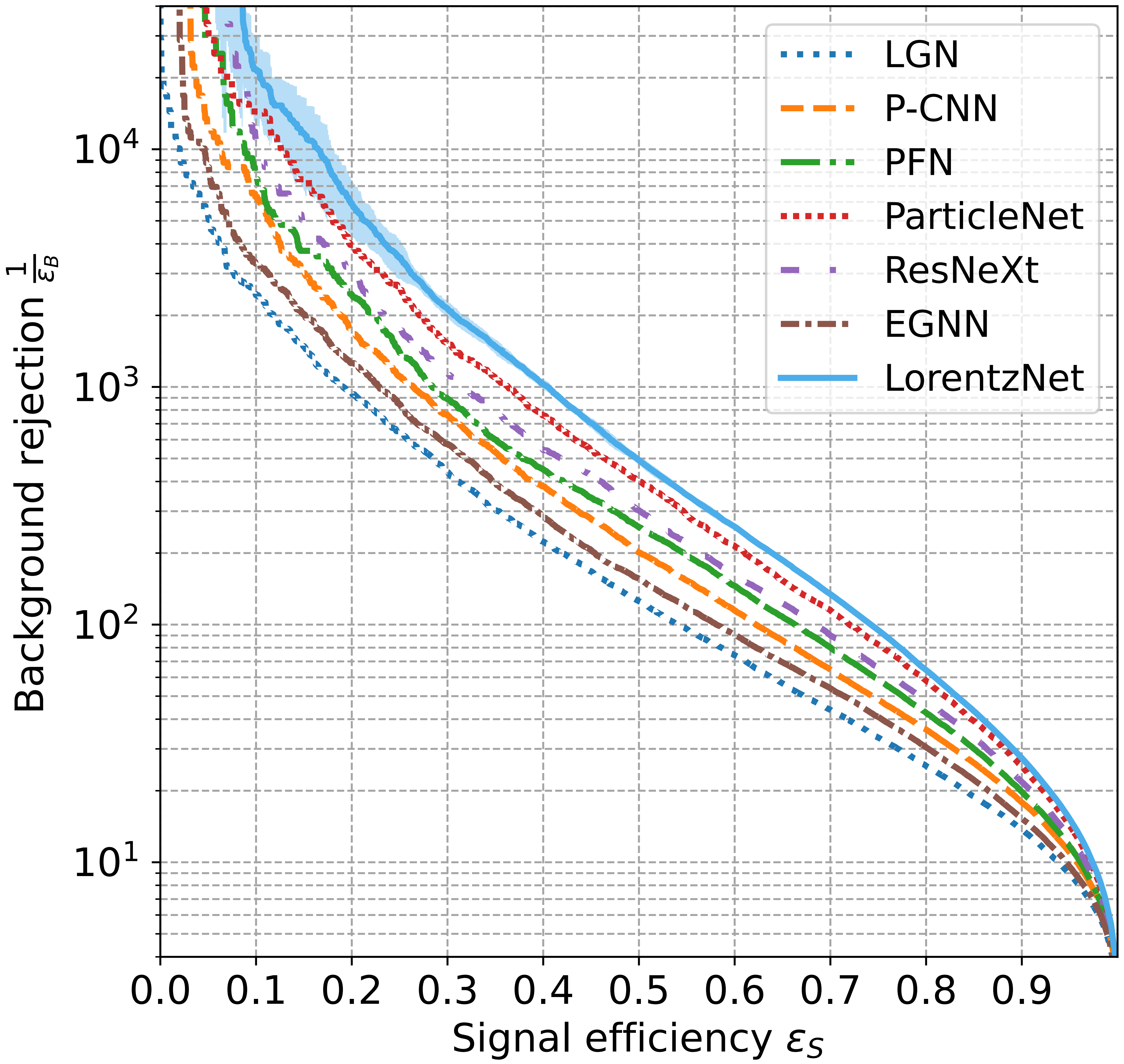


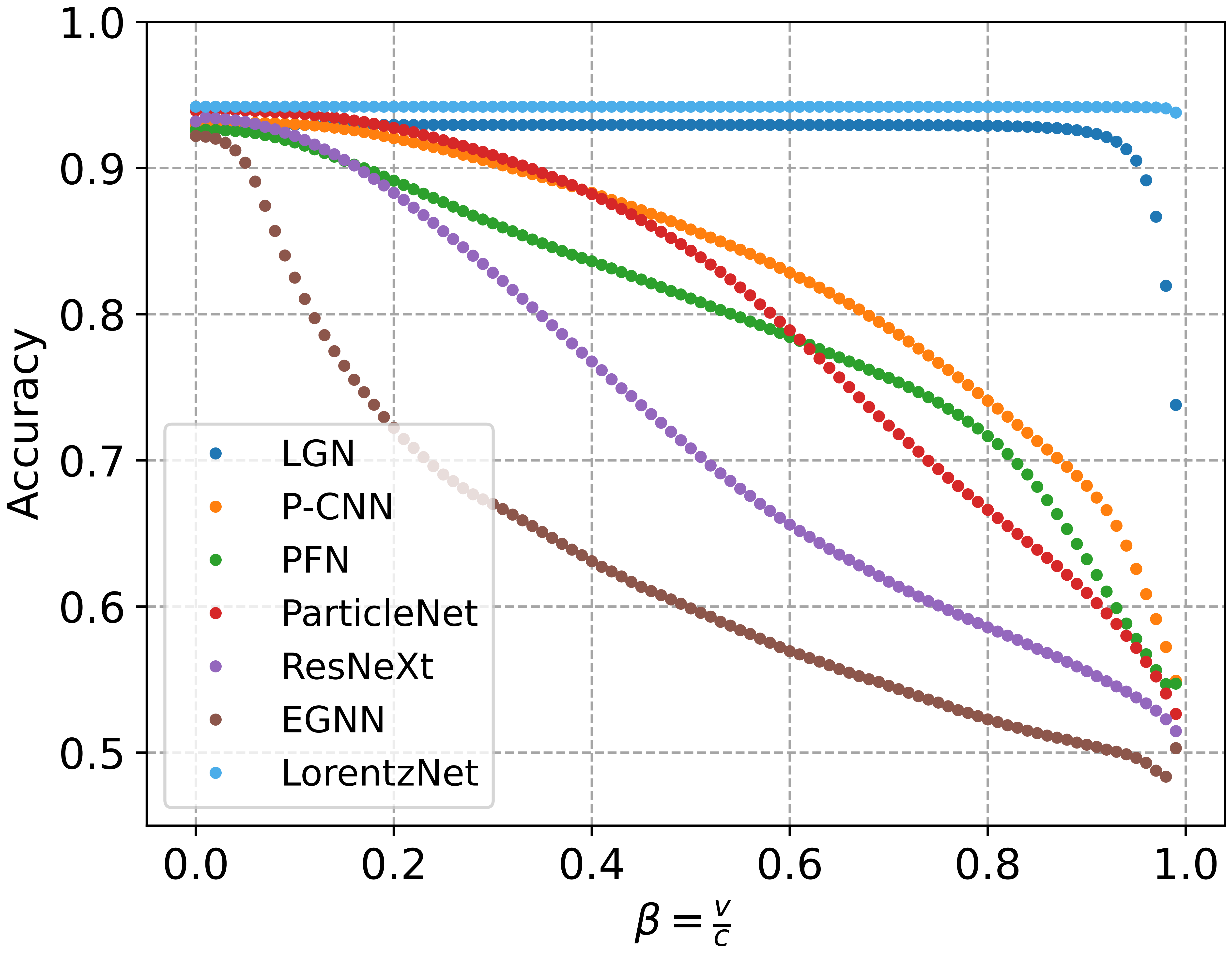
arXiv:2201.08187

LORENTZNET PERFORMANCE

arXiv:2201.08187

- State-of-the-art performance for top quark tagging
- Lorentz group invariance confirmed





I. BASICS

II. DATA REPRESENTATIONS & SYMMETRIES

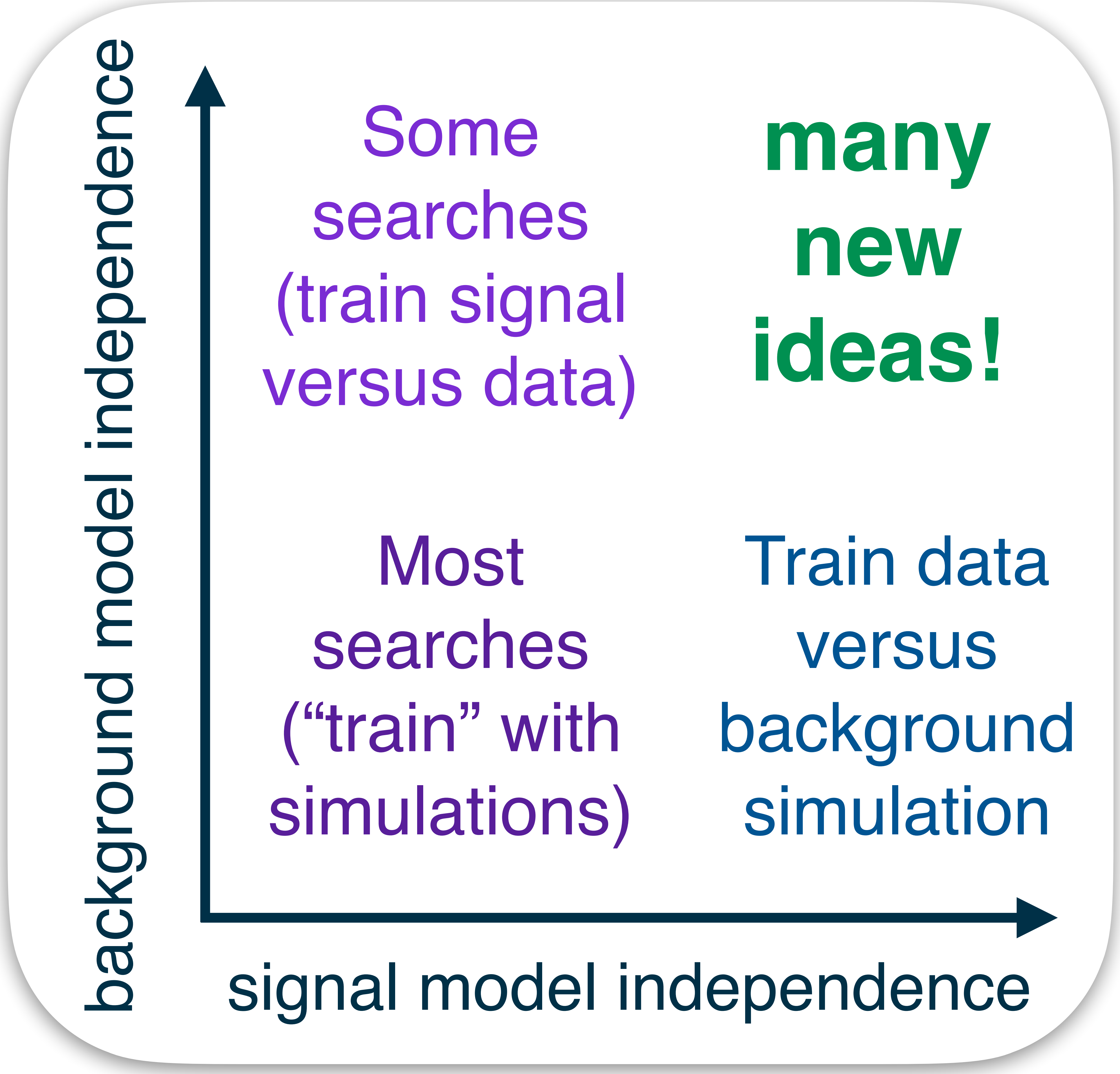
III. ANOMALY DETECTION

IV. GENERATIVE MODELING

V. SUMMARY & OUTLOOK

LANDSCAPE OF BSM SEARCHES

Credit: B. Nachman
https://indico.cern.ch/event/1188153/



background model independence

Some searches (train signal versus data)

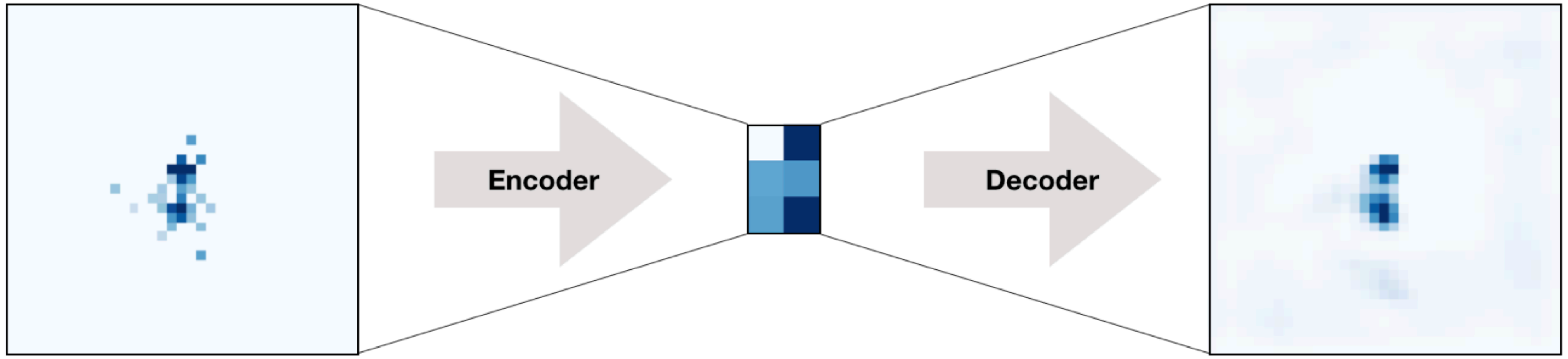
many new ideas!

Most searches ("train" with simulations)

Train data versus background simulation

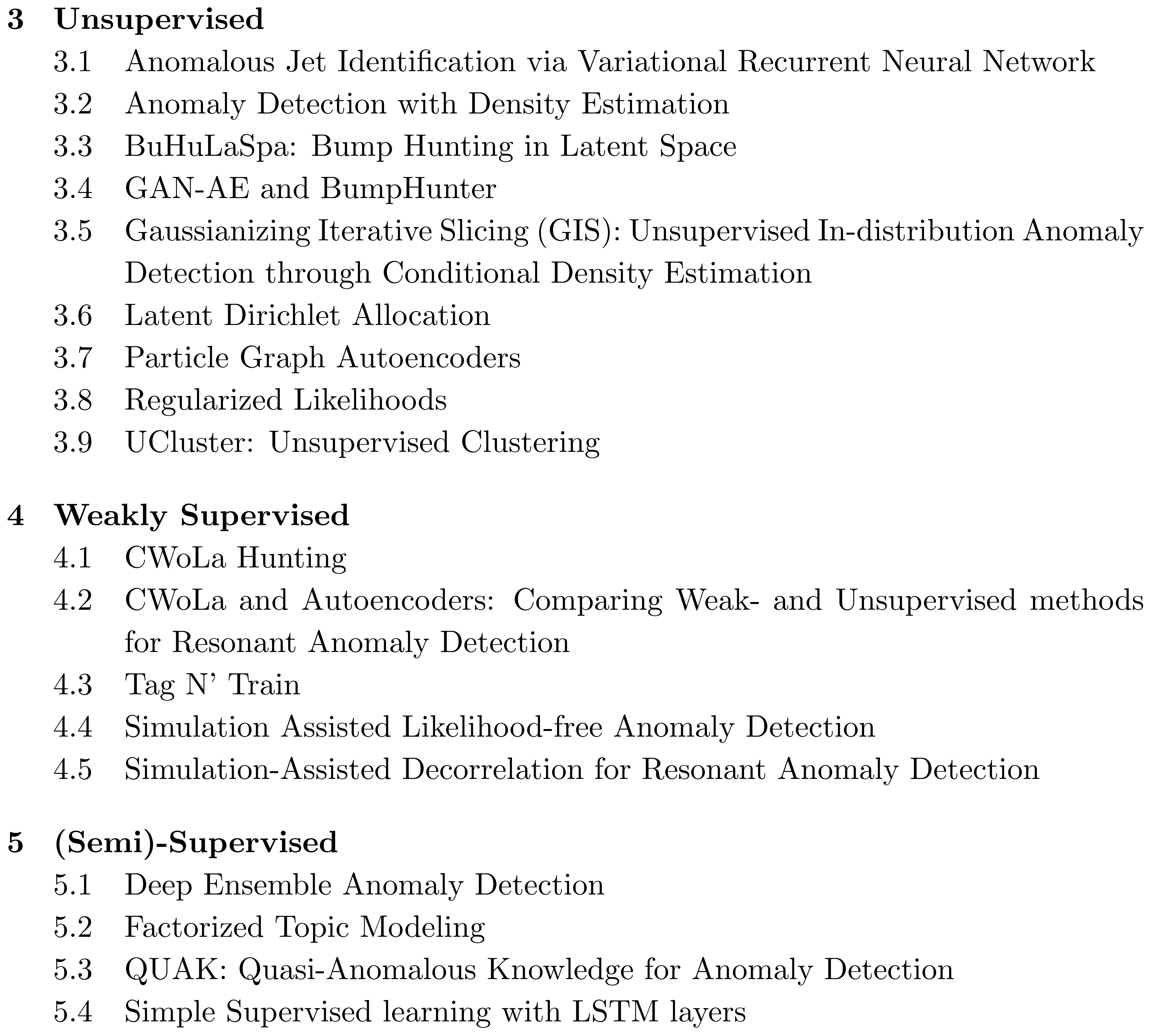
signal model independence

- Supervised = full label information
- Semi-supervised = partial labels
- Weakly-supervised = noisy labels
- Unsupervised = no labels
  - Example: autoencoders compress data and then uncompress it
  - Assumption: if x is far from $\mathrm{Decoder}(\mathrm{Encoder}(x))$, then x has low $p_{\mathrm{bkgd}}(x)$



LHC OLYMPICS 2020

- Challenge with “black box” signals run in 2020–2021
- Plethora of new techniques

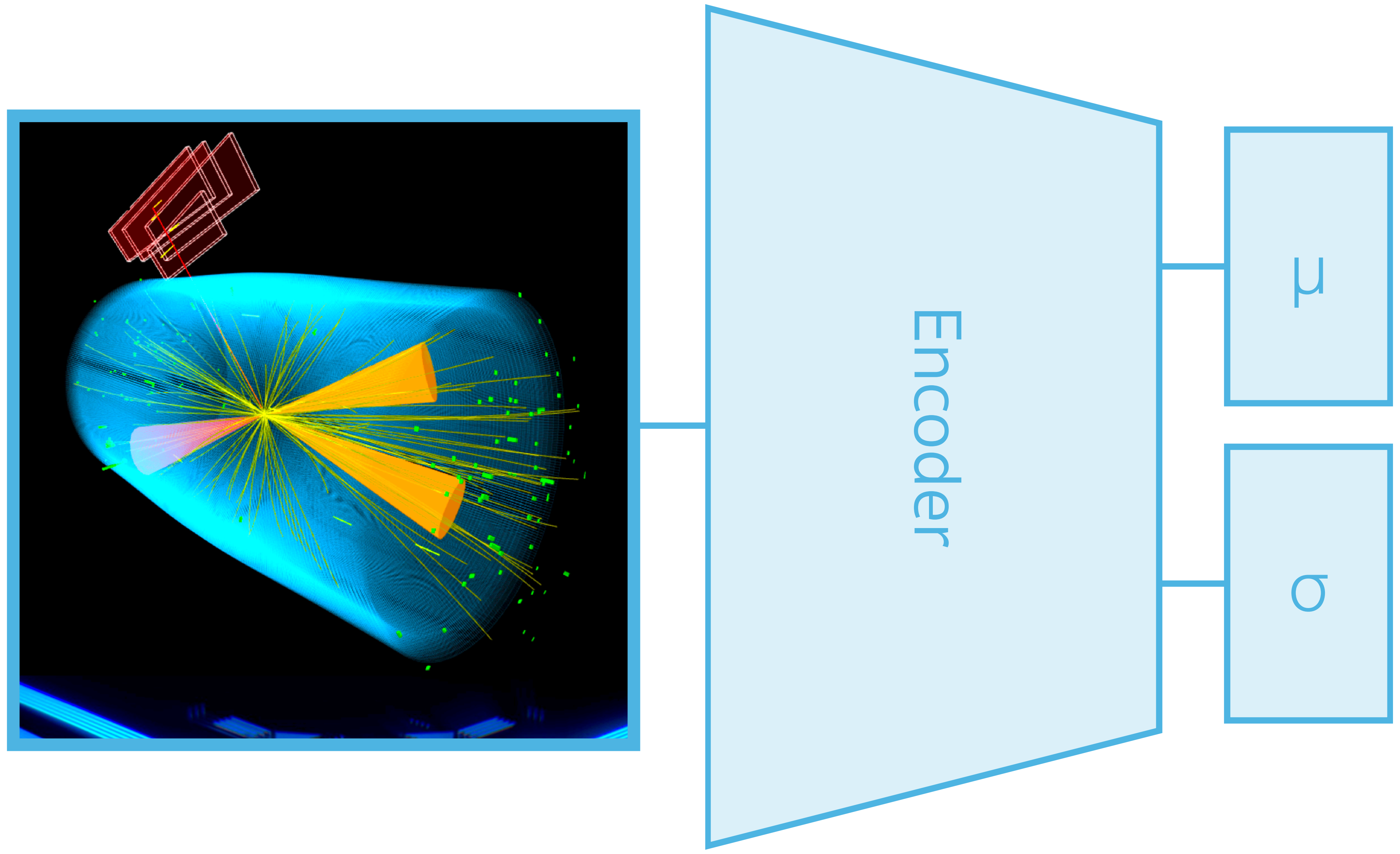


APPLICATION: ANOMALY DETECTION AT 40 MHZ

arXiv:2108.03986

Data challenge: mpp-hep.github.io/ADC2021

- Challenge: if new physics has an unexpected signature that doesn't align with existing triggers, precious BSM events may be discarded at trigger level
- Can we use unsupervised algorithms to detect non-SM-like anomalies?
  - Autoencoders (AEs): compress input to a smaller dimensional latent space then decompress and calculate difference
  - Variational autoencoders (VAEs): model the latent space as a probability distribution; possible to detect anomalies purely with latent space variables



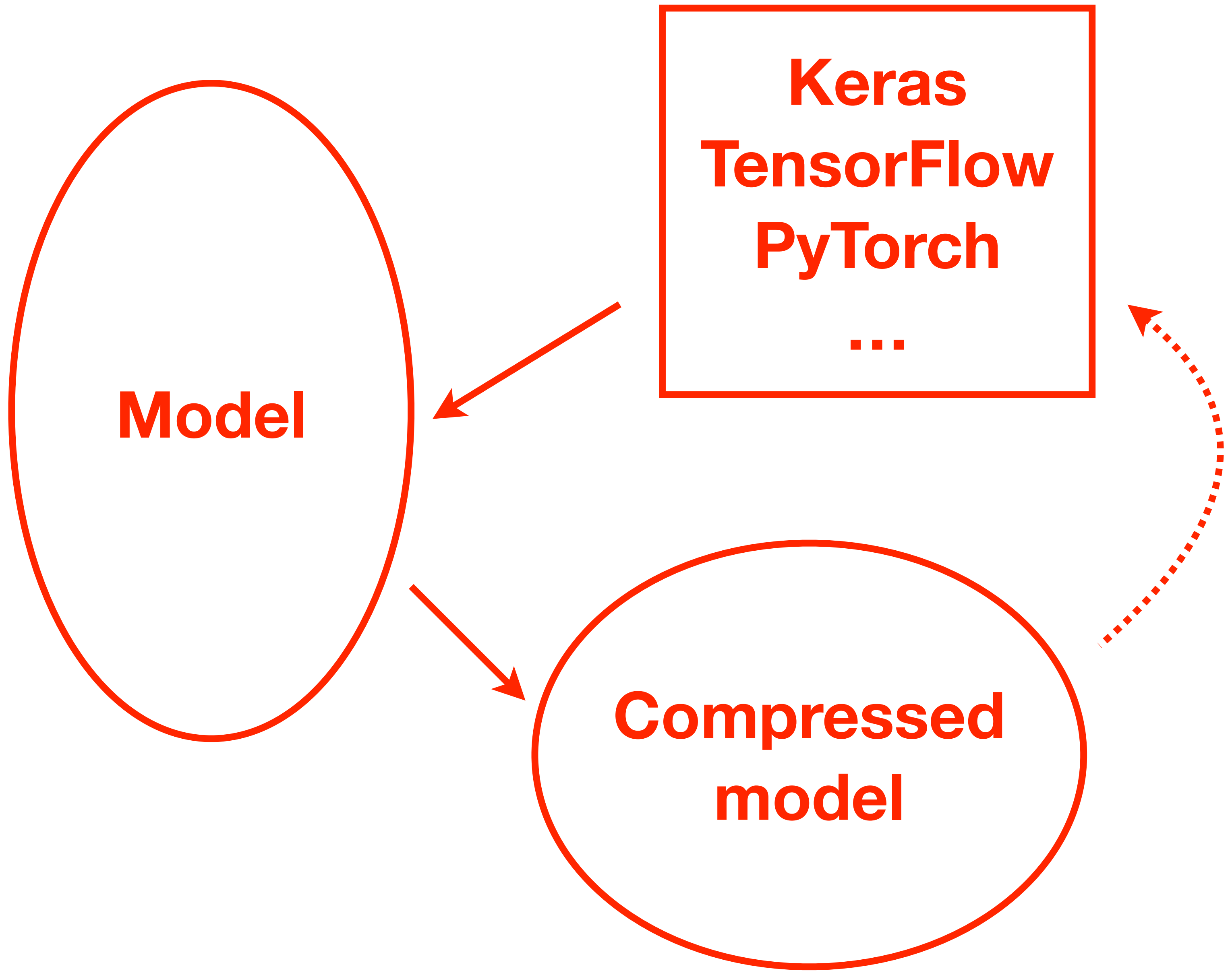
Key observation: Can build an anomaly score from the latent space of VAE directly! No need to run decoder!

$$R_z = \sum_i \frac{\mu_i^2}{\sigma_i^2}$$

DESIGN EXPLORATION WITH HLS4ML

J. Instrum. 13, P07027 (2018)

- hls4ml for scientists or ML experts to translate ML algorithms into RTL firmware



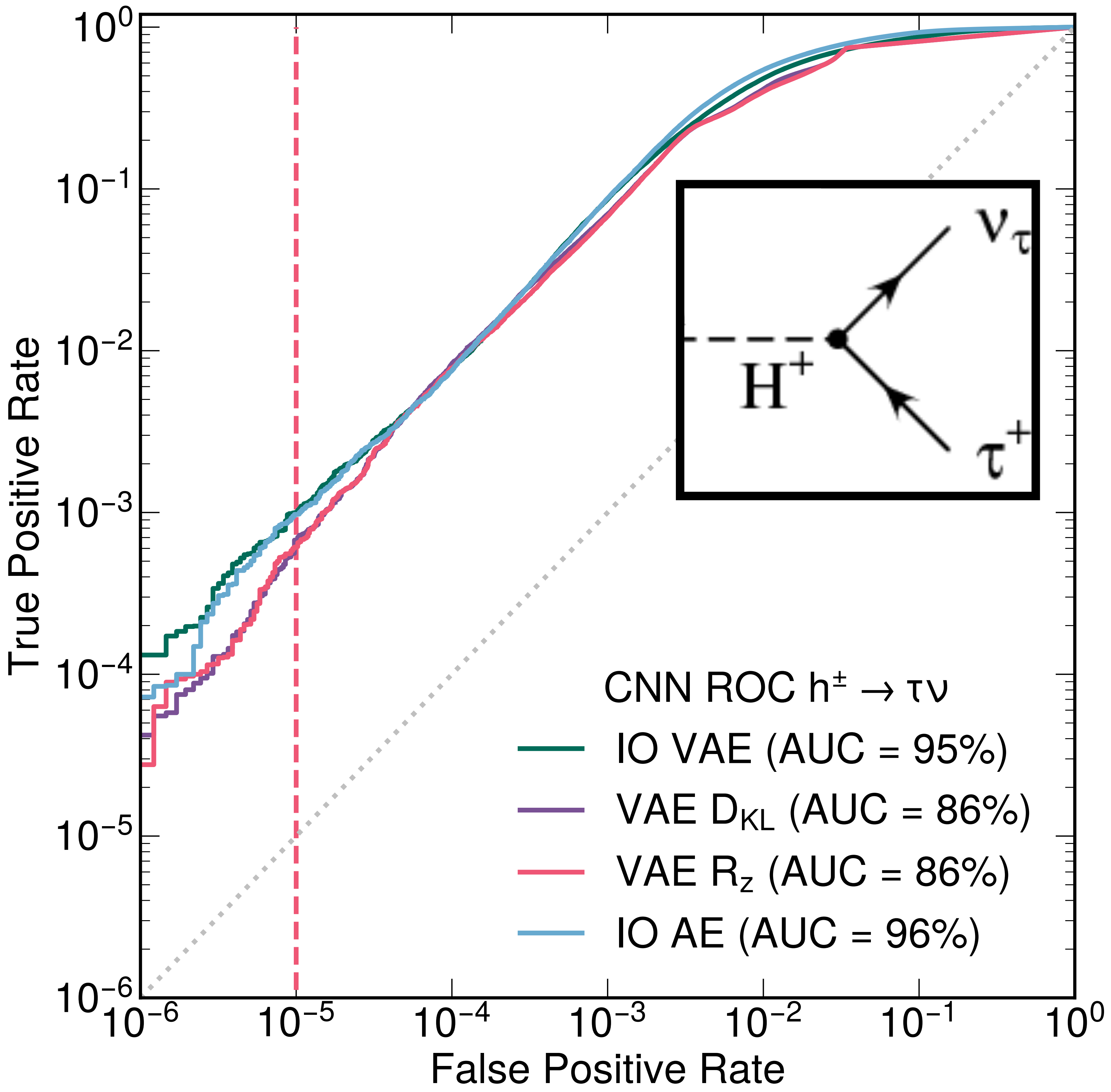
Machine learning model optimization, compression

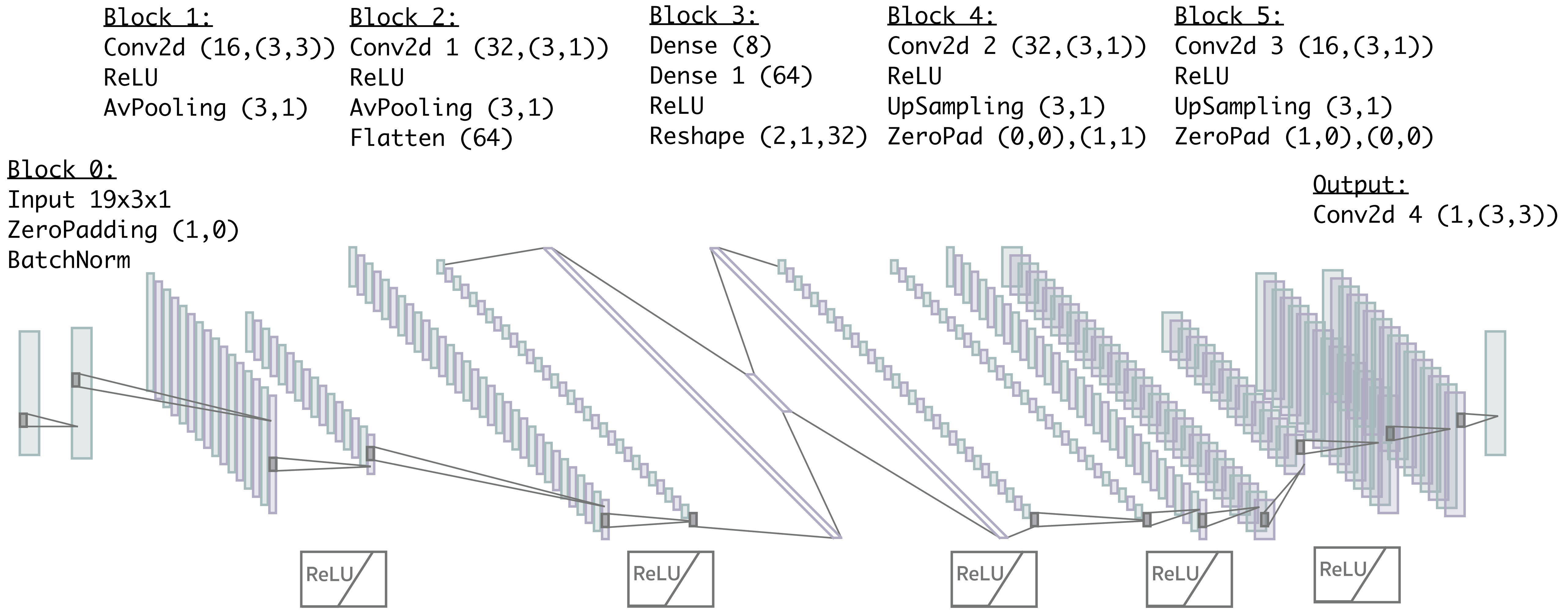
FPGA IMPLEMENTATION

arXiv:2108.03986

Data challenge: mpp-hep.github.io/ADC2021

- CNNs as the basis for (V)AEs for anomaly detection
- Good anomaly detection performance for unseen signals (LQ → bτ, A → 4l, **h± → τν**, h^0 → ττ)
- **VAE** fits in latency and resource requirements for HL-LHC!





Block 0:
Input 19x3x1
ZeroPadding (1,0)
BatchNorm

Block 1:
Conv2d (16,(3,3))
ReLU
AvPooling (3,1)

Block 2:
Conv2d 1 (32,(3,1))
ReLU
AvPooling (3,1)
Flatten (64)

Block 3:
Dense (8)
Dense 1 (64)
ReLU
Reshape (2,1,32)

Block 4:
Conv2d 2 (32,(3,1))
ReLU
UpSampling (3,1)
ZeroPad (0,0),(1,1)

Block 5:
Conv2d 3 (16,(3,1))
ReLU
UpSampling (3,1)
ZeroPad (1,0),(0,0)

Output:
Conv2d 4 (1,(3,3))

Model	DSP [%]	LUT [%]	FF [%]	BRAM [%]	Latency [ns]	II [ns]	AUC [%]	TPR @ FPR=10^{-5}
CNN VAE R_z	10	**12**	**4**	**2**	**365**	**115**	86	0.06%
CNN AE	**7**	47	5	6	1480	895	**96**	**0.10%**

I. BASICS

II. DATA REPRESENTATIONS & SYMMETRIES

III. ANOMALY DETECTION

IV. GENERATIVE MODELING

V. SUMMARY & OUTLOOK

ML4SIM STRATEGIES

- Several different strategies:
  - Replace (part of) FullSim: increase speed, preserve accuracy
  - Replace (part of) FastSim: maintain speed, increase accuracy
  - Conditional: map generated → reconstructed events
  - **End-to-end: map random noise → reconstructed events directly**

END-TO-END

CONDITIONAL

Full Simulation

FULL DETECTOR SIM W/ ML

DIGITIZATION EMULATION

RECONSTRUCTION

HARD PROCESS GENERATION

SHOWERING/ HADRONIZATION/ UNDERLYING EVT

Fast Simulation

APPROXIMATE DETECTOR SIM W/ ML

PARTIAL DIGITIZATION EMULATION

SIMPLIFIED RECONSTRUCTION

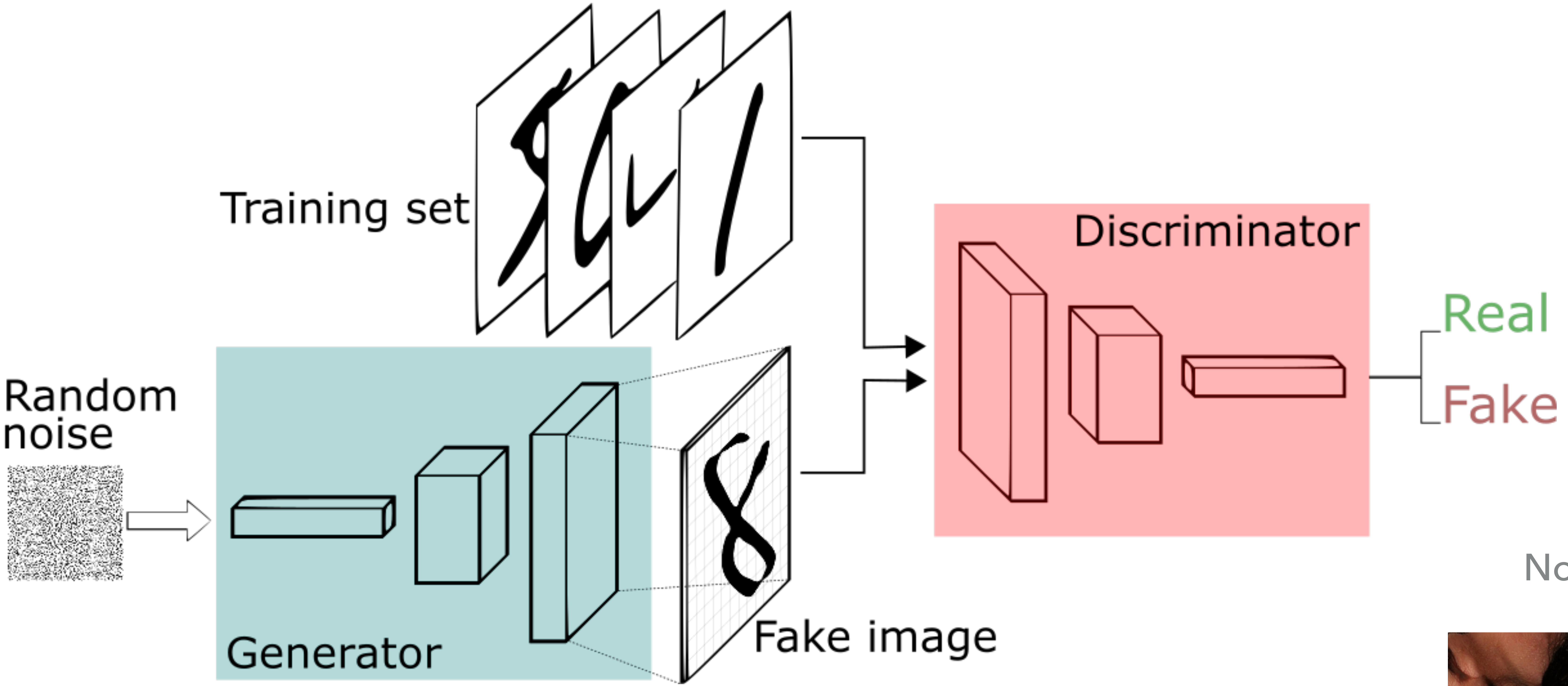
ANALYSIS/ NTUPLING

Delphes

PARAMETRIZED SMEARING

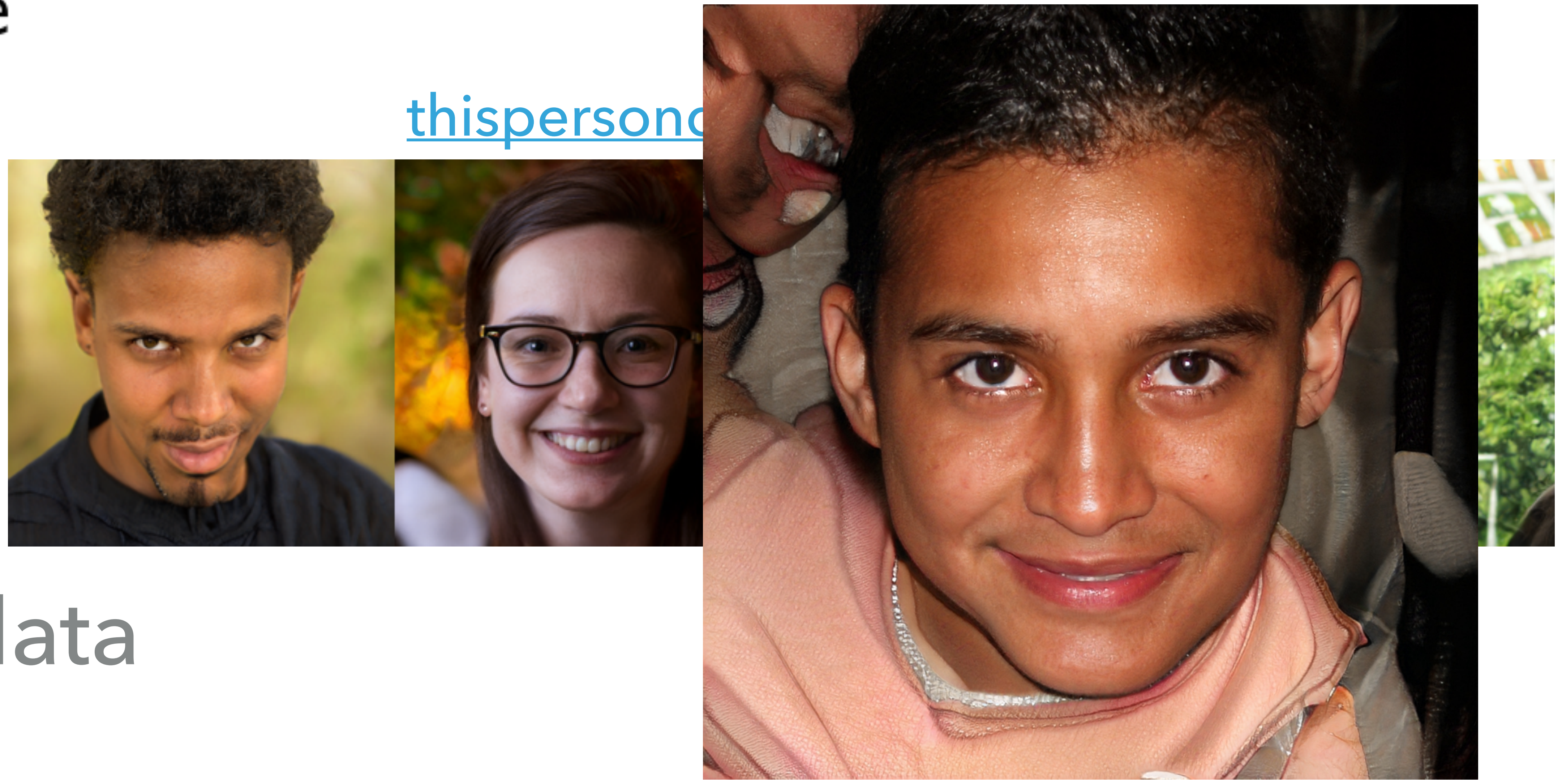
GENERATIVE ADVERSARIAL NETWORKS

arXiv:1406.2661
arXiv:1912.04958



Note: failure modes!

thispersond



- Train two neural networks in tandem:
  - one to generate realistic “fake” data
  - the other to discriminate “real” from “fake” data

GENERATIVE AI EVALUATION METRICS

arXiv:2012.00173
arXiv:2106.11535

- Evaluation of generative models is in general difficult
- We want to evaluate quantitatively:
  - the **quality** of the data
  - the **diversity** of the data
  - ultimately, **physics performance**

	Minimum Matching Distance	Coverage	Fréchet ParticleNet Distance	1-Wassersstein Distance (W_1)
Quality	✅		✅	✅
Diversity		✅	✅	✅
Physics Perf.				✅

MESSAGE-PASSING GAN ARCHITECTURE

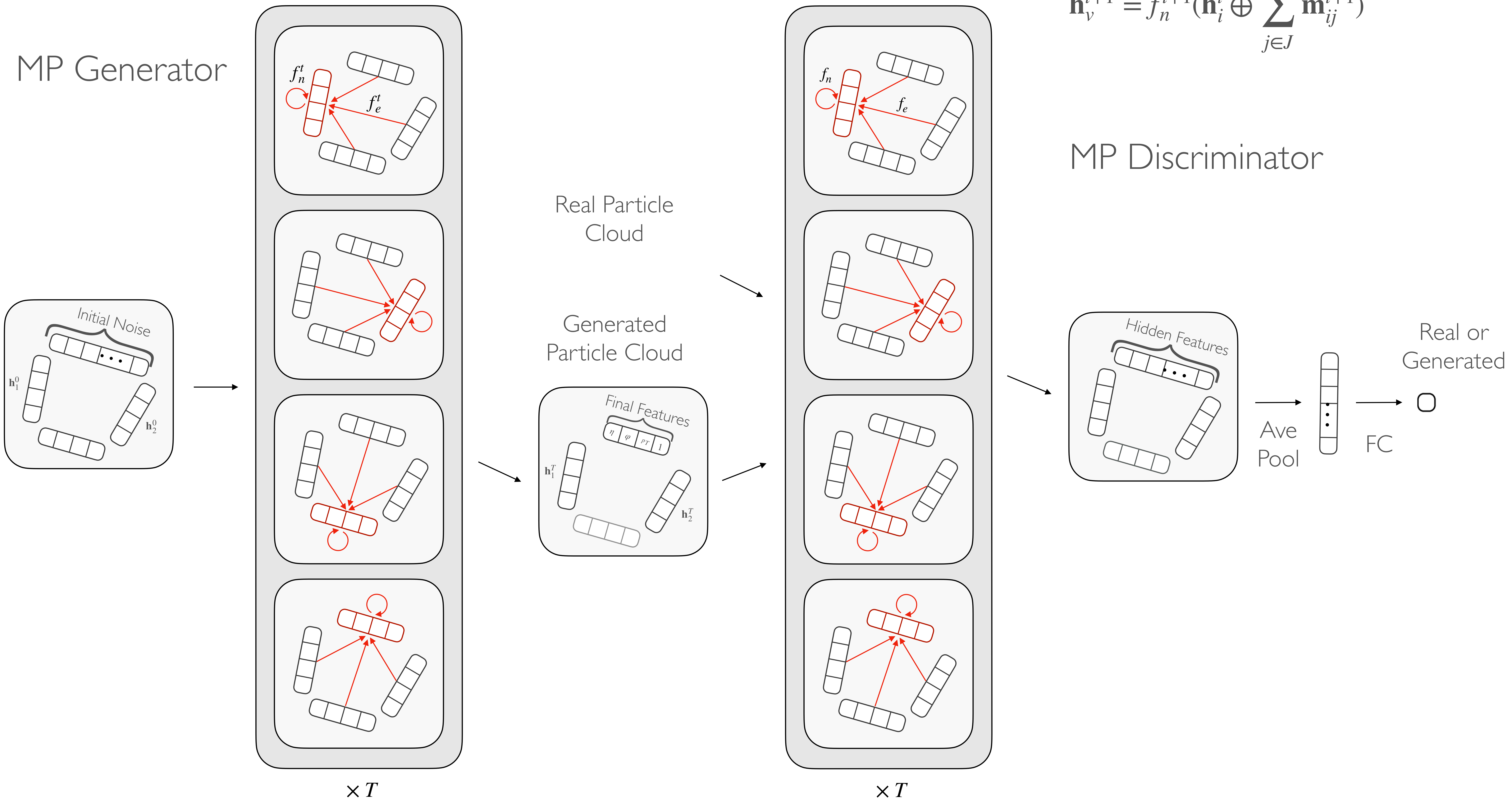
arXiv:2012.00173

arXiv:2106.11535

- As an alternative to voxelization, a graph-based GAN can be used to generate jets as particle clouds

$$\mathbf{m}_{ij}^{t+1} = f_e^{t+1}(\mathbf{h}_i^t \oplus \mathbf{h}_j^t)$$

$$\mathbf{h}_v^{t+1} = f_n^{t+1}(\mathbf{h}_i^t \oplus \sum_{j \in J} \mathbf{m}_{ij}^{t+1})$$

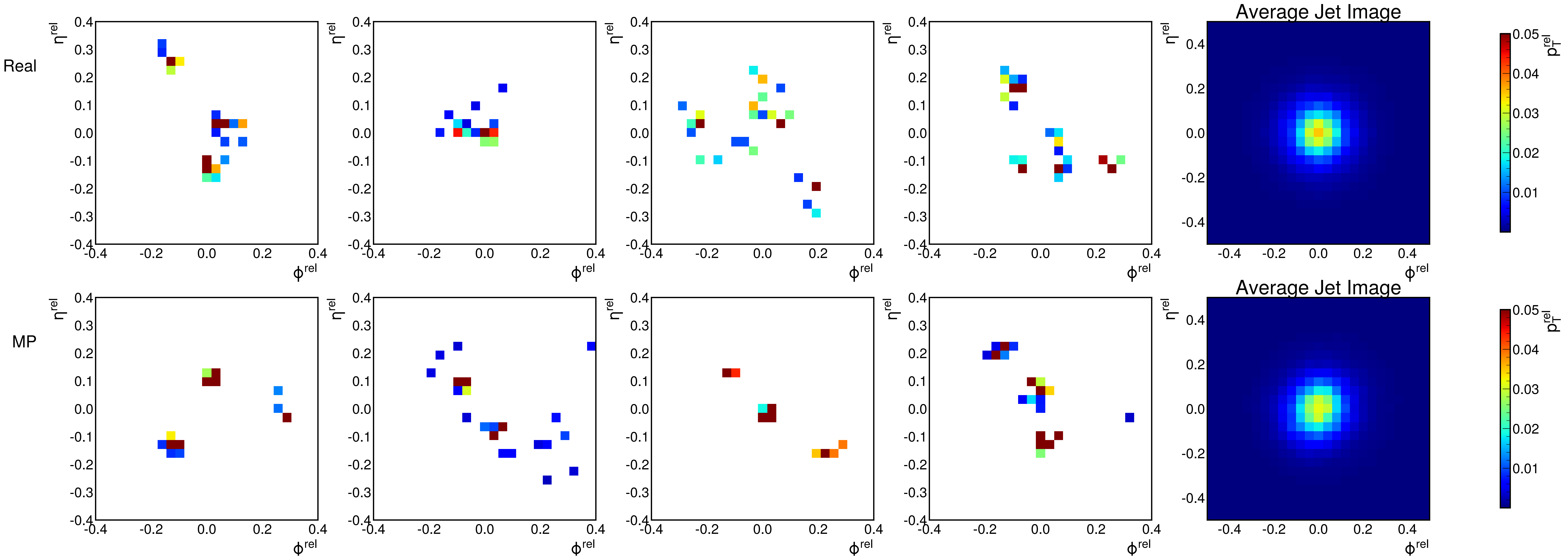


QUALITATIVE TOP QUARK JET RESULTS

arXiv:2012.00173

arXiv:2106.11535

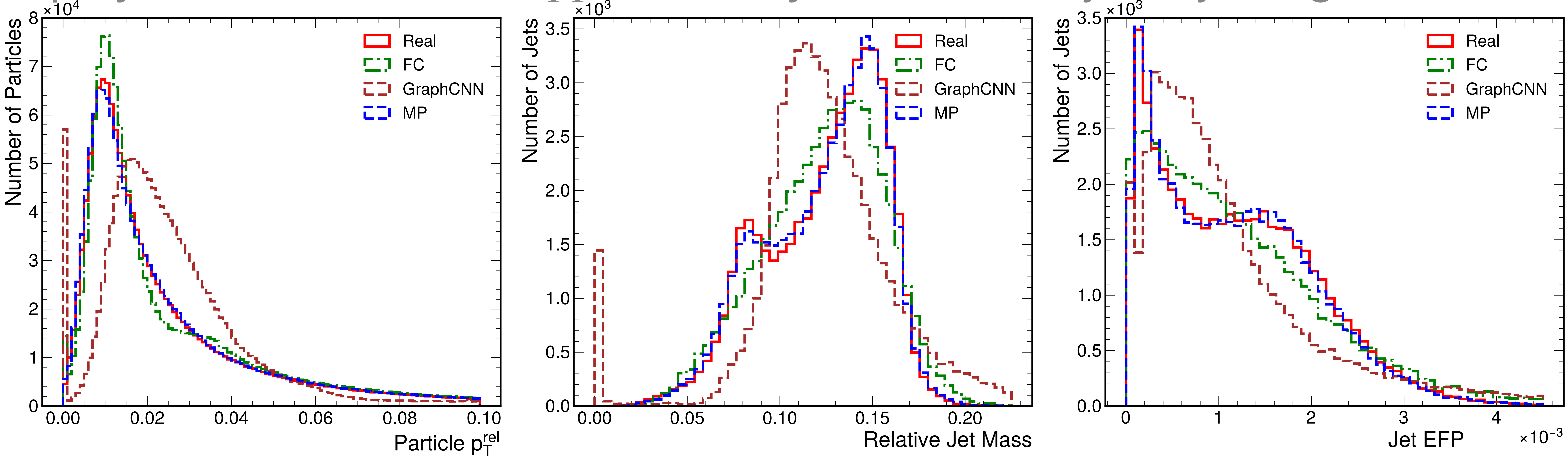
- To easily visualize the generated particle clouds, we can make “jet images”



QUANTITATIVE TOP QUARK JET RESULTS

arXiv:2012.00173
arXiv:2106.11535

- Reproduces nontrivial properties like top quark jet mass and energy-flow polynomials ($t \to Wb \to qqb$ so 3 subjets + not always fully merged)

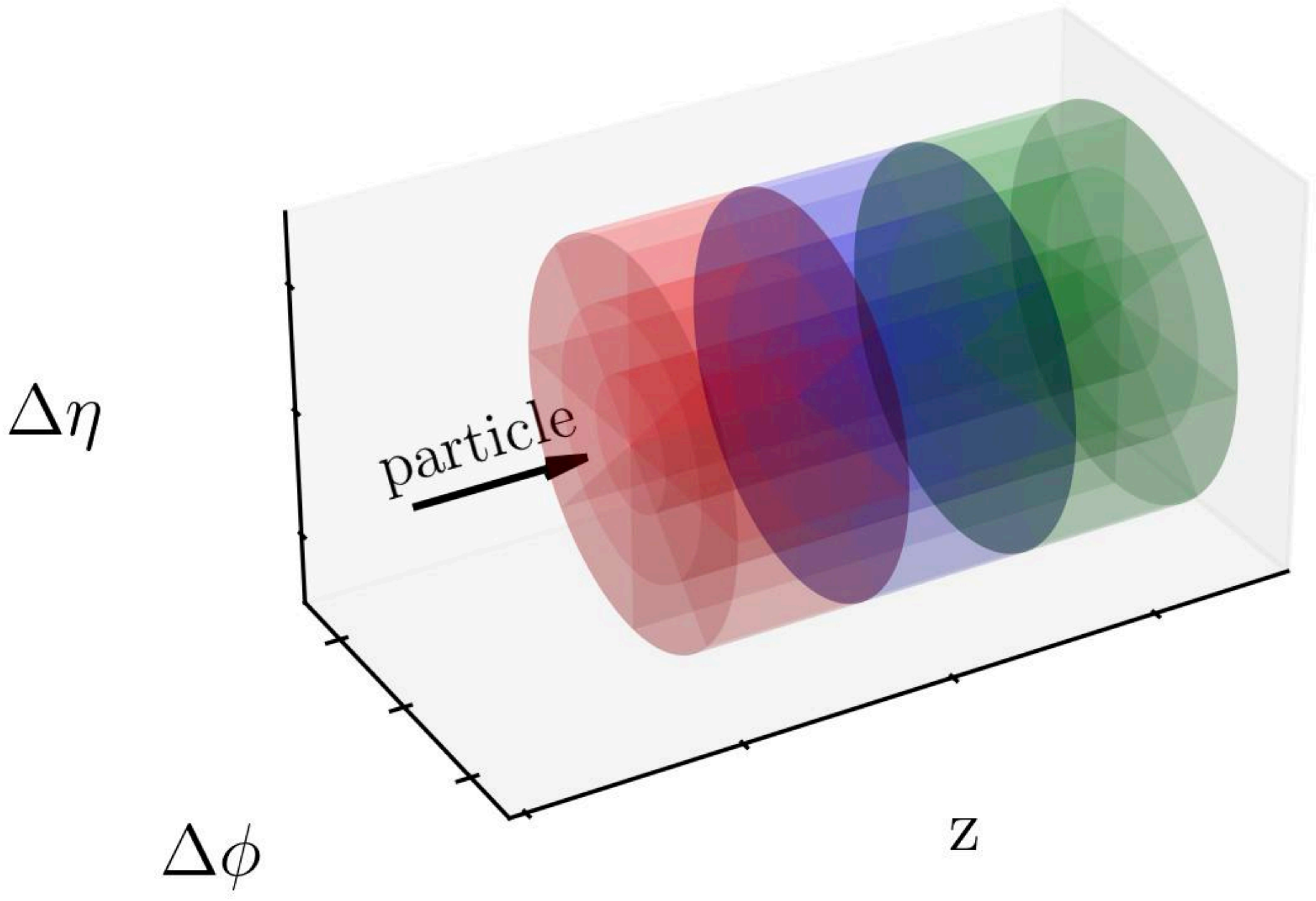


Generator	Discriminator	W_1-P (10^{-3})	W_1-M (10^{-3})	W_1-EFP (10^{-5})	FPND	Coverage	MMD	Time
	Real	**0.55 ± 0.07**	**0.51 ± 0.07**	**1.1 ± 0.1**	–	–	–	**O(1 s) per jet**
FC	PointNet	**1.5 ± 0.2**	2.6 ± 0.1	8 ± 3	225	0.56	0.076	
GraphCNN	PointNet	37 ± 3	10.8 ± 0.5	39 ± 18	2M	0.39	0.084	
MP	**MP**	2.1 ± 0.2	**0.7 ± 0.1**	**1.8 ± 0.8**	**6.4**	0.56	**0.071**	**36 μs per jet**
MP	PointNet	**1.5 ± 0.1**	1.0 ± 0.3	5 ± 2	12	**0.58**	**0.071**	

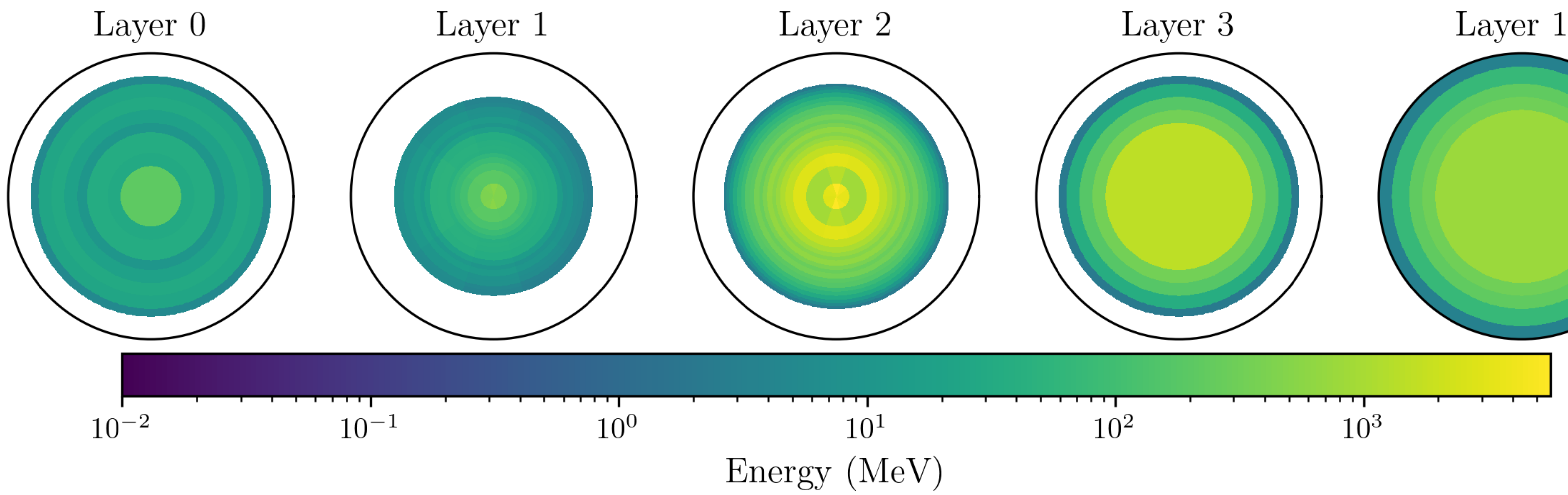
CALO CHALLENGE

calochallenge.github.io

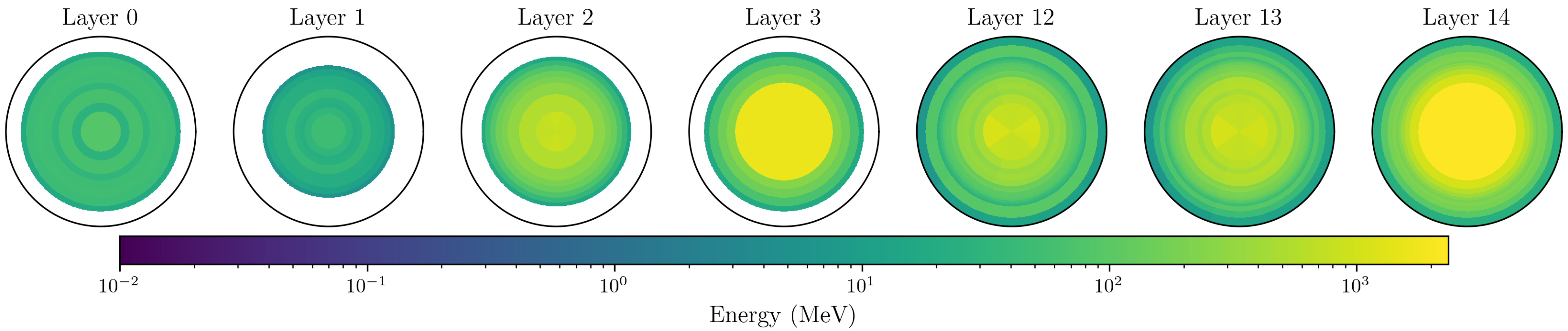
- Ongoing challenge for generative modeling of calorimeter showers in HEP!
- Many new approaches presented at CaloChallenge Workshop: https://agenda.infn.it/event/34036/



Shower average GEANT4 photon reference dataset



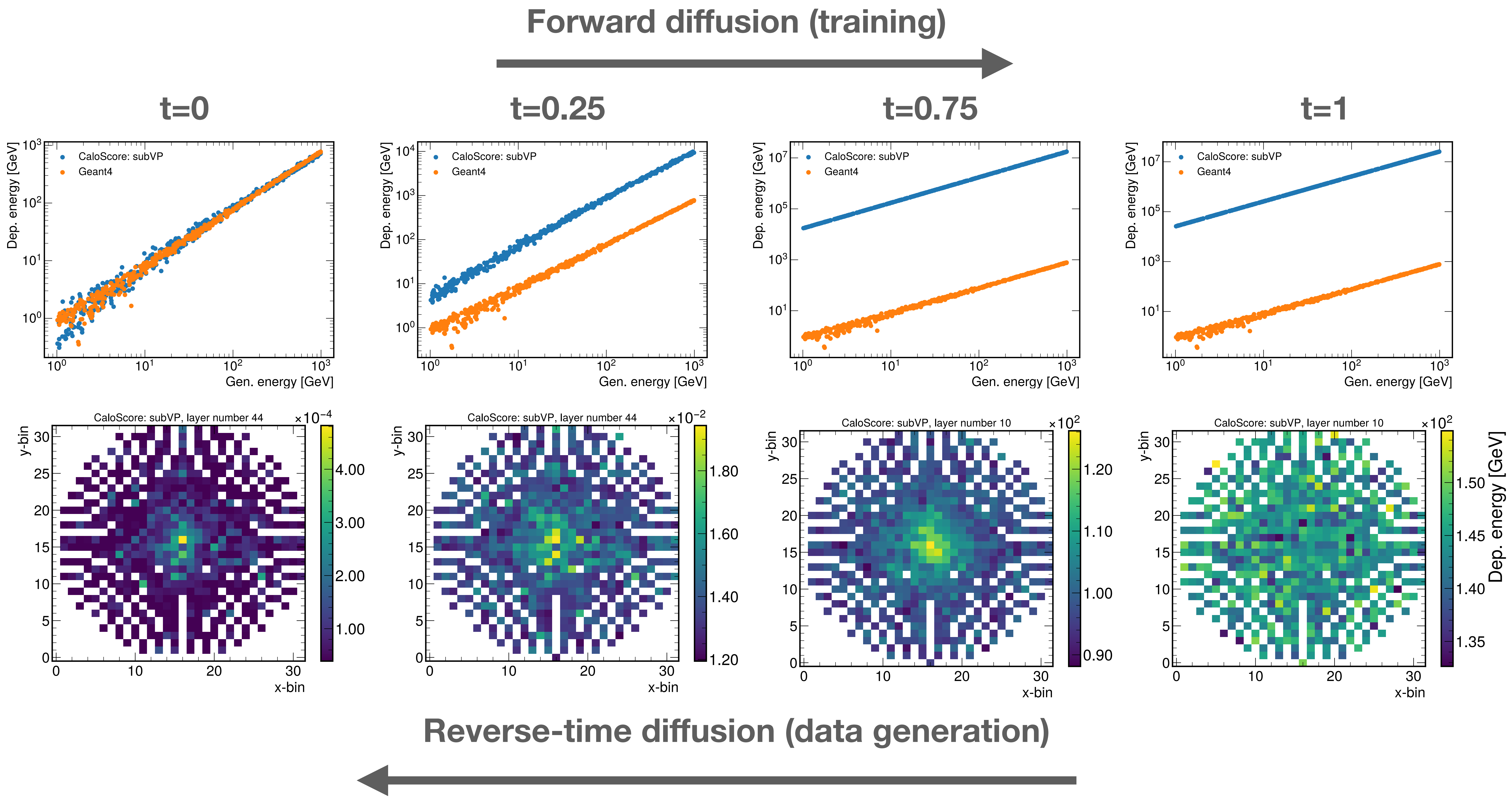
Shower average GEANT4 pion reference dataset



DIFFUSION MODELS IN HEP

arXiv:2011.13456
arXiv:2206.11898

- Diffusion models have recently dethroned GANs for natural images
- Generative model is trained using a diffusion process that slowly perturbs the data by adding noise – model learns to **denoise**
- Generation of new samples by reversing the diffusion process



Forward diffusion (training)

t=0 t=0.25 t=0.75 t=1

Reverse-time diffusion (data generation)

- Distribution of deposited energies for generated particle energies (top) and the energy deposition in a single layer of a calorimeter (bottom) vs time step

I. BASICS

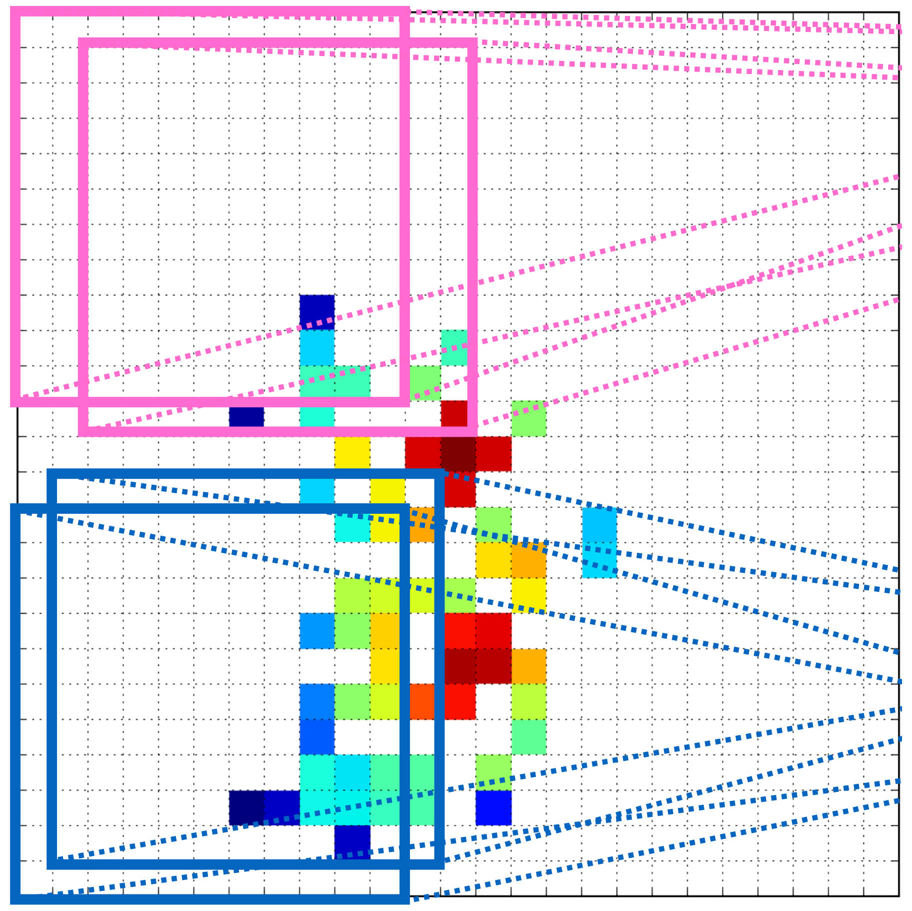
II. DATA REPRESENTATIONS & SYMMETRIES

III. ANOMALY DETECTION

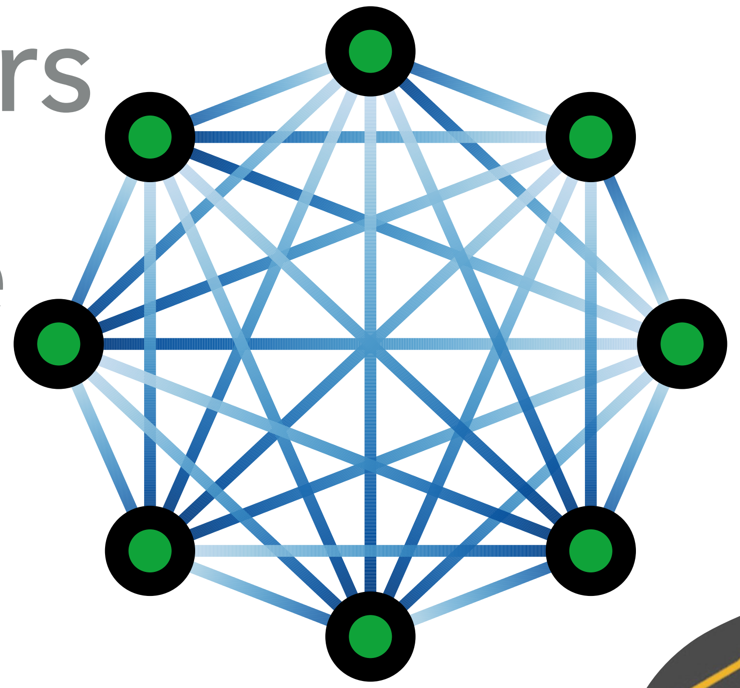
IV. GENERATIVE MODELING

V. SUMMARY & OUTLOOK

SUMMARY AND OUTLOOK



- Different representations of HEP data, from tabular data, image data, set data, graph data, paired with corresponding algorithms can achieve excellent performance
- Plethora of ML techniques in HEP from anomaly detection to generative modeling have exploded in recent years
  - Availability of public datasets and challenges have advanced the state-of-the-art



- Fast ML can accelerate science allowing us to test hypotheses faster, enhance performance of detectors/accelerators, and save potentially overlooked data
- Generative modeling can enhance or even replace current simulators