



UNIVERSITAT DE
BARCELONA



European Research Council
Established by the European Commission



ERC GA: 101077940



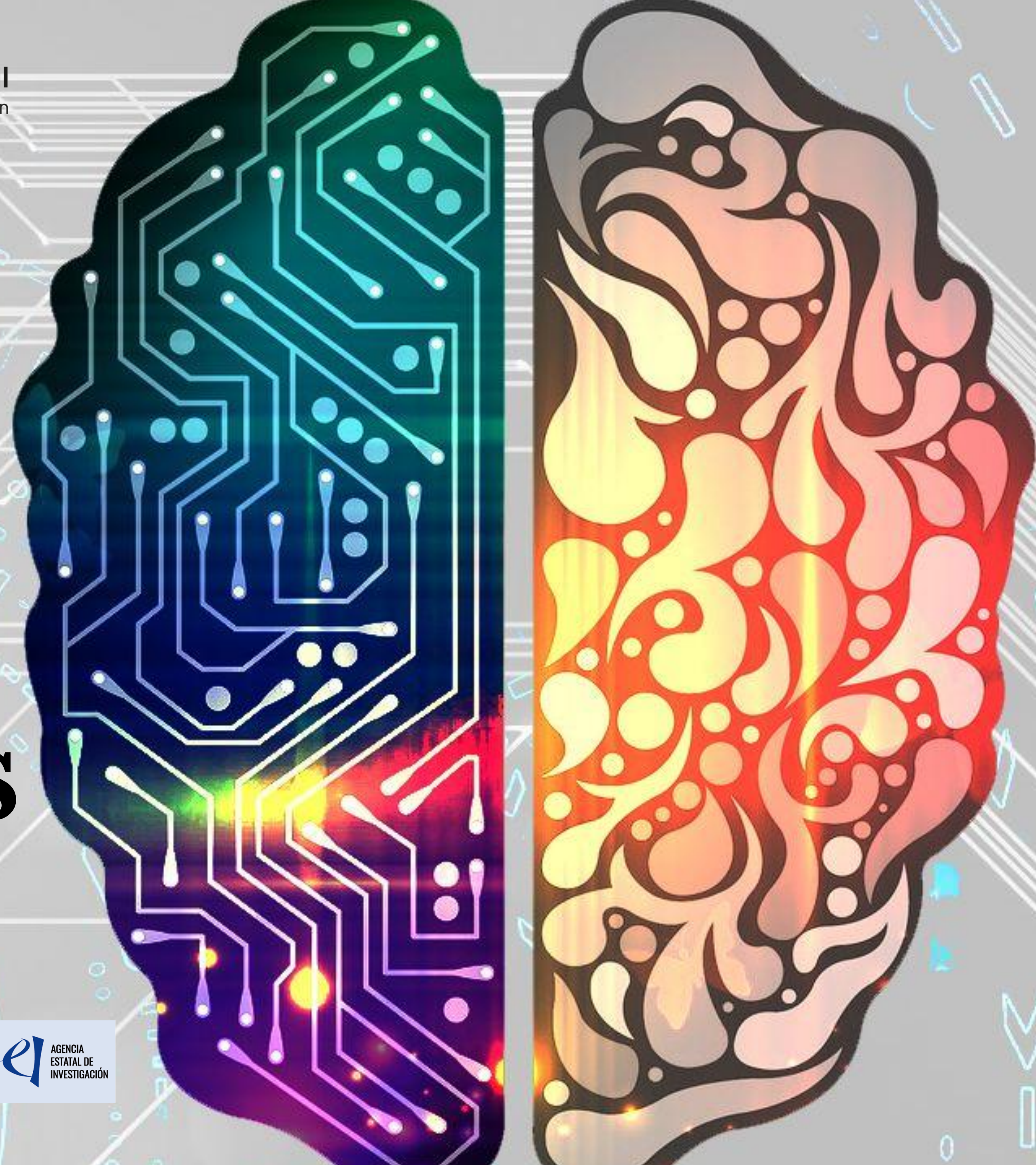
MACHINE LEARNING FUNDAMENTALS

Felipe Luan Souza de Almeida

4th COMCHA School — 08/04/2026



Institut de Ciències del Cosmos
UNIVERSITAT DE BARCELONA



Outline (lecture 1)

Introduction to machine learning

- ▶ What / why / how?

The ML pipeline

- ▶ Data → model → loss → training → evaluation

Key concepts

- ▶ Overfitting

- ▶ Generalization

Types of problems

- ▶ Supervised vs unsupervised

- ▶ Classification, regression

References

Lectures:

- ▶ *M. Kagan*: [CERN Summer Student Lecture](#) (2024) and [TRISEP school](#) (2024)
- ▶ *F. Fleuret*: [Deep Learning Course](#) (2018-now, EPFL and University of Geneva)
- ▶ *T. Sculac*: [CERN Thematic School of Computing on Machine Learning](#) (2024)
- ▶ *F. Vaselli*: [CERN Thematic School of Computing on Machine Learning](#) (2024)
- ▶ *D. Noll*: [MLPF Berkley school](#) (2024)
- ▶ *L. Moneta*: [Machine Learning for Fundamental Physics school](#) (2025)

Books:

- ▶ Deep Learning (*I. Goodfellow, Y. Bengio, A. Courville*)
- ▶ The Elements of Statistical Learning (*T. Hastie, R. Tibshirani, J. Friedman*)
- ▶ Pattern Recognition and Machine learning (*C. Bishop*)

Machine learning: a bit of history



Alan Turing: the father of
“Artificial Intelligence”

📌 “Computing Machinery and
Intelligence” (Mind, 1950)

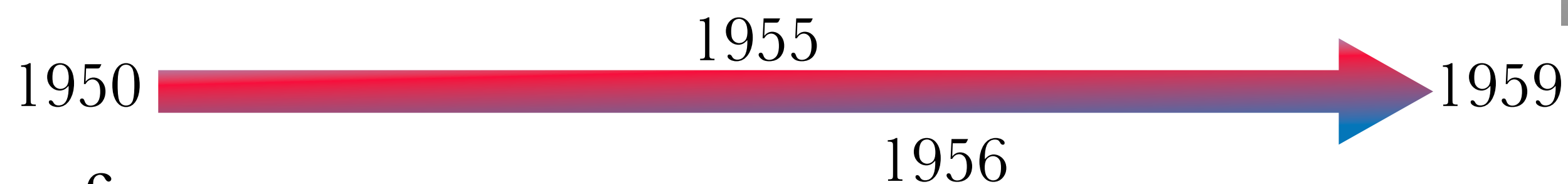
▶ Turing test

▶ “Learning machines”

“Session on Learning Machines” (1955)

Western Joint Computer Conference,
United States

“It is perhaps a misnomer to label such
devices as ‘learning machines,’ since **it is
not the machine itself, but the program
that guides the machine toward its goal,
which learns.**”



John McCarthy et al.
“Dartmouth Summer Research Project
on **Artificial Intelligence**” (1956)

Dartmouth College, United States



1959 **Arthur Samuel**

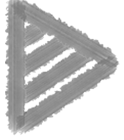
“Some Studies in **Machine Learning**
Using the Game of Checkers”

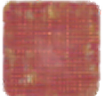
(*IBM Journal of Research and
Development*, 1959)

What is Machine Learning?

What is Machine Learning?

 Arthur Samuel (1959):

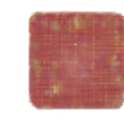
 “Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.”

 *Some Studies in Machine Learning Using the Game of Checkers*

What is Machine Learning?

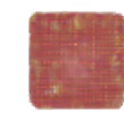
 Arthur Samuel (1959):

▶ “Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed.”

 *Some Studies in Machine Learning Using the Game of Checkers*

 Tom M. Mitchel (1997):

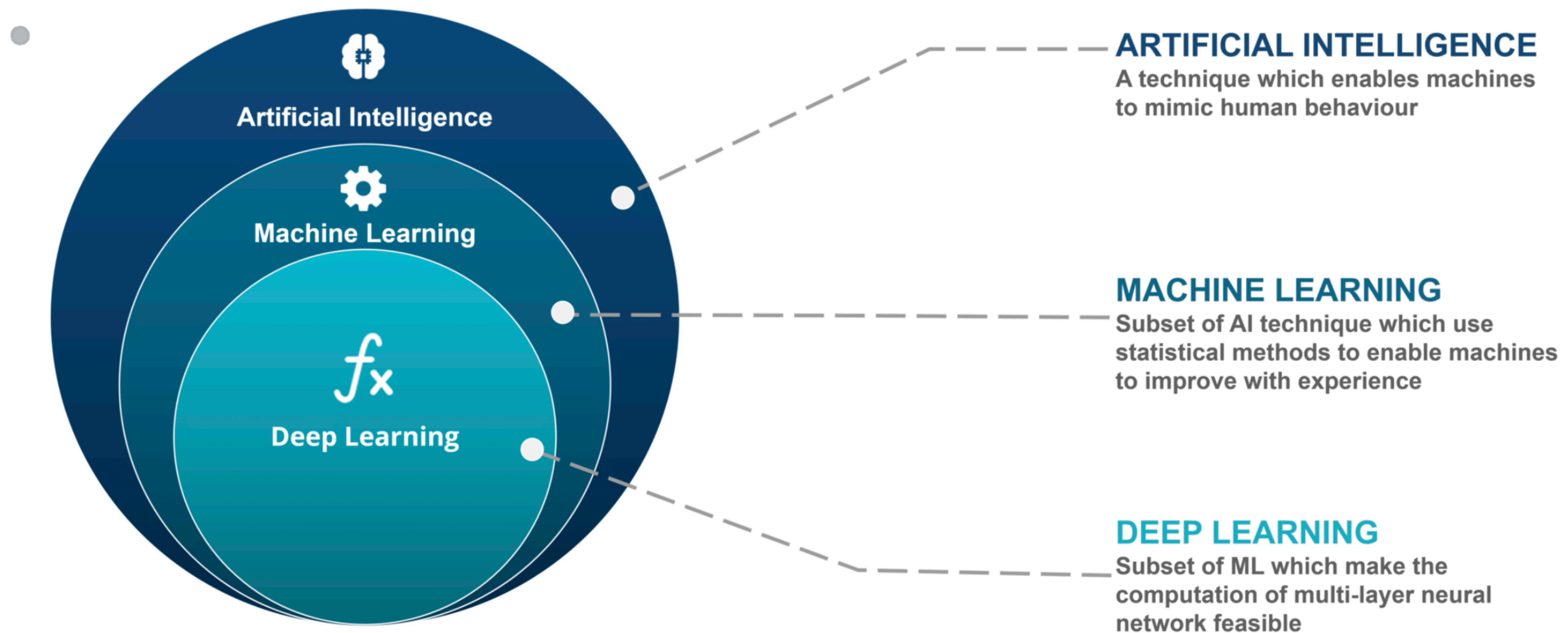
▶ “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .”

 *Machine Learning*

Machine learning and AI

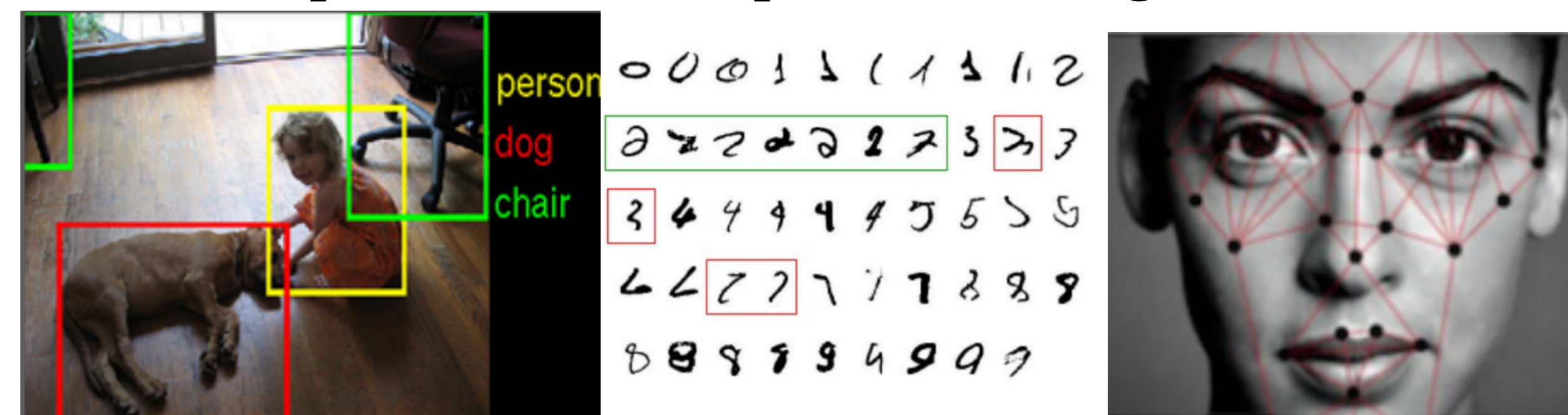
📌 Wang (*Principles of Machine Learning: The Three Perspectives*, 2024):

▶ “Machine learning is a branch of artificial intelligence that studies and builds models that can learn from data or the environment to make predictions about unknown data or make decisions based on environmental information”



Machine learning usage

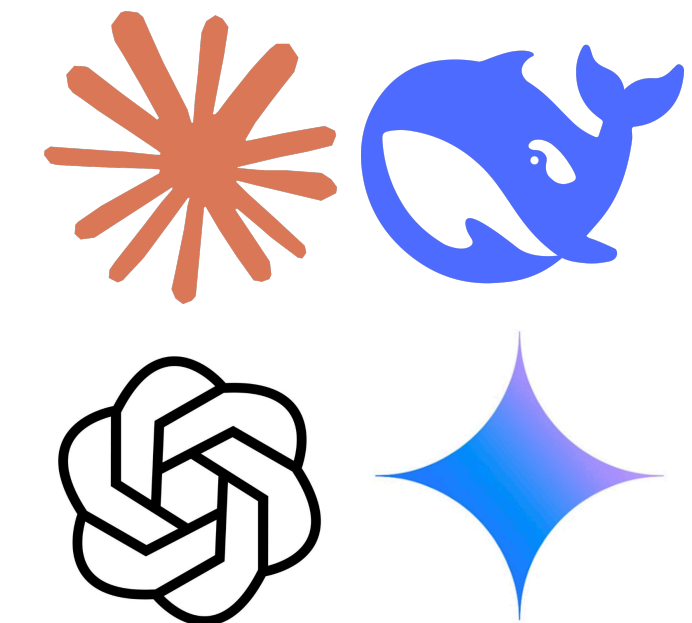
Computer vision & pattern recognition



Self-driving cars



LLM models



Recommendation engines



Medical diagnostics



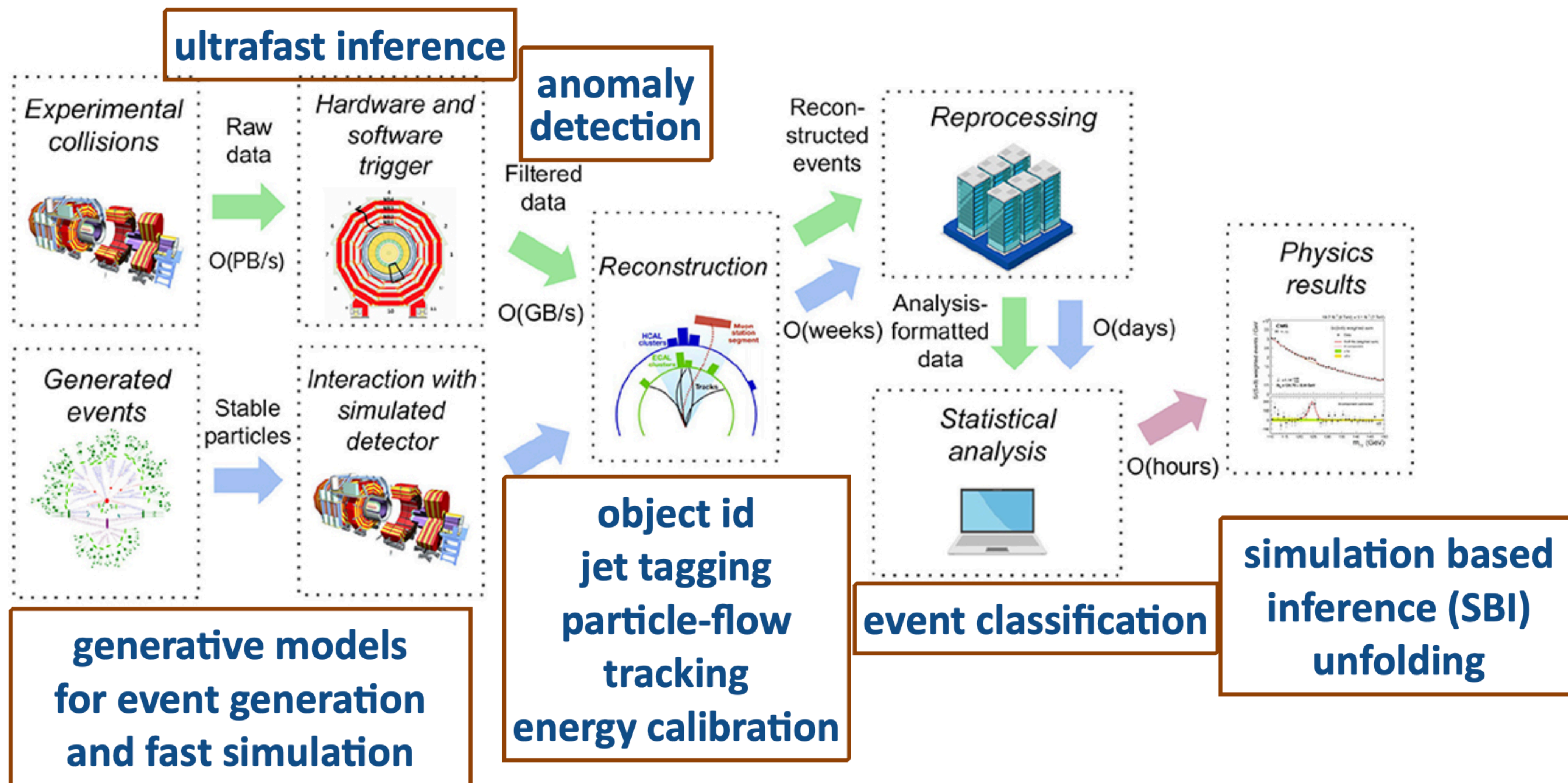
Dynamic pricing



Fraud detection



ML in HEP



But how?

📌 Key element in machine learning: the **mathematical model**

▶ A mathematical function $f(x) \rightarrow y$

▶ Choice depends on task and data

📌 Learning = optimization!

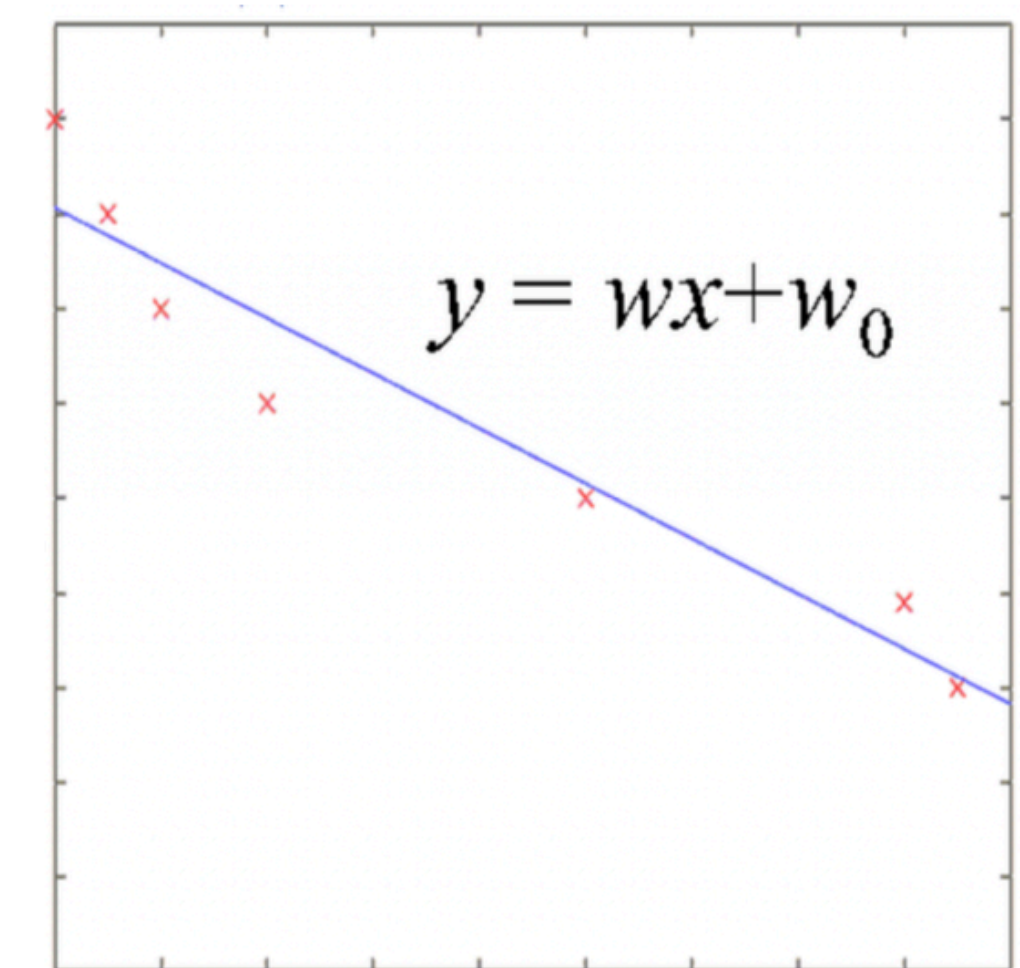
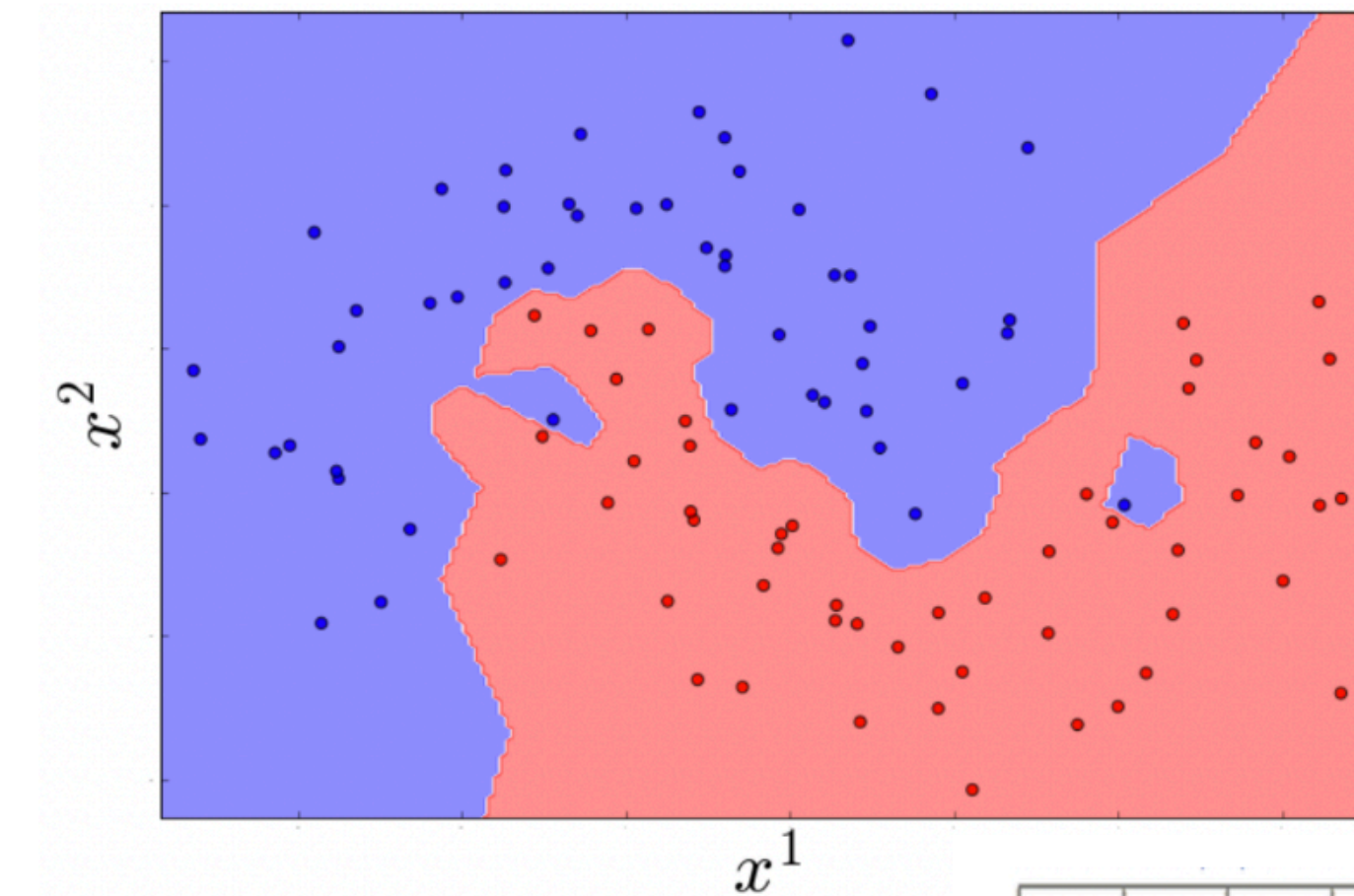
▶ Find parameters that best describe the data

▶ Minimize a measure of error (loss)

📌 Prediction and inference:

▶ Apply the model to new data

▶ Extract information about the system



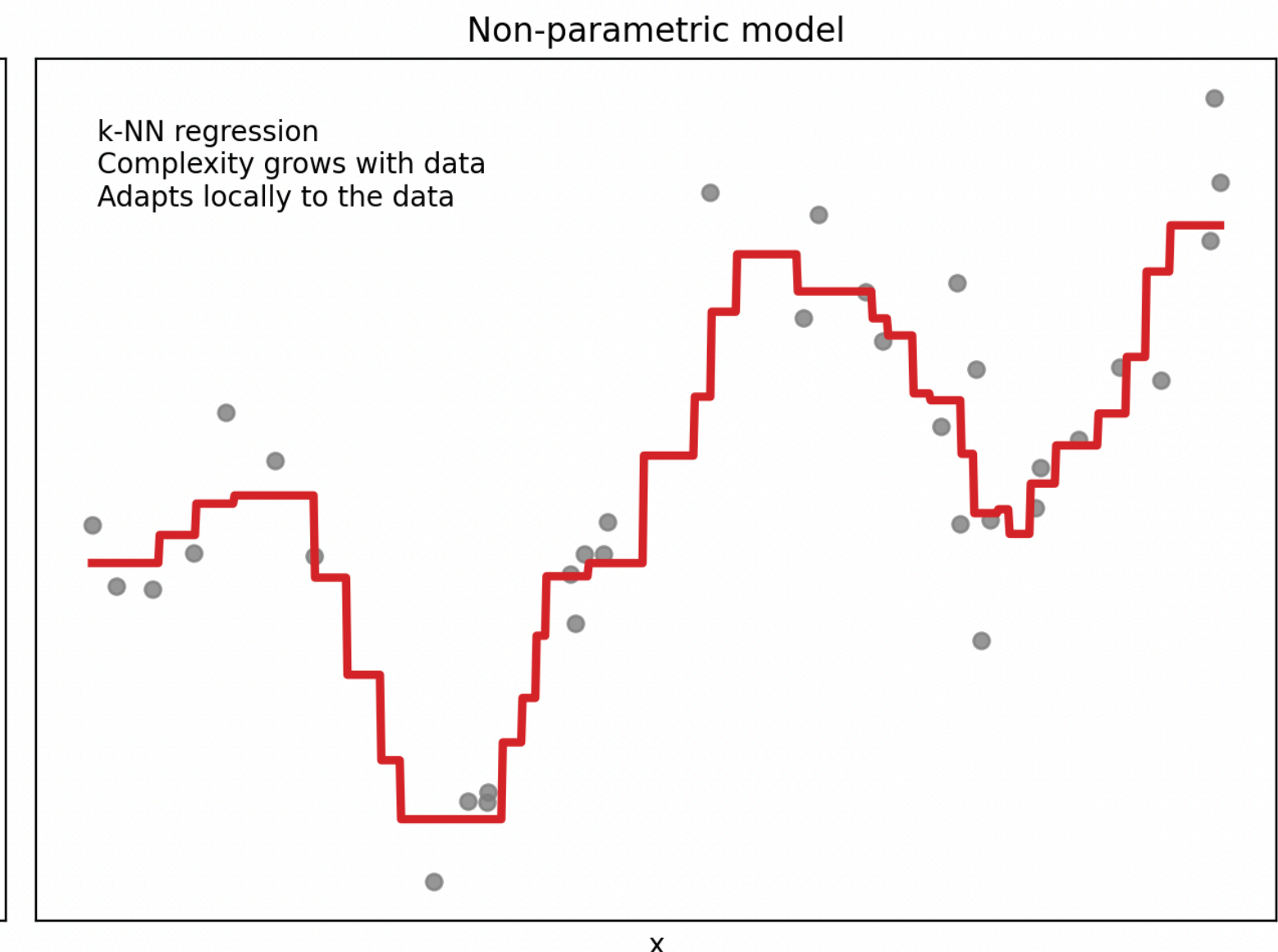
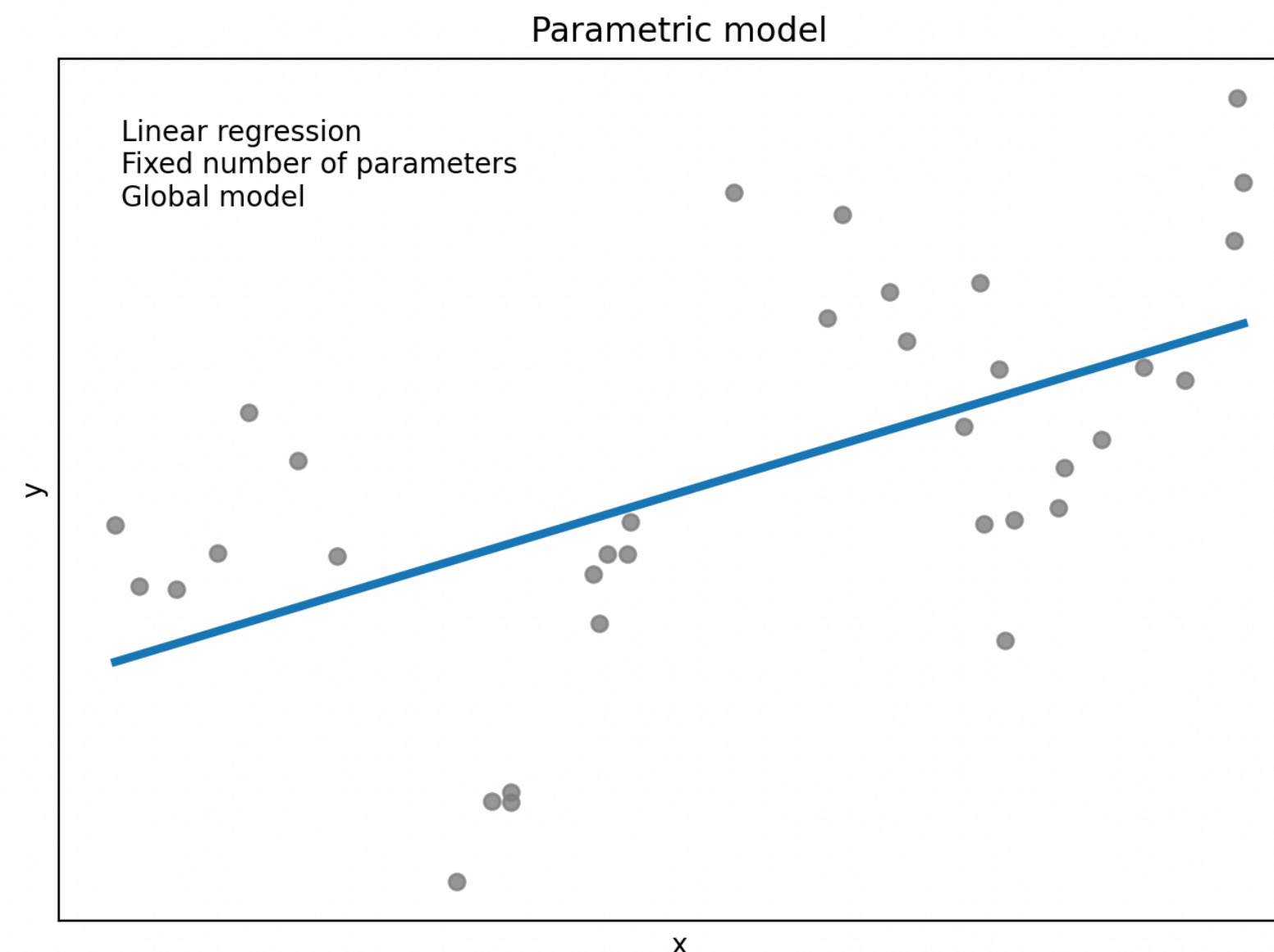
Model: parametric vs non-parametric

Parametric models:

- ▶ Assumes known distribution
- ▶ Fixed number of parameters
- ▶ Simple, fast
- ▶ E.g.: linear regression, logistic regression, ...

Non-parametric models:

- ▶ No direct assumptions
- ▶ Complexity grows with data
- ▶ More flexible
- ▶ E.g.: k-NN, decision trees, ...



Model: generative vs discriminative

📌 Generative models:

▶ Model $P(x, y)$

■ Learns how the data is generated

📌 Discriminative models:

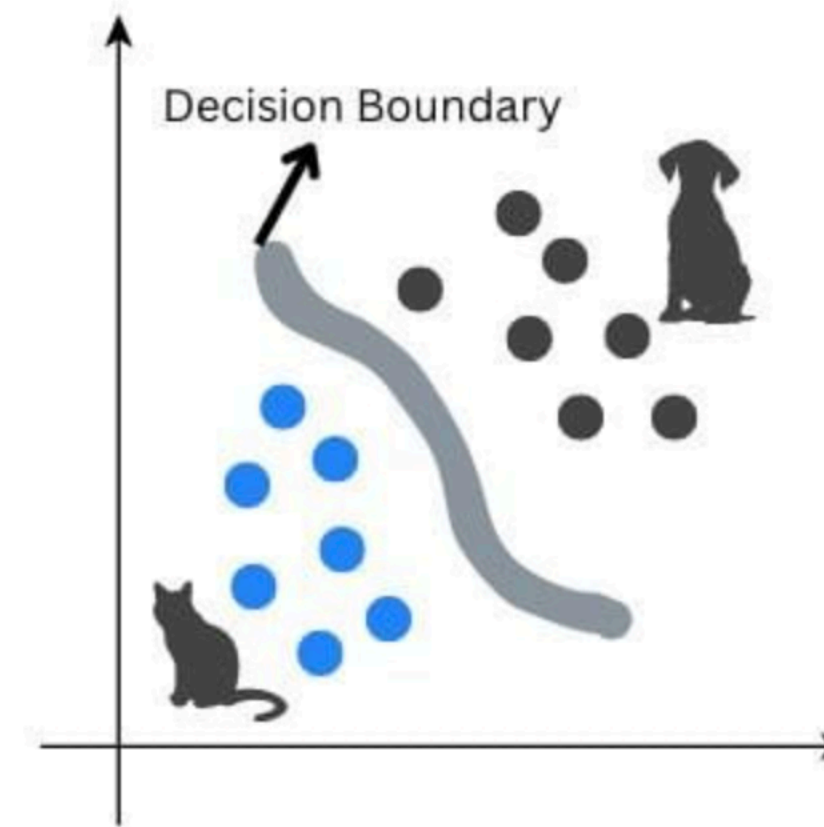
▶ Model $P(y | x)$

■ Learns decision boundary

▶ Focus on predictions

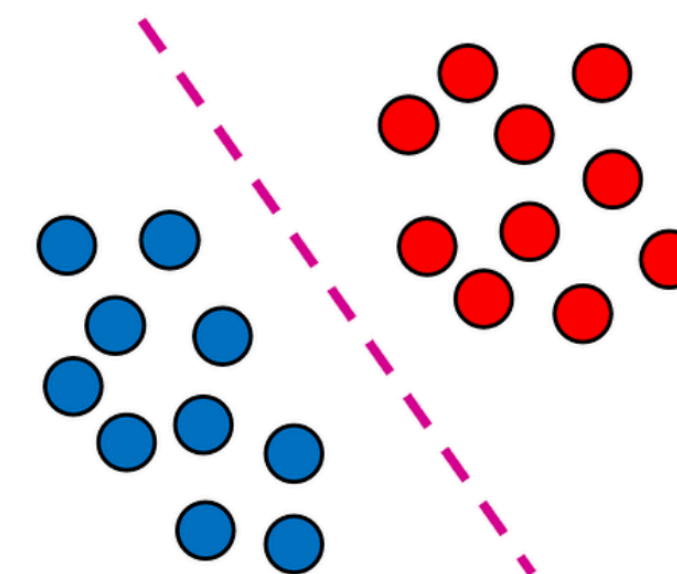
▶ E.g. logistic regression, neural networks, ...

Discriminative

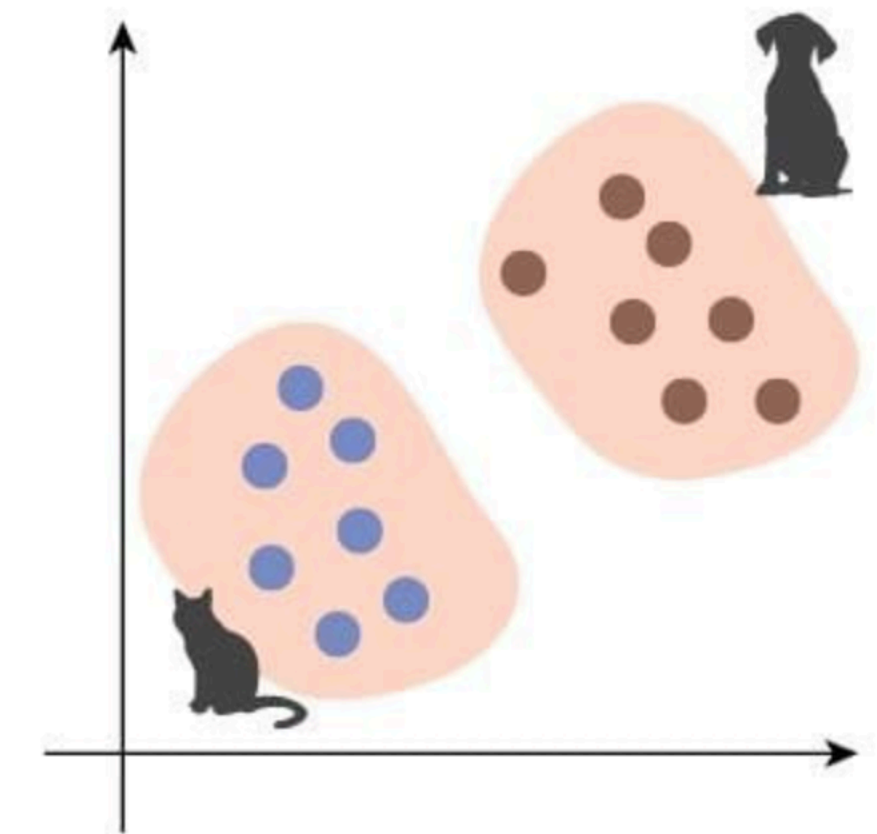


Classify or Label data point
as cat or dog

Discriminative

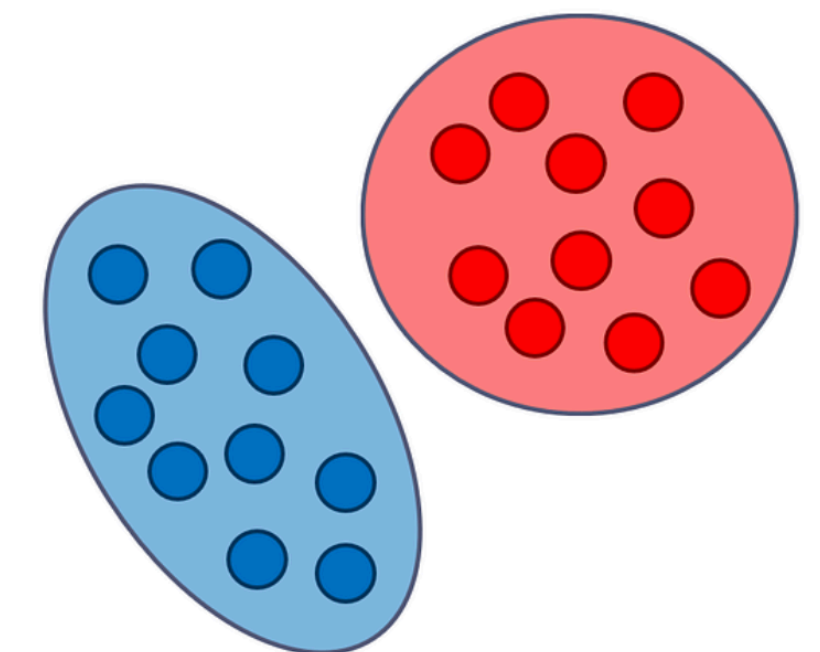


Generative

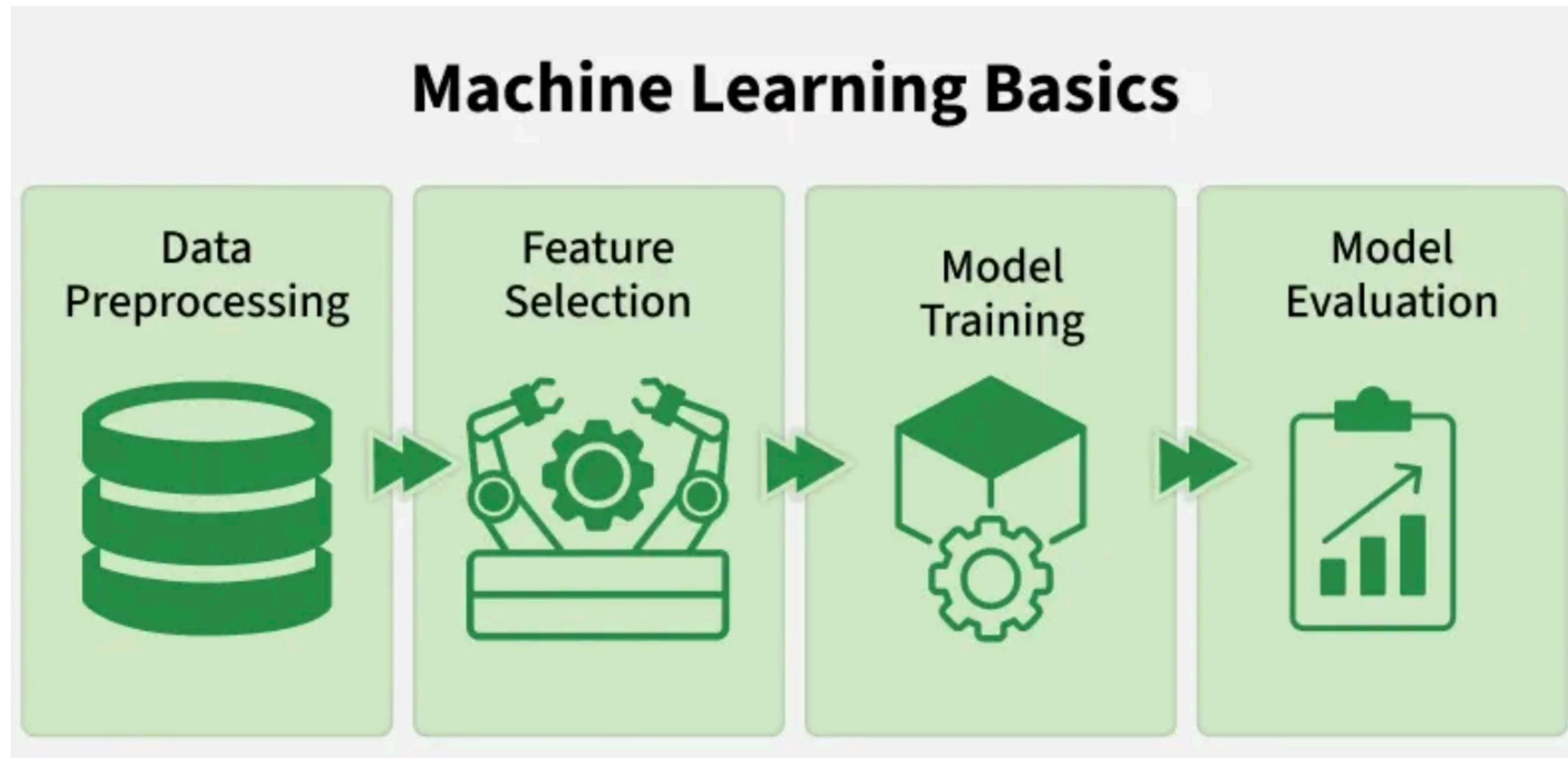


Produce a new data
point that looks like
cats or dogs

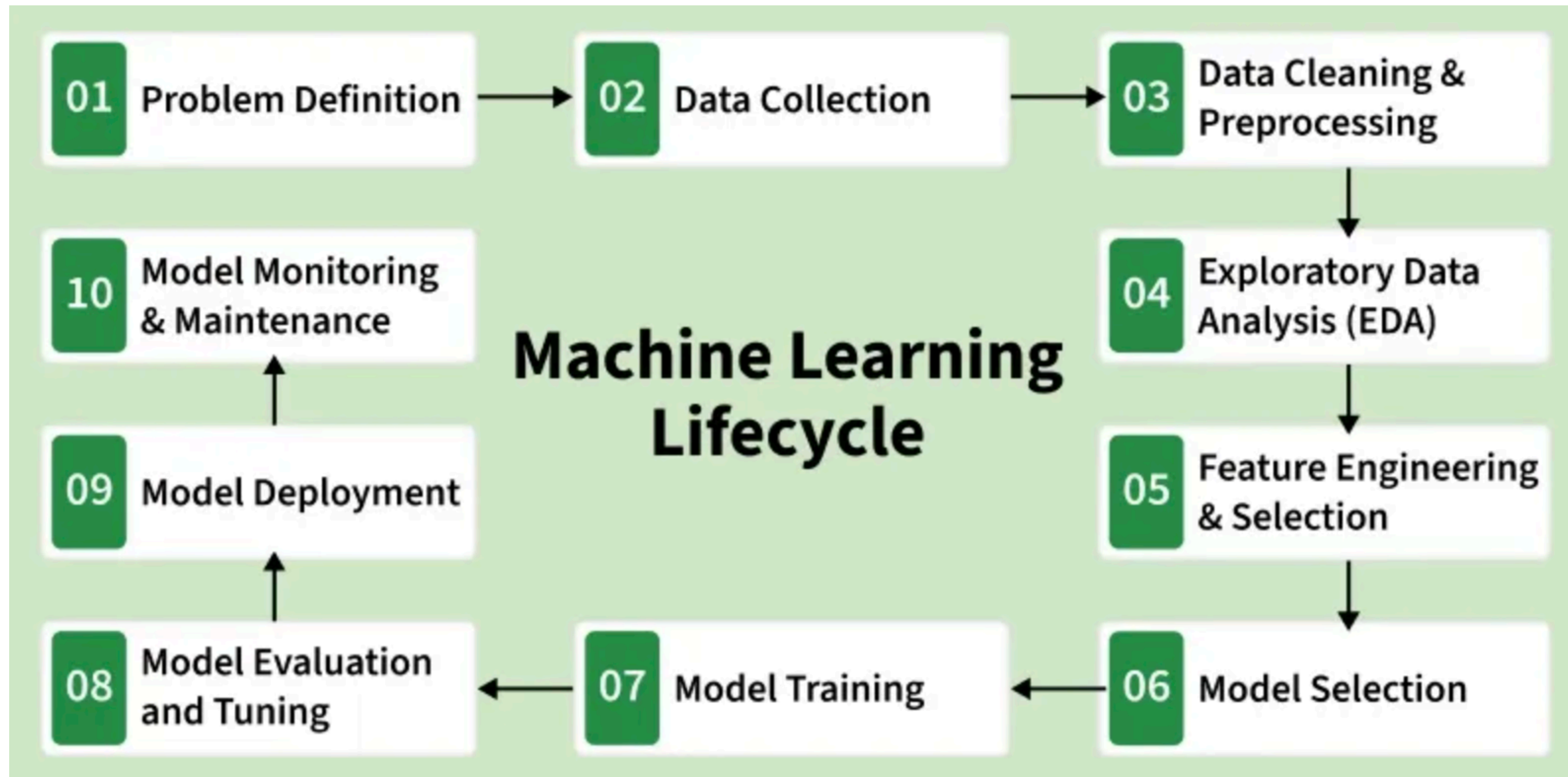
Generative



The ML pipeline



The ML pipeline



Data cleaning and preprocessing

📌 Real-world data is messy

▶ Outliers

▶ Missing data

▶ Inconsistent formats

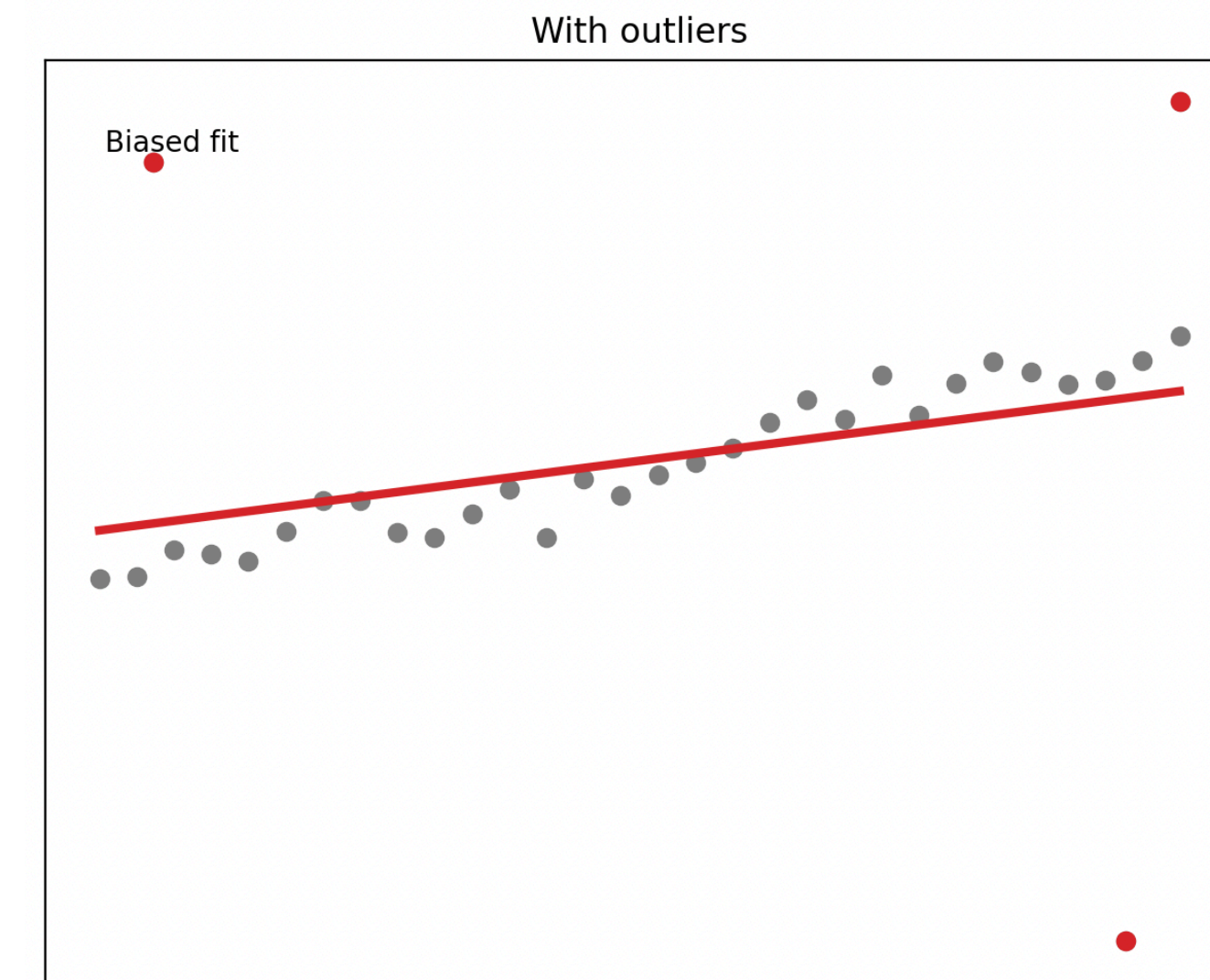
📌 Cleaning improves model stability and performance

📌 Typical steps:

▶ Remove / treat outliers

▶ Handle missing data

▶ Normalize / scale features



Data cleaning and preprocessing

📌 Real-world data is messy

▶ Outliers

▶ Missing data

▶ Inconsistent formats

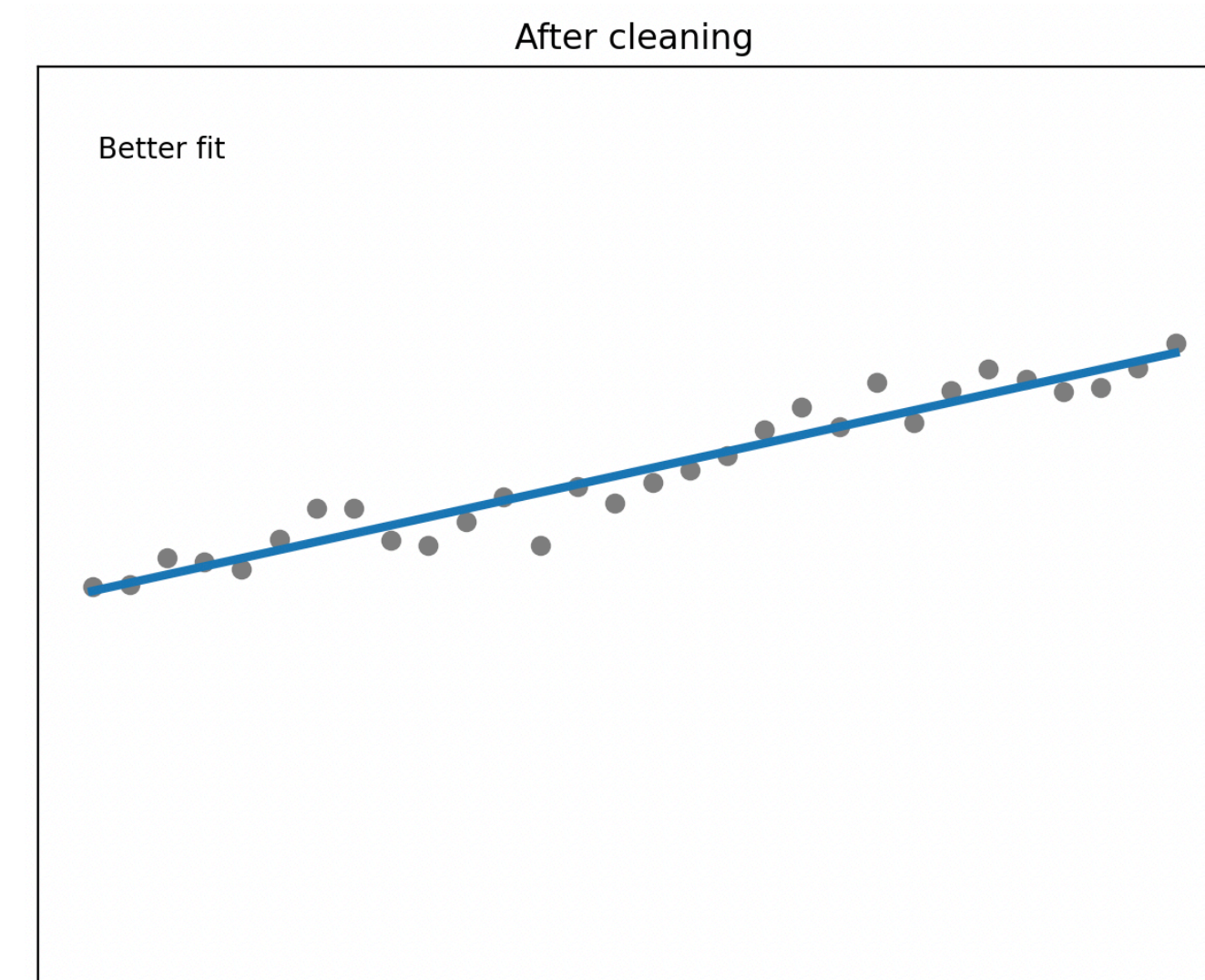
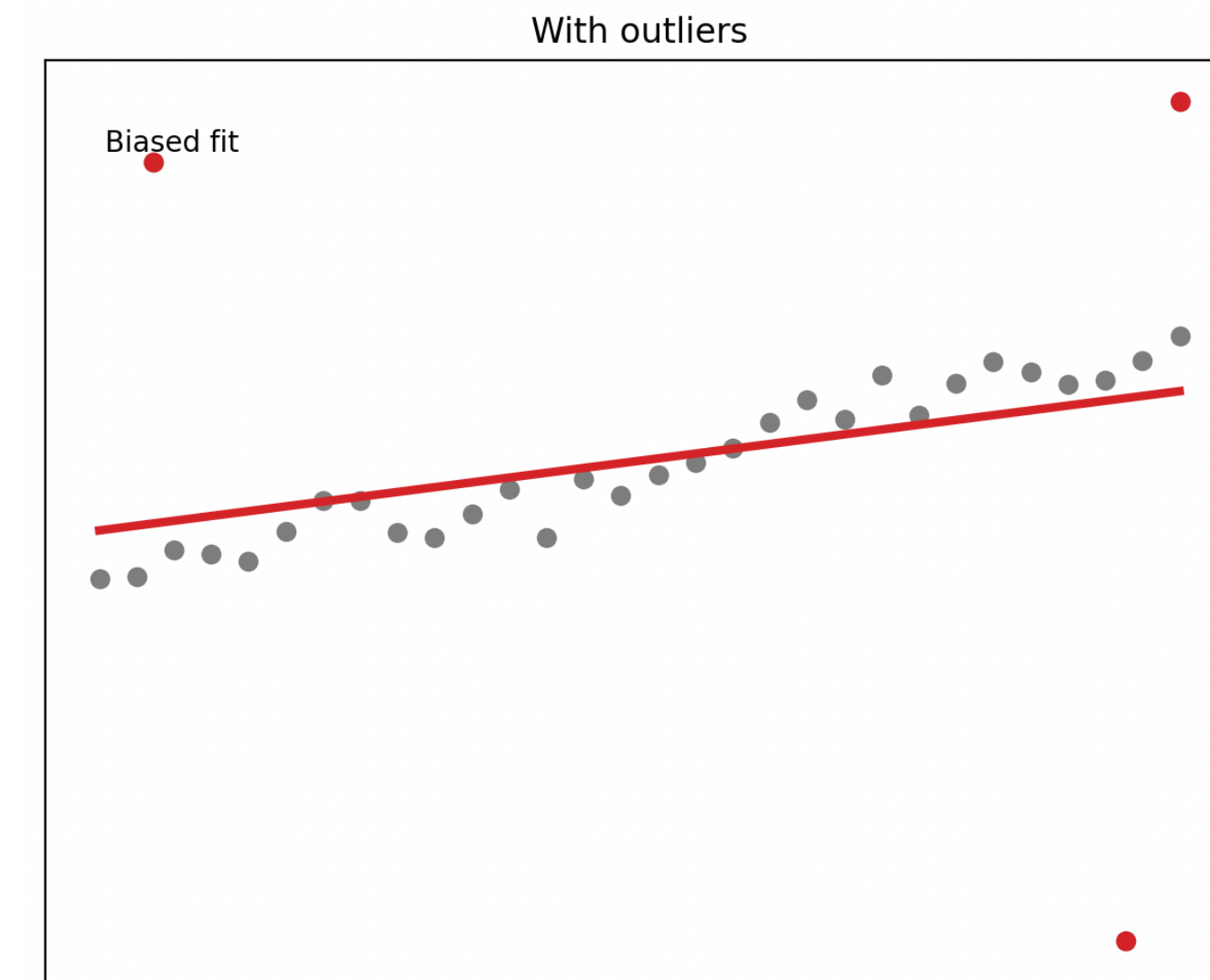
📌 Cleaning improves model stability and performance

📌 Typical steps:

▶ Remove / treat outliers

▶ Handle missing data

▶ Normalize / scale features



Data preprocessing: missing values

📌 Recorded data often has missing values (e.g. limited amount of jets in HEP events)

📌 Strategies:

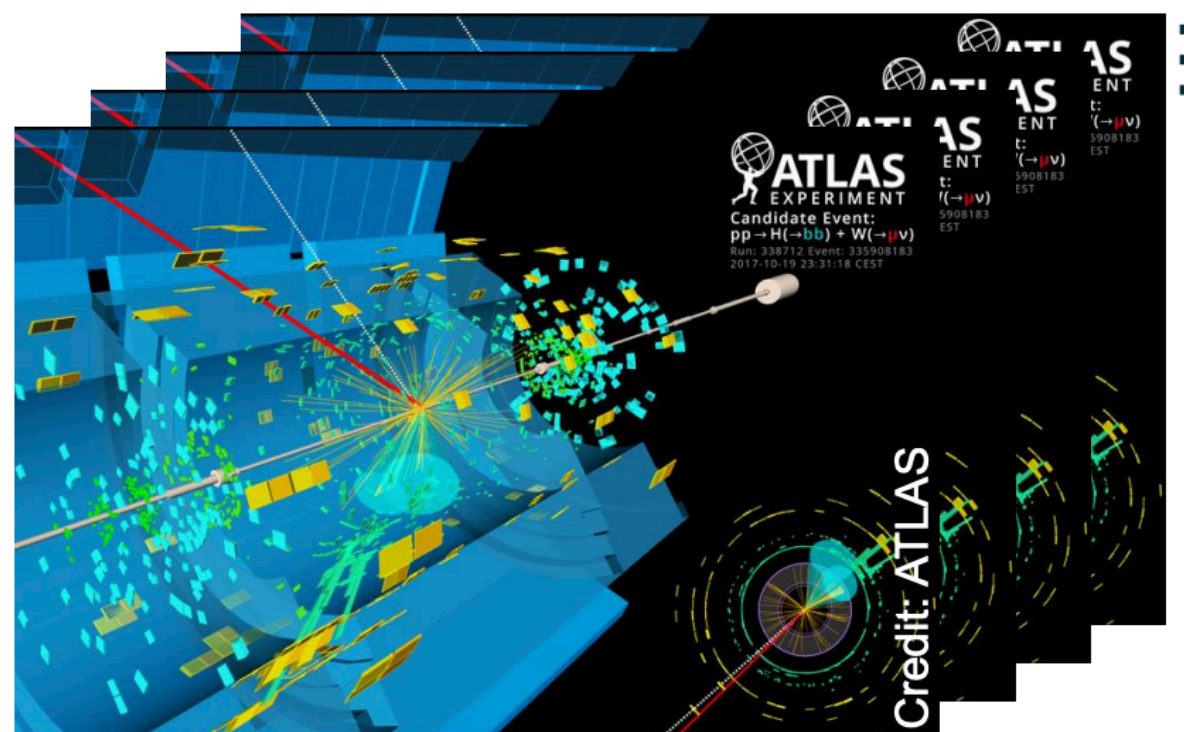
▶ **Delete rows / columns**: easy, but can lose a lot of important information

▶ **Fill values with mean / mode / interpolation**: not always applicable

▶ **Fill values with categorical value (e.g. 0, -1)**: involves educated guess

Use what works!

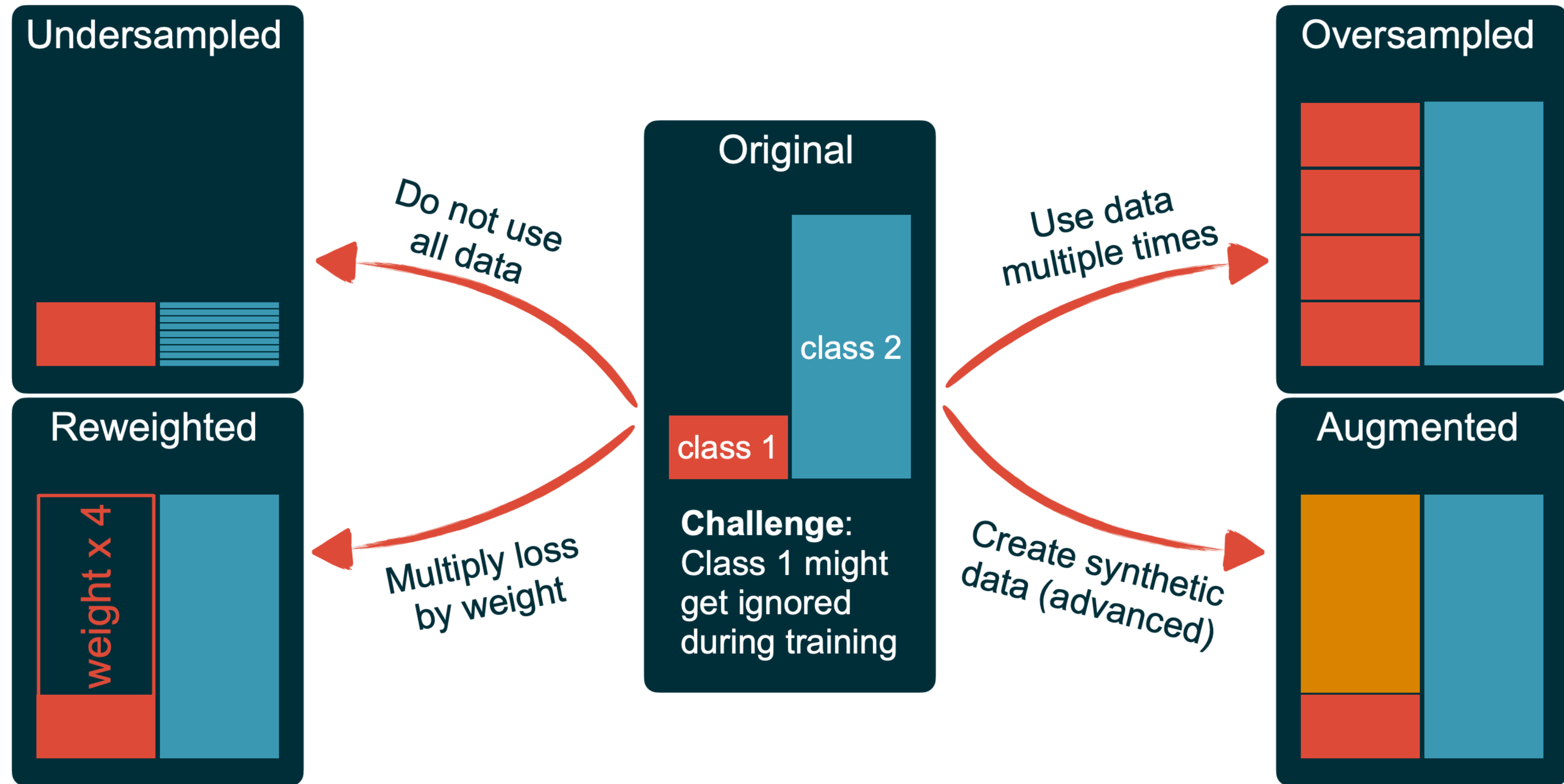
Detected events



Training data

	Jet 1 p_T	Jet 1 η	Jet 2 p_T	Jet 2 η
Event #1	100	1.5	80	0.2
Event #2	250	2.1	180	1.1
Event #3	180	0.3	-	-
Event #4	200	0.8	-	-
Event #5	170	1.2	100	0.1
Event #6	210	0.5	-	-

Data preprocessing: class imbalance



Feature engineering

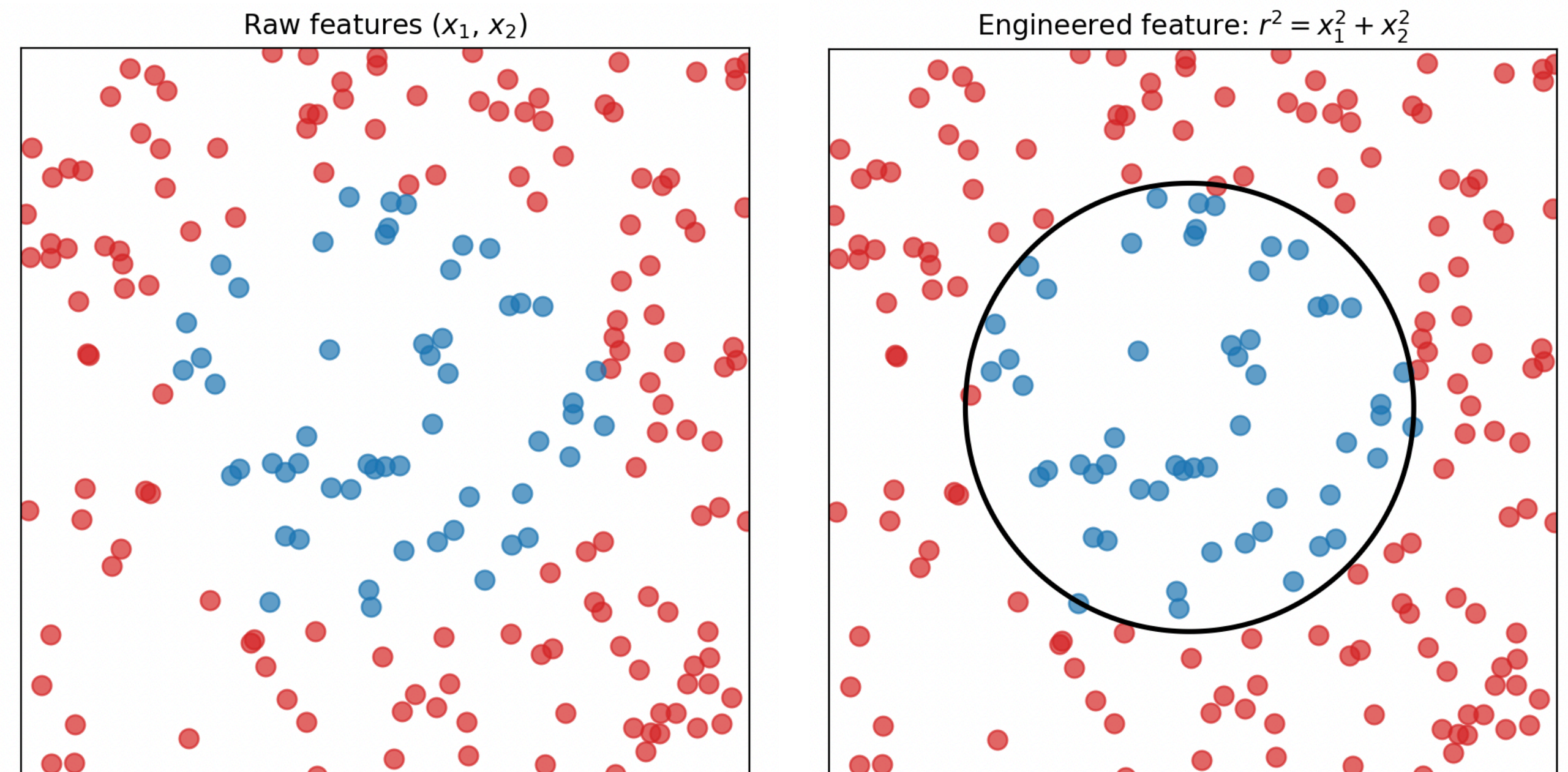
- 📌 Transform raw data into meaningful inputs

- 📌 Good features simplify the problem

- ▶ Reveal patterns
- ▶ Improve separability

- 📌 Examples:

- ▶ Scaling / normalization
- ▶ log / non-linear transforms
- ▶ Combining variables ($x_1 \cdot x_2$, ratios, ...)



- 📌 Same model, different representation

- 📌 Non-linear problem → linear problem

Feature engineering

- 📌 Transform raw data into meaningful inputs

- 📌 Good features simplify the problem

 - ▶ Reveal patterns

 - ▶ Improve separation

- 📌 Examples:

 - ▶ Scaling / normalization

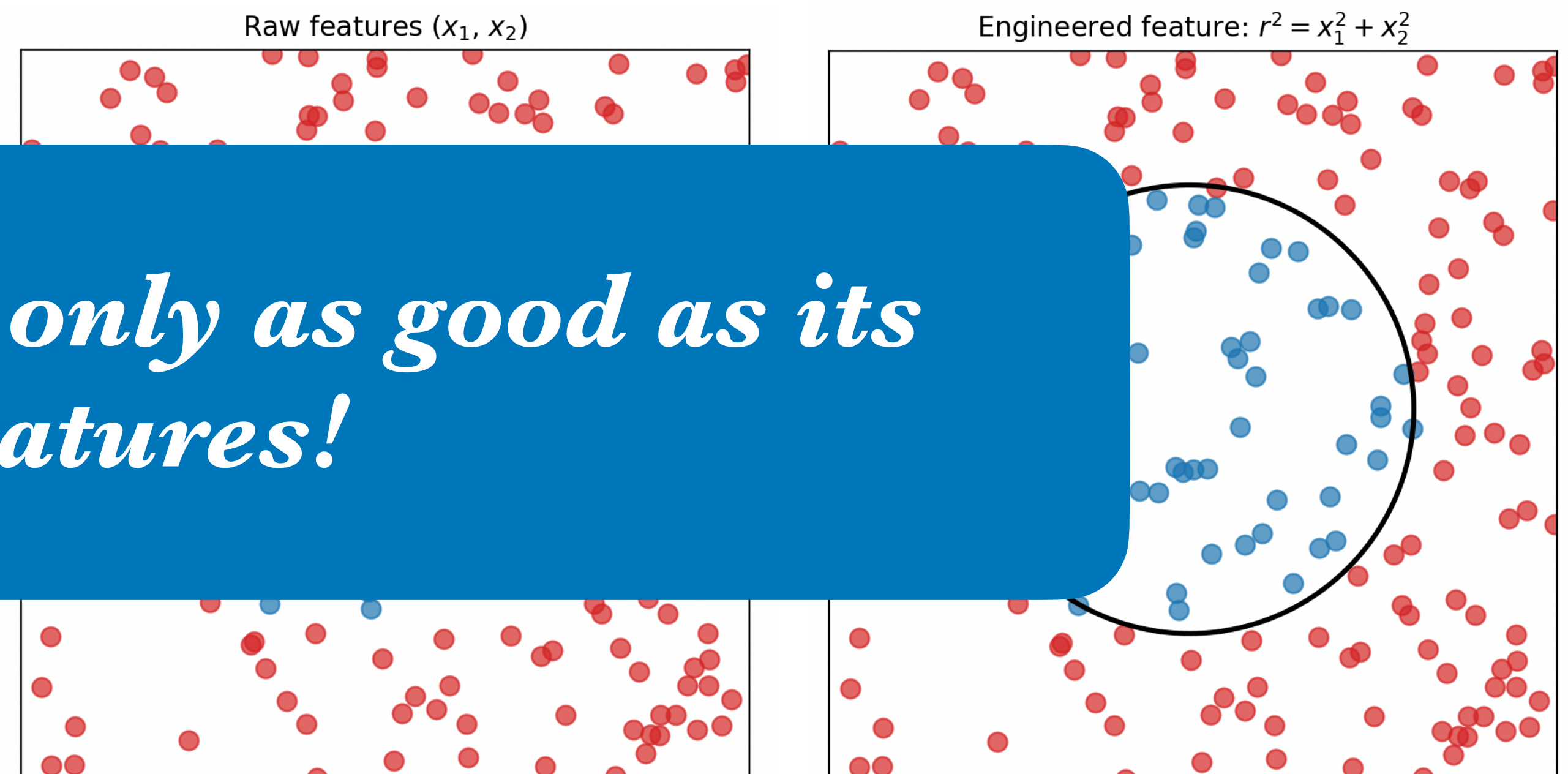
 - ▶ log / non-linear transforms

 - ▶ Combining variables ($x_1 \cdot x_2$, ratios, ...)

 - 🌐 Same model, different representation

 - 🌐 Non-linear problem → linear problem

The model is only as good as its features!



The loss function

- 📌 The loss measures how wrong predictions are
- 📌 Training = find parameters that minimizes the loss

$$\min_{\theta} \sum_i \mathcal{L}(y_i, f(x_i; \theta))$$

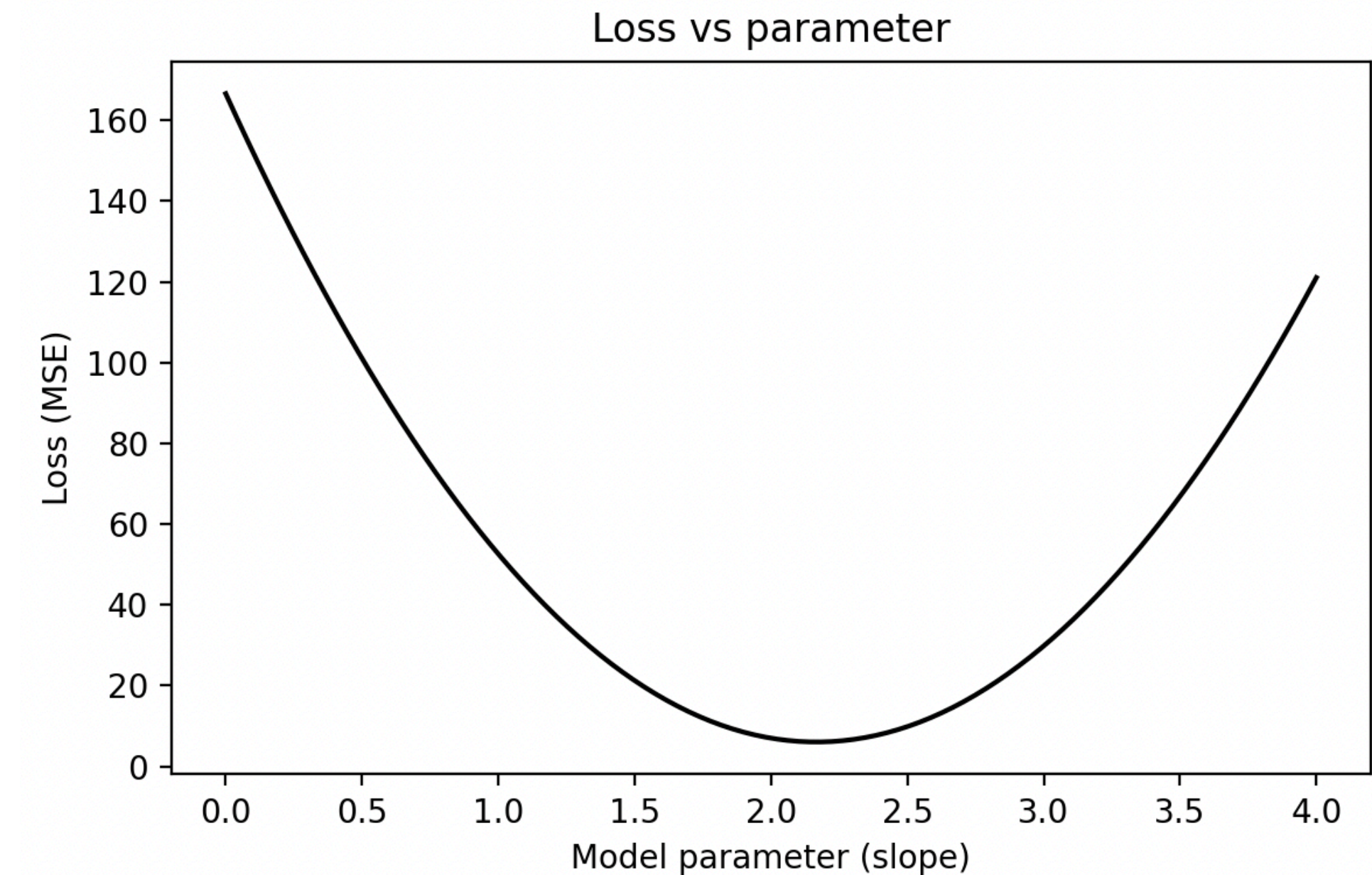
- 📌 Different problems → different loss functions

► Regression → Mean Squared Error (MSE)

► Classification → Cross-entropy

$$\mathcal{L}(y, \hat{y}) = \frac{1}{N} \sum_i (y_i - \hat{y}_i)^2$$

$$\mathcal{L}(y, \hat{p}) = - [y \log(\hat{p}) + (1 - y) \log(1 - \hat{p})]$$



Example: MSE vs MAE

📌 MSE: $\mathcal{L}(y, \hat{y}) = \frac{1}{N} \sum_i (y_i - \hat{y}_i)^2$

- ▶ Squares the error → sensitive to outliers
- ▶ Penalizes large errors strongly

📌 MAE: $\mathcal{L}(y, \hat{y}) = \frac{1}{N} \sum_i |y_i - \hat{y}_i|$

- ▶ Linear penalty → robust to outliers



★ Outliers dominate MSE but not MAE

Training and optimization

📌 Training = minimizing the loss

$$\min_{\theta} \sum_i \mathcal{L}(y_i, f(x_i; \theta))$$

📌 But how do we minimize it?

Training and optimization

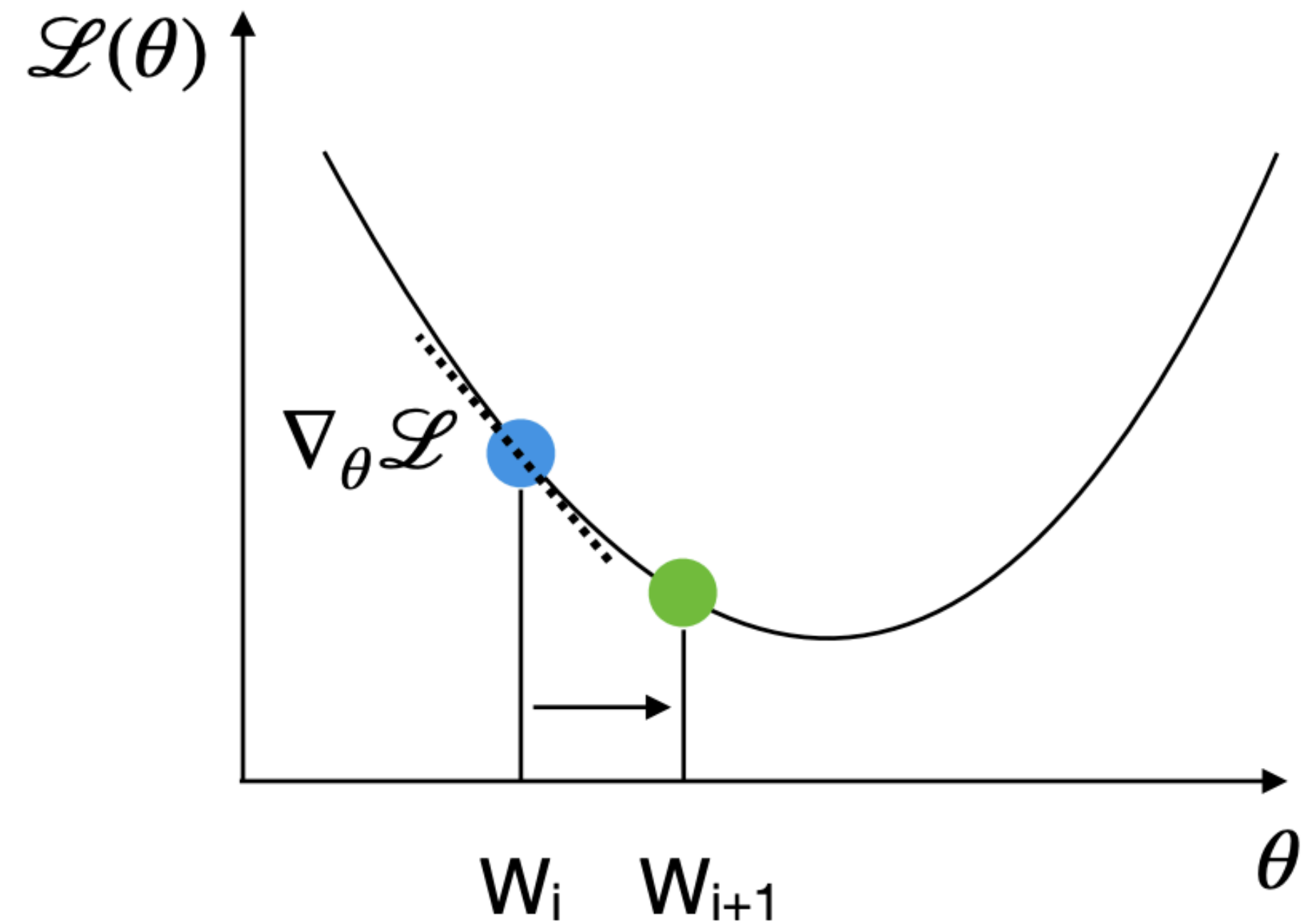
📌 Training = minimizing the loss

$$\min_{\theta} \sum_i \mathcal{L}(y_i, f(x_i; \theta))$$

📌 But how do we minimize it?

▶ Follow the negative **gradient**!

- ▣ Gradient = direction of steepest increase
- ▣ We move in the opposite direction
- ▣ Step size controlled by the **learning rate** (η)



★ Gradient descent

$$\theta^{(t+1)} = \theta^{(t)} - \eta \sum_i \nabla_{\theta} \mathcal{L}(y_i, f(x_i; \theta^{(t)}))$$

Learning rate (η)

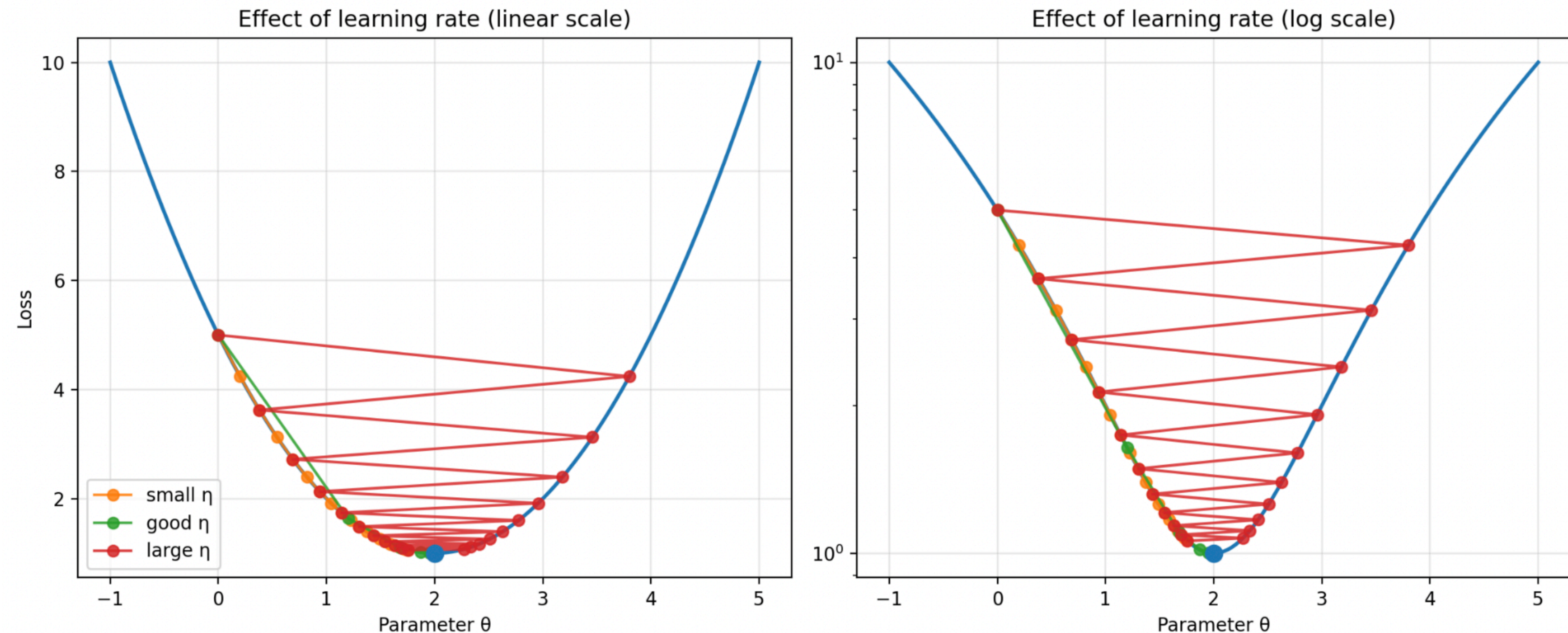


Controls the step size

▶ Too small $\rightarrow$ slow convergence

▶ Too large $\rightarrow$ divergence, oscillations, poor convergence

▶ Just right $\rightarrow$ fast convergence



Even when training does not diverge, a large learning rate can prevent efficient convergence

Not all bad learning rates diverge: some just learn very poorly!

Learning rate scheduler

📌 Fixed learning rate is rarely optimal: use learning rate schedules to improve convergence

▶ Adapt step size during training

📌 Common strategies:

▶ Step decay: reduce η in steps

▶ Reduce on plateau: monitor validation metric, reduce η when improvement stall

▶ Cosine annealing: smooth, continuous decay of η

▶ Cosine annealing + reduce on plateau: global smooth schedule with adaptive correction



Learning rate scheduler

📌 Fixed learning rate is rarely optimal: use learning rate schedules to improve convergence

➤ Adapt step size during training

📌 Common strategies:

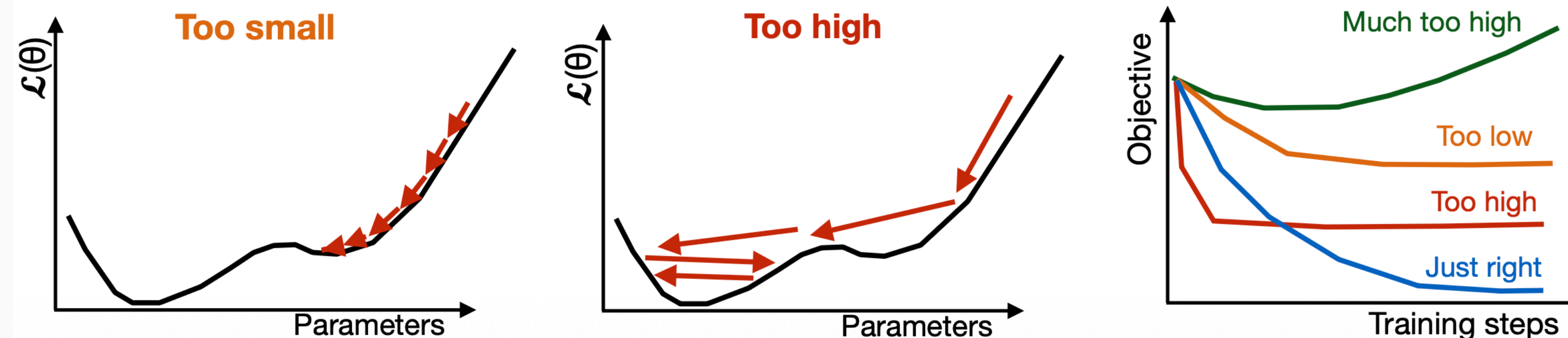
➤ Step decay: red

➤ Reduce on plat

➤ Cosine anneal

➤ Cosine annealing + reduce on plateau: global smooth schedule with adaptive correction

The learning rate is not just a hyperparameter; it is a core part of the training strategy!



Batch vs Stochastic Gradient Descent

📌 Gradient uses all data

📌 How much data should we use per step?

▶ Batch gradient descent

■ Uses all data

$$\theta \leftarrow \theta - \eta \sum_{i=1}^N \nabla_{\theta} \mathcal{L}(y_i, f(x_i; \theta))$$

■ Stable but slow

▶ Stochastic gradient descent (SGD)

■ Uses one sample

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathcal{L}(y_j, f(x_j; \theta))$$

■ Fast but noisy

▶ Mini-batch gradient descent

■ Uses small batches

$$\theta \leftarrow \theta - \eta \frac{1}{|B|} \sum_{i \in B} \nabla_{\theta} \mathcal{L}(y_i, f(x_i; \theta))$$

■ Best trade-off

Batch vs Stochastic Gradient Descent

📌 Gradient uses all data

📌 How much data should we use per step?

▶ Batch gradient descent

■ Uses all data

■ Stable but slow

$$\theta \leftarrow \theta - \eta \sum_{i=1}^N \nabla_{\theta} \mathcal{L}(y_i, f(x_i; \theta))$$

▶ Stochastic gradient descent (SGD)

■ Uses one sample

■ Fast but noisy

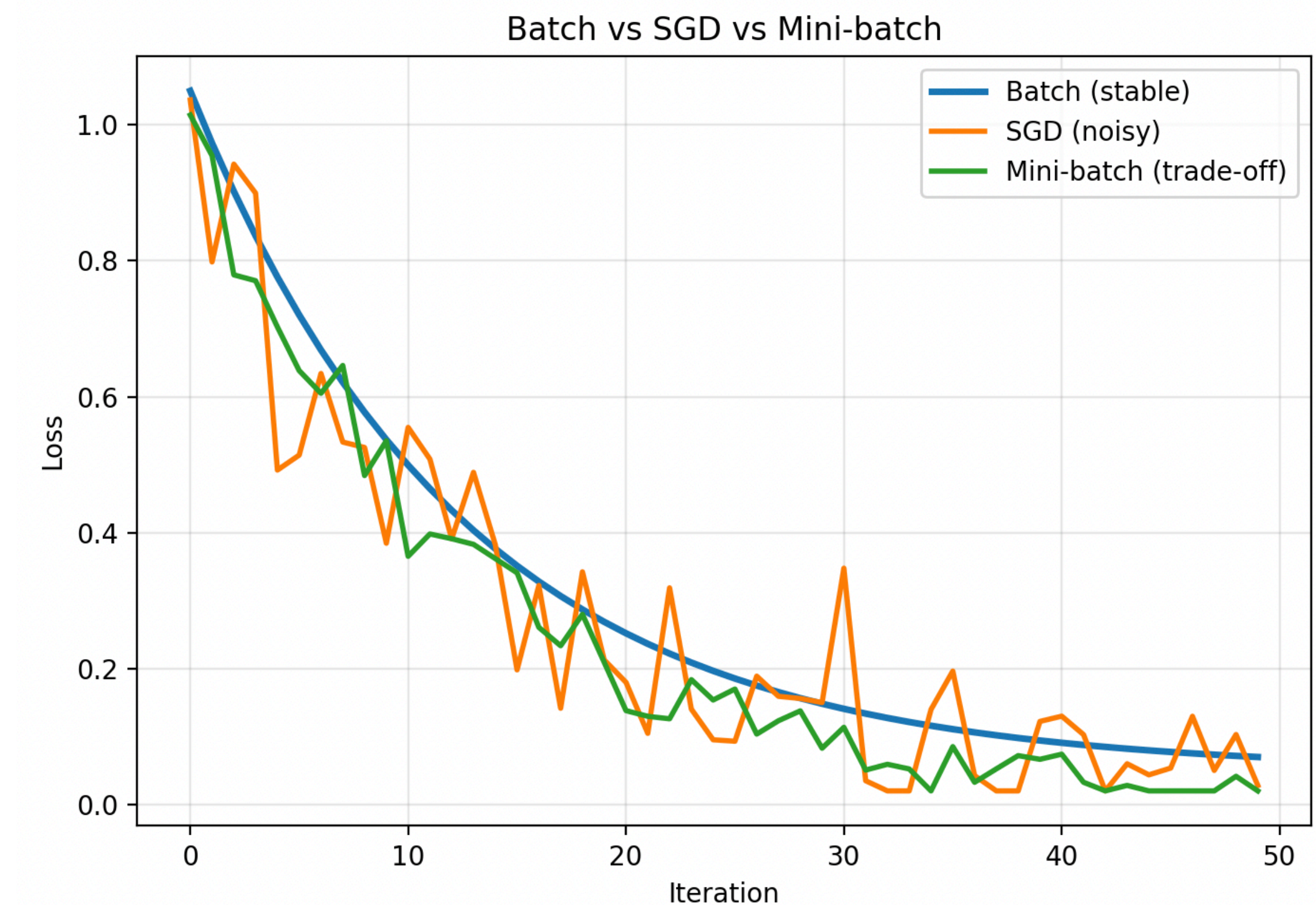
$$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathcal{L}(y_j, f(x_j; \theta))$$

▶ Mini-batch gradient descent

■ Uses small batches

■ Best trade-off

$$\theta \leftarrow \theta - \eta \frac{1}{|B|} \sum_{i \in B} \nabla_{\theta} \mathcal{L}(y_i, f(x_i; \theta))$$



Batch vs Stochastic Gradient Descent

📌 Gradient uses all data

📌 How much data should we use per step?

▶ Batch gradient descent

■ Uses all data

■ Stable but slow

▶ Stochastic gradient descent

■ Uses one sample

■ Fast but noisy

▶ Mini-batch gradient descent

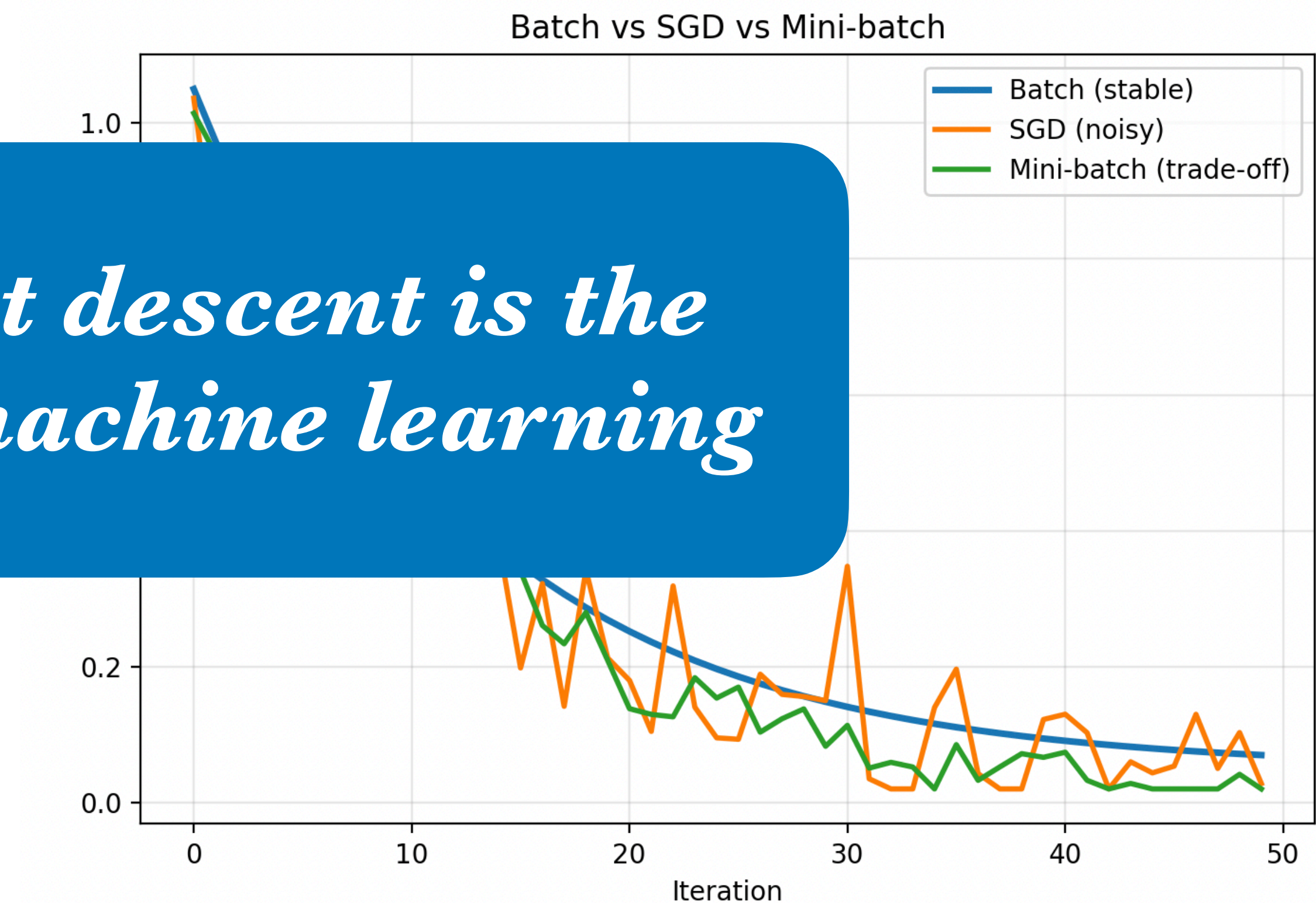
■ Uses small batches

■ Best trade-off

Mini-batch gradient descent is the default in modern machine learning

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \mathcal{L}(y_j, f(x_j; \theta))$$

$$\theta \leftarrow \theta - \eta \frac{1}{|B|} \sum_{i \in B} \nabla_{\theta} \mathcal{L}(y_i, f(x_i; \theta))$$



Advanced optimizers (Adam)

📌 Gradient may not be enough

▶ Same learning rate for all parameters

▶ Can be slow or unstable

📌 Modern optimizers improve this

▶ Momentum → smooth updates

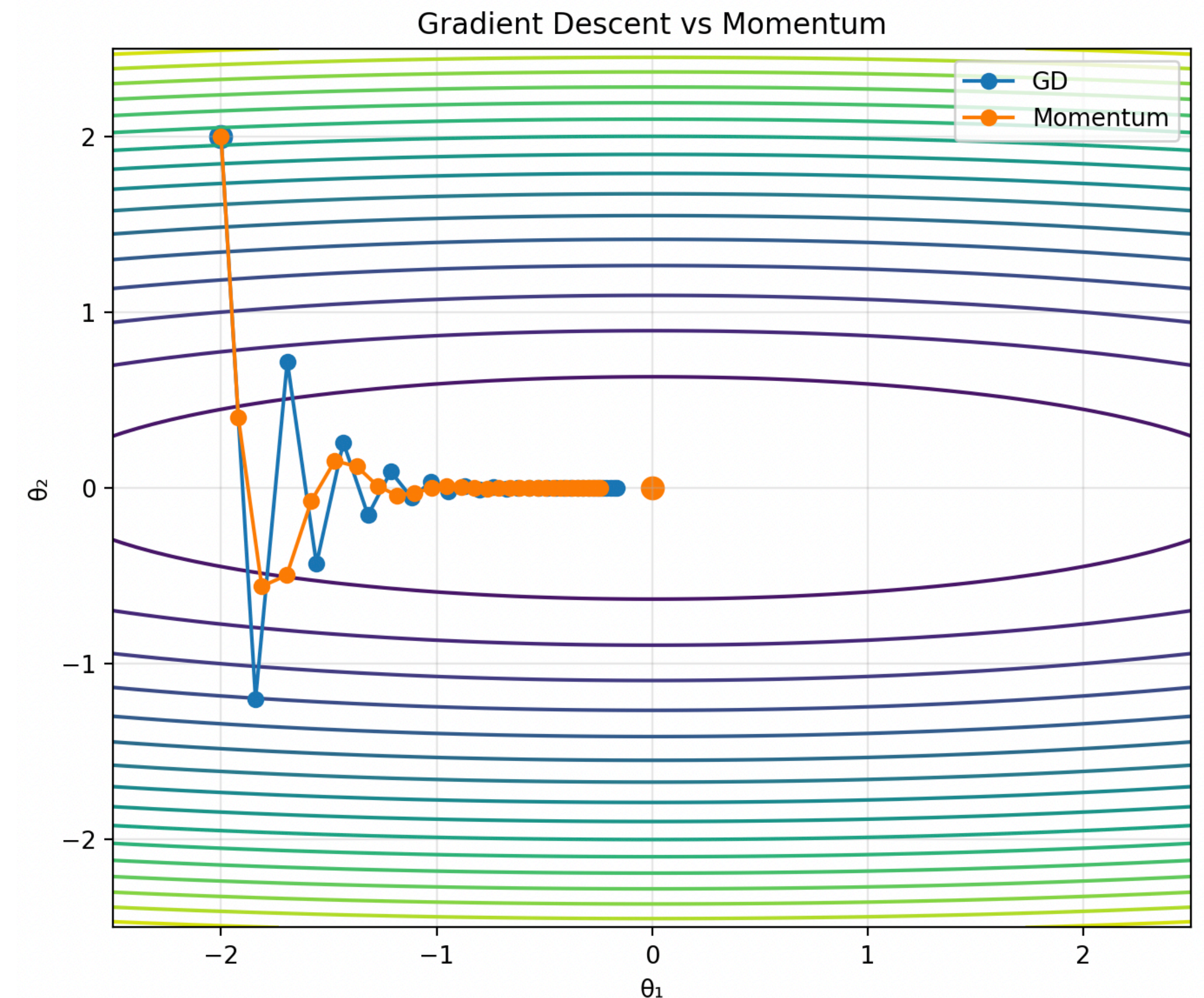
$$v_t = \beta v_{t-1} + \nabla \mathcal{L}(\theta), \quad \theta \leftarrow \theta - \eta v_t$$

▶ RMSprop → adaptive step sizes

$$g_t = \nabla \mathcal{L}(\theta_t), \quad v_t = \beta v_{t-1} + (1 - \beta) g_t^2, \quad \theta \leftarrow \theta - \eta \frac{g_t}{\sqrt{v_t} + \epsilon}$$

▶ Adam → combines both

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \quad \theta \leftarrow \theta - \eta \frac{m_t}{\sqrt{v_t} + \epsilon}$$



Advanced optimizers (Adam)

📌 Gradient may not be enough

➤ Same learning rate for all parameters

➤ Can be slow or unstable

📌 Modern optimizers

➤ Momentum →

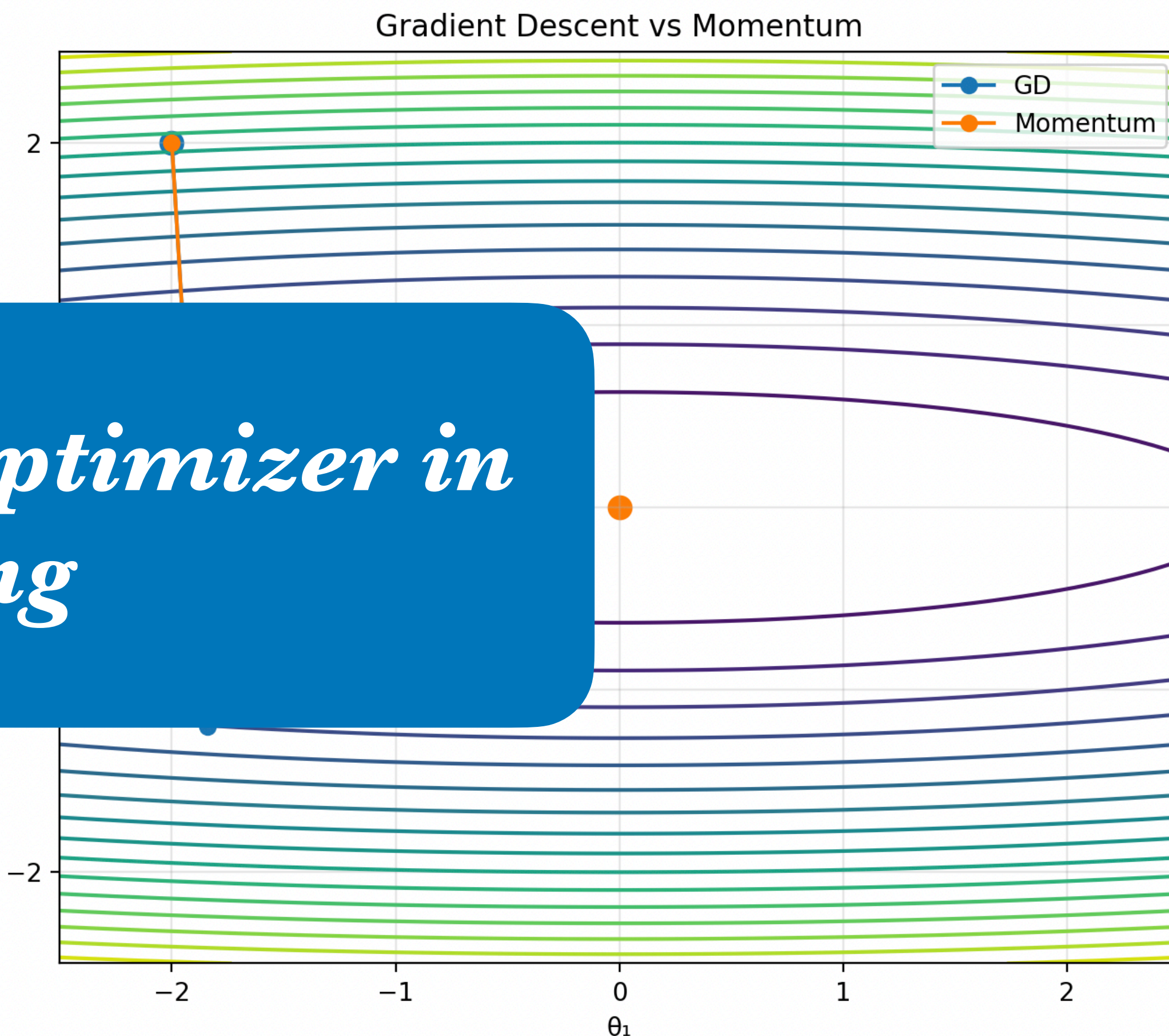
$$v_t = \beta v_{t-1} + \nabla \mathcal{L}(\theta_t)$$

➤ RMSprop → adaptive step sizes

$$g_t = \nabla \mathcal{L}(\theta_t), \quad v_t = \beta v_{t-1} + (1 - \beta)g_t^2, \quad \theta \leftarrow \theta - \eta \frac{g_t}{\sqrt{v_t} + \epsilon}$$

➤ Adam → combines both

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t, \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2, \quad \theta \leftarrow \theta - \eta \frac{m_t}{\sqrt{v_t} + \epsilon}$$



Adam is the default optimizer in deep learning

Model evaluation

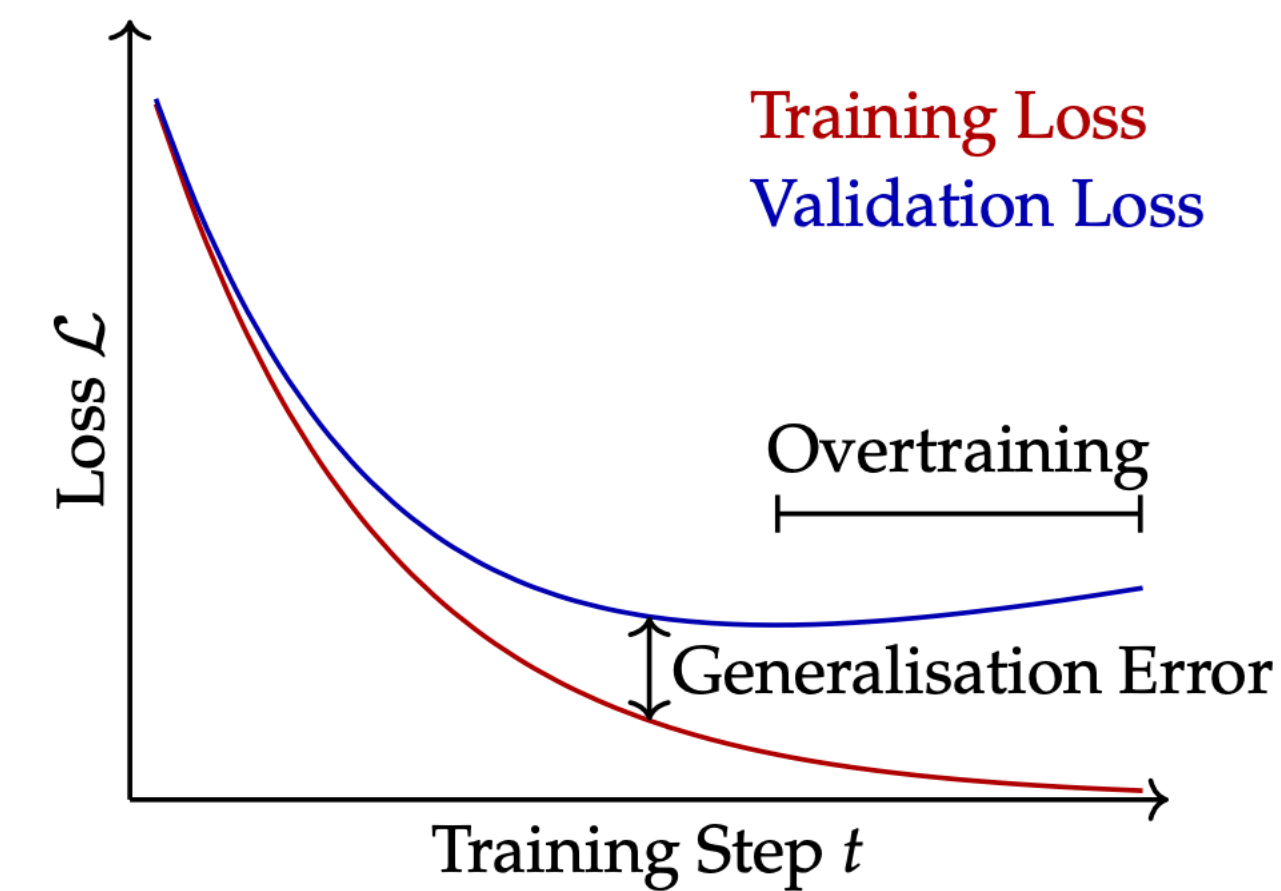
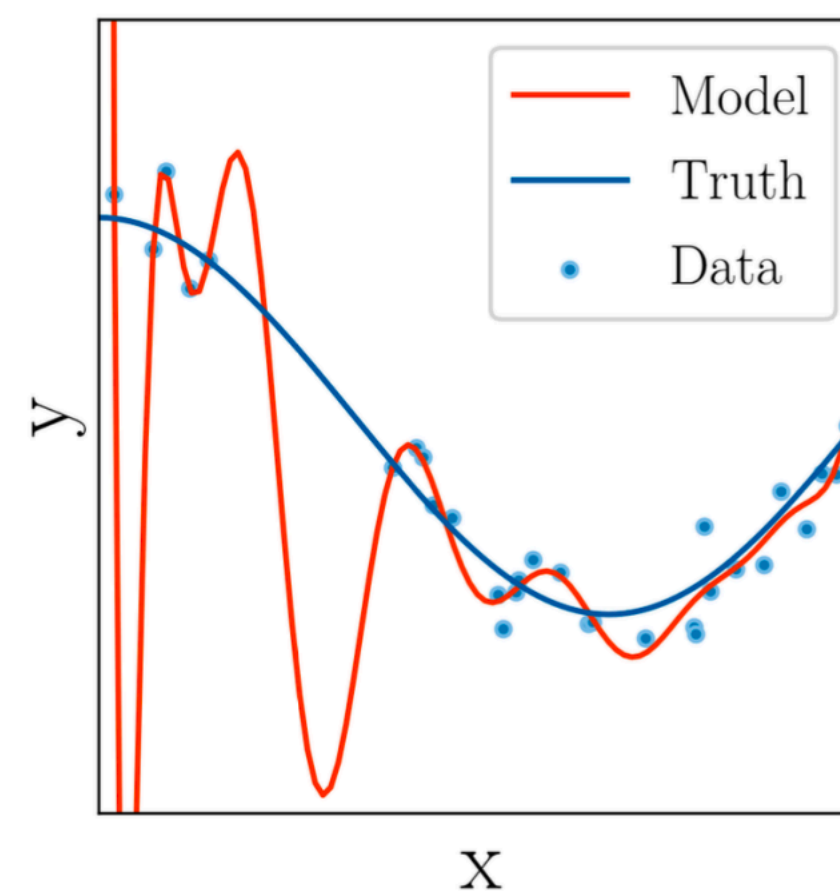
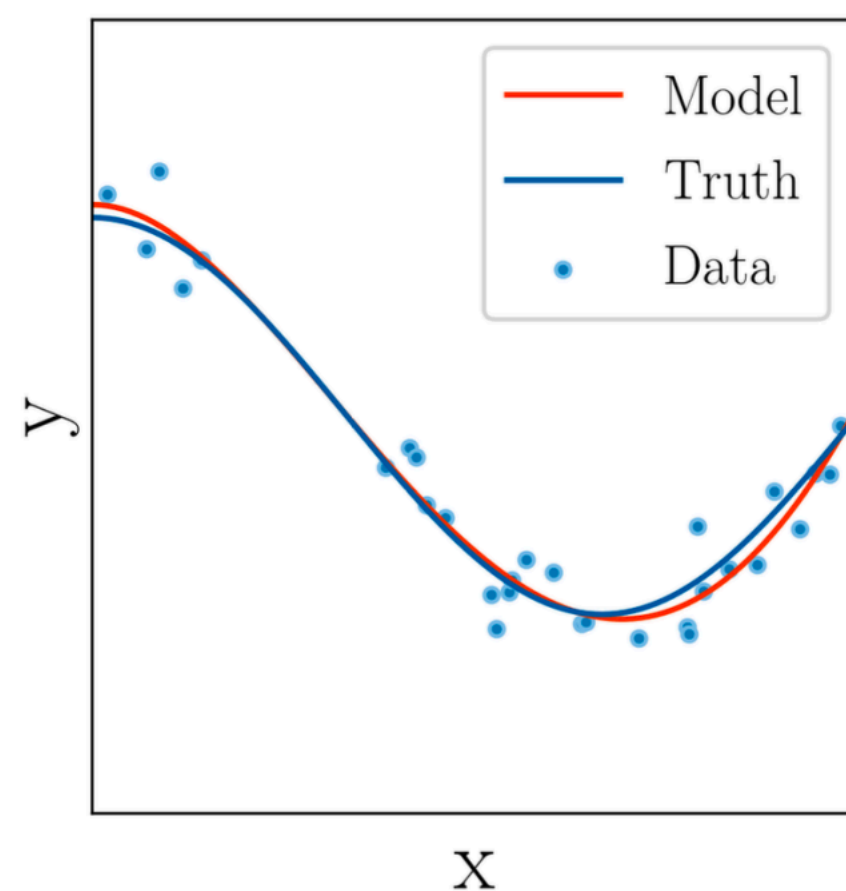
- 📌 A complex model can be fitted to any function if trained for long enough

- 📌 It might perform well on training data, but not **generalize** (**overtraining**)

 - ▶ **Generalization**: keep performance on new, unseen data

- 📌 To measure this:

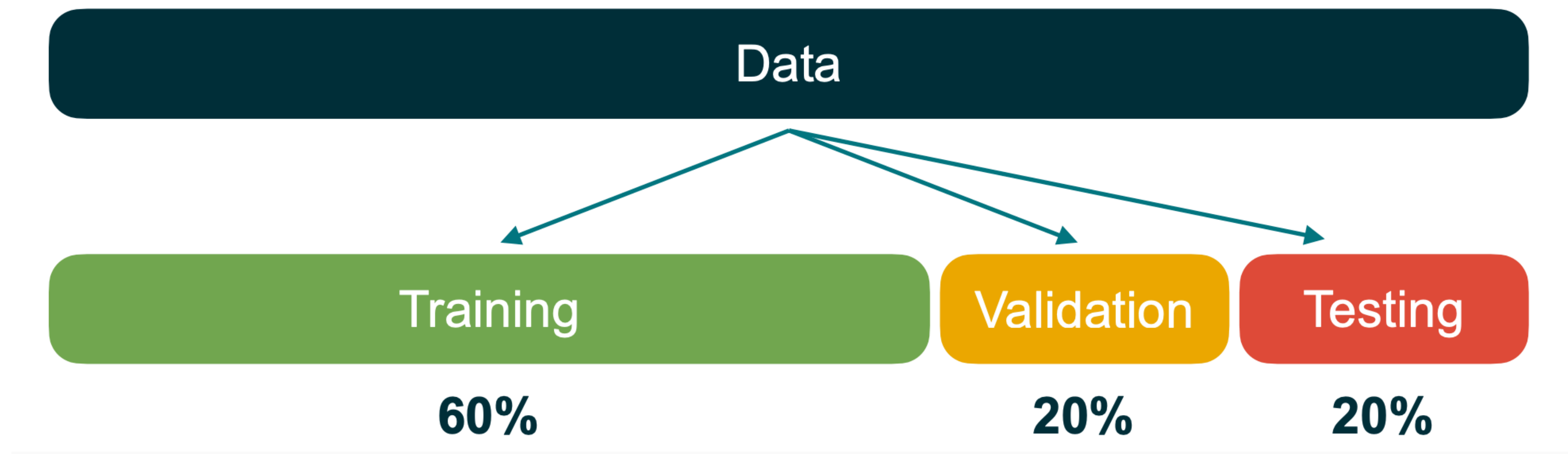
 - ▶ Split data into train / validation / test



Train-val-test split

📌 Data splits:

- ▶ **Training set**: main split, used to train parameters of the model
- ▶ **Validation set**: to monitor and tune training procedure (e.g. learning rate)
- ▶ **Testing set**: to estimate the final performance (used only once)



Data splitting: cross-validation

- 📌 We do not want to loose data due to splitting
- 📌 Solution: use **rotating cross-validation** to train multiple independent models



Types of problems: supervised learning

📌 Given N observations (events in HEP) with features $\{x_i \in \mathcal{X}\}$ (training data) and targets $\{y_i \in \mathcal{Y}\}$:

▶ Learn the function mapping $y = f(x)$

▶ $\mathbf{x}_i$ is typically a n -dimensional vector (n = number of features)

▶ y_i is typically a scalar

📌 Examples:

▶ Classification

▶ Regression

Supervised learning: Classification

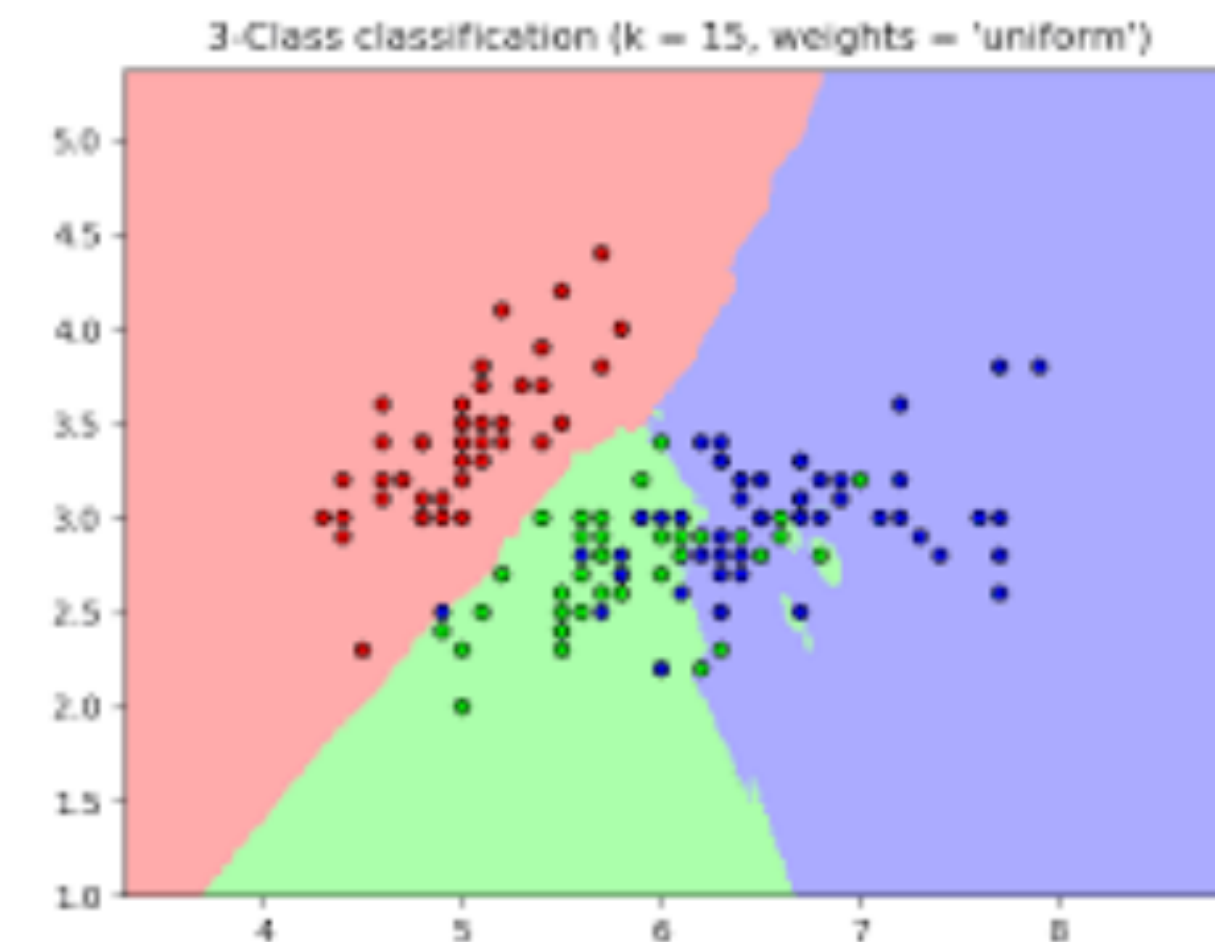
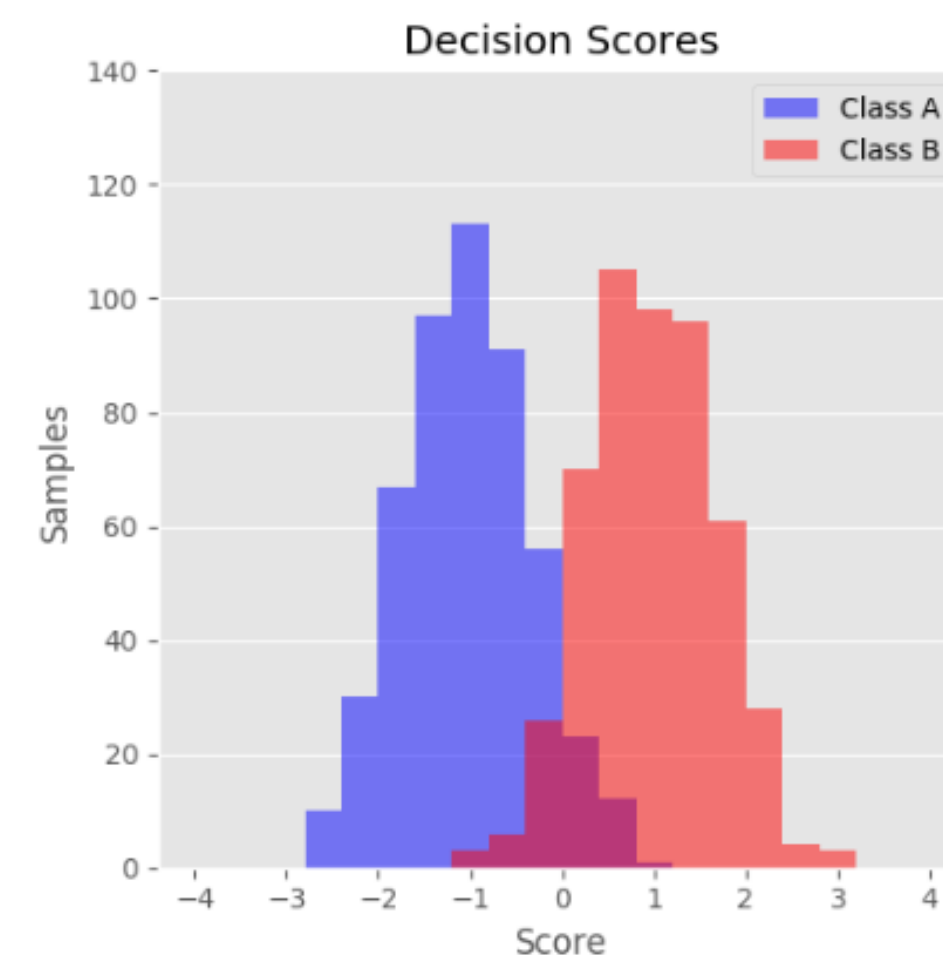
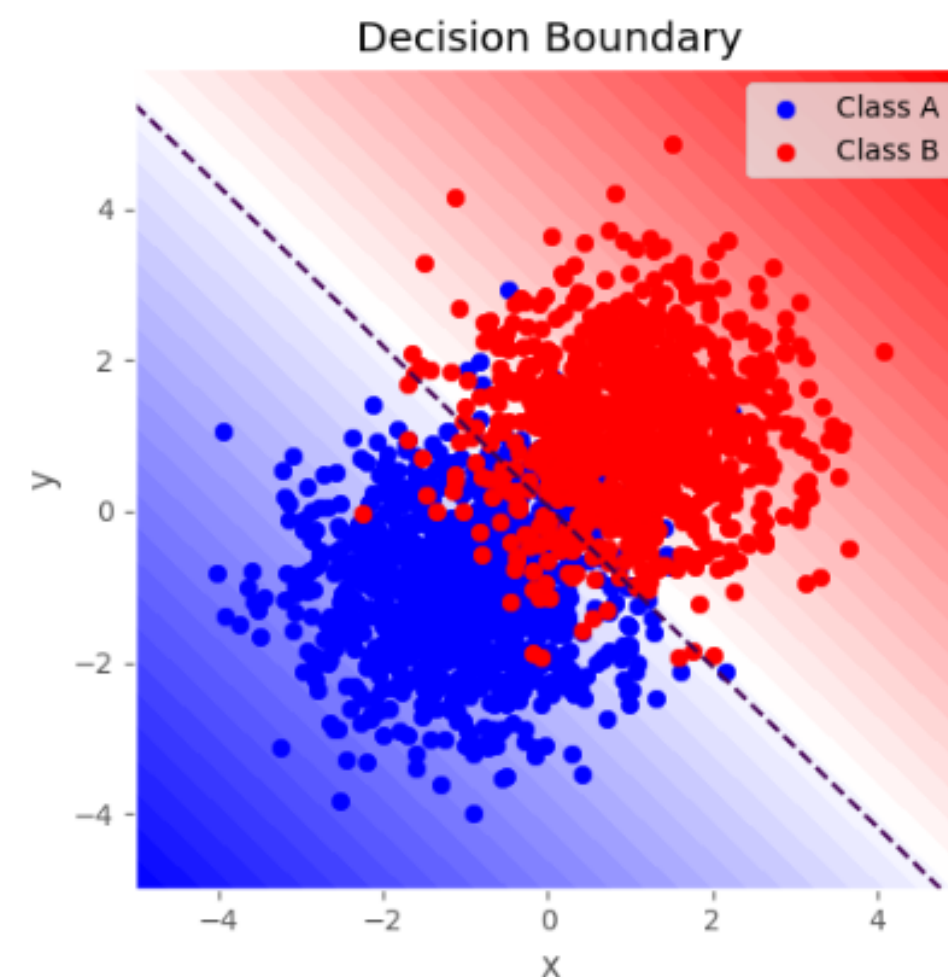
📌 Features $\{x_i \in \mathcal{X}\}$ and targets $\{y_i \in \mathcal{Y}\}$

📌 Classification:

▶ $\mathcal{Y}$ is a finite set of labels

▶ Binary classification: $\mathcal{Y} = \{0,1\}$

▶ Multi-class classification: $\mathcal{Y} = \{c_0, c_1, \dots, c_n\}$

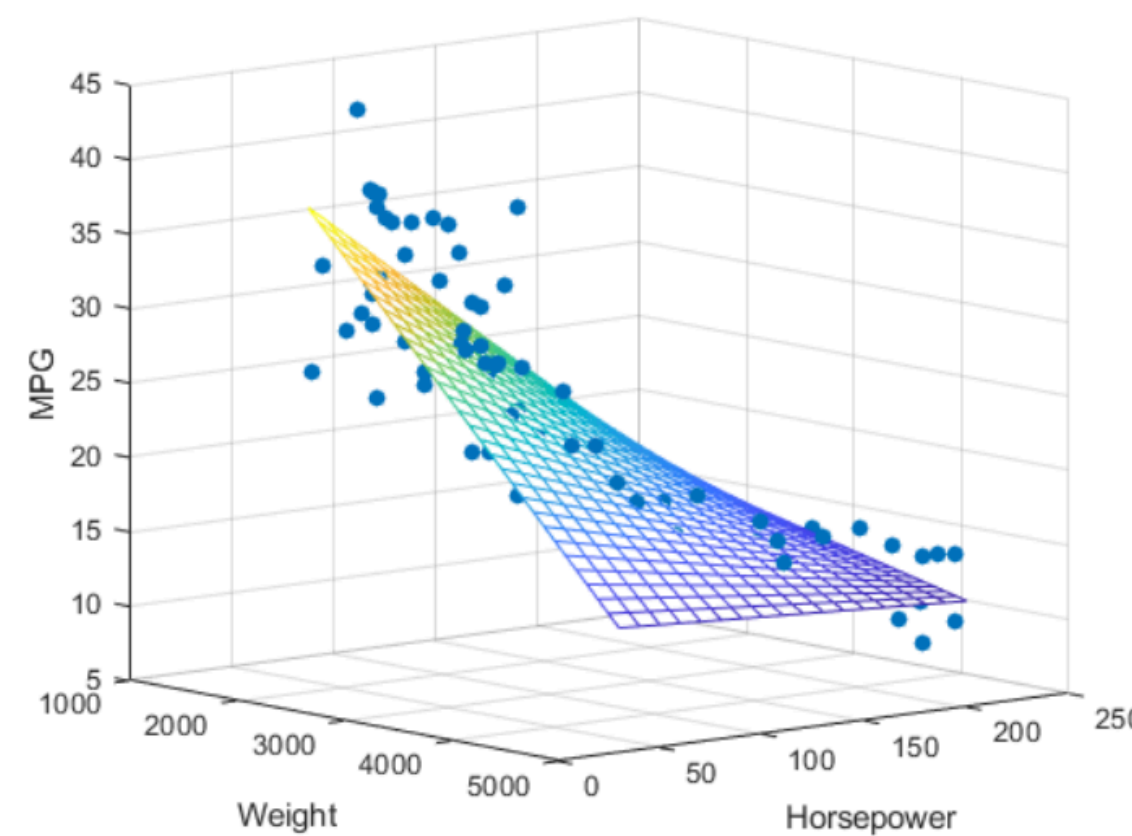
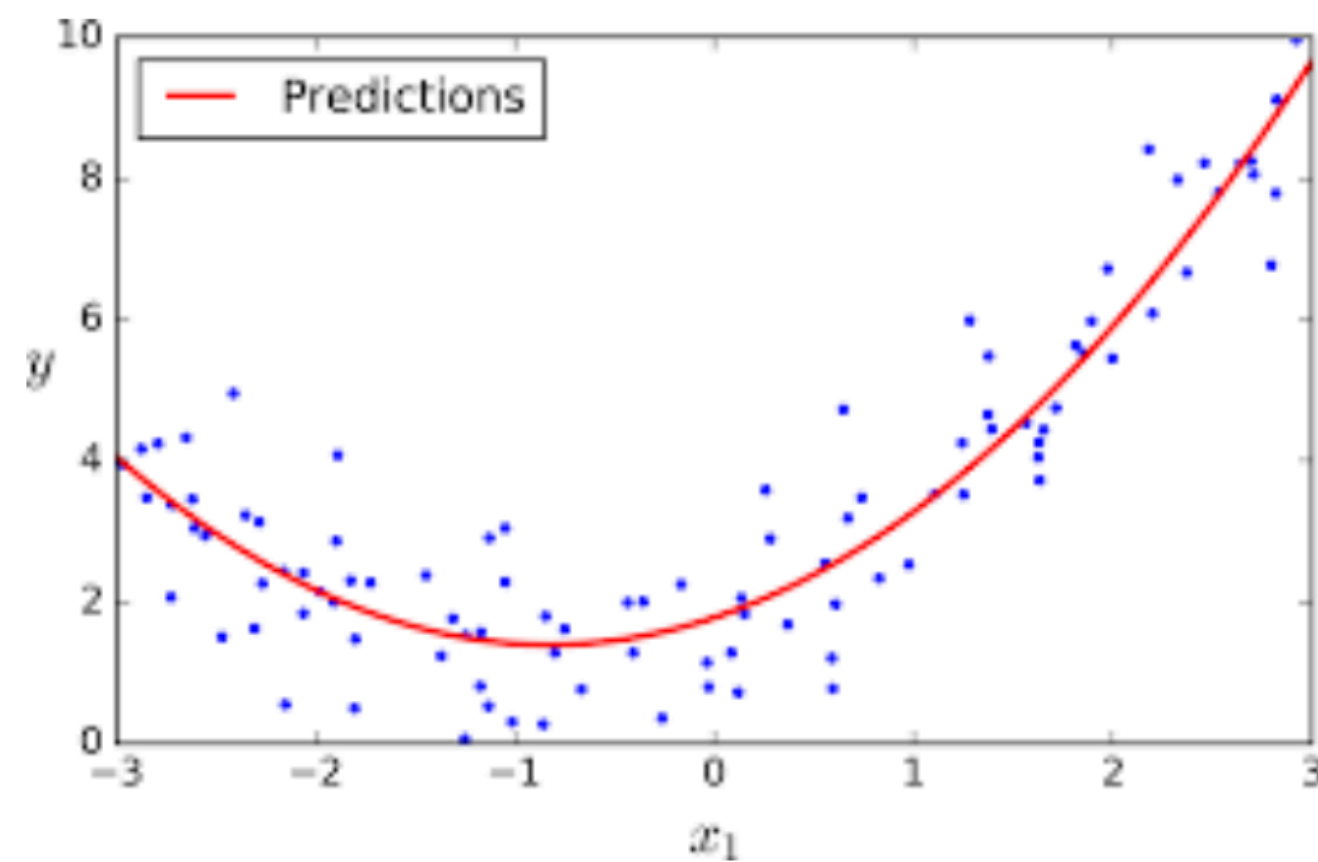


Supervised learning: Regression

📌 Features $\{x_i \in \mathcal{X}\}$ and targets $\{y_i \in \mathcal{Y}\}$

📌 Regression:

▶ $\mathcal{Y}$ are real numbers

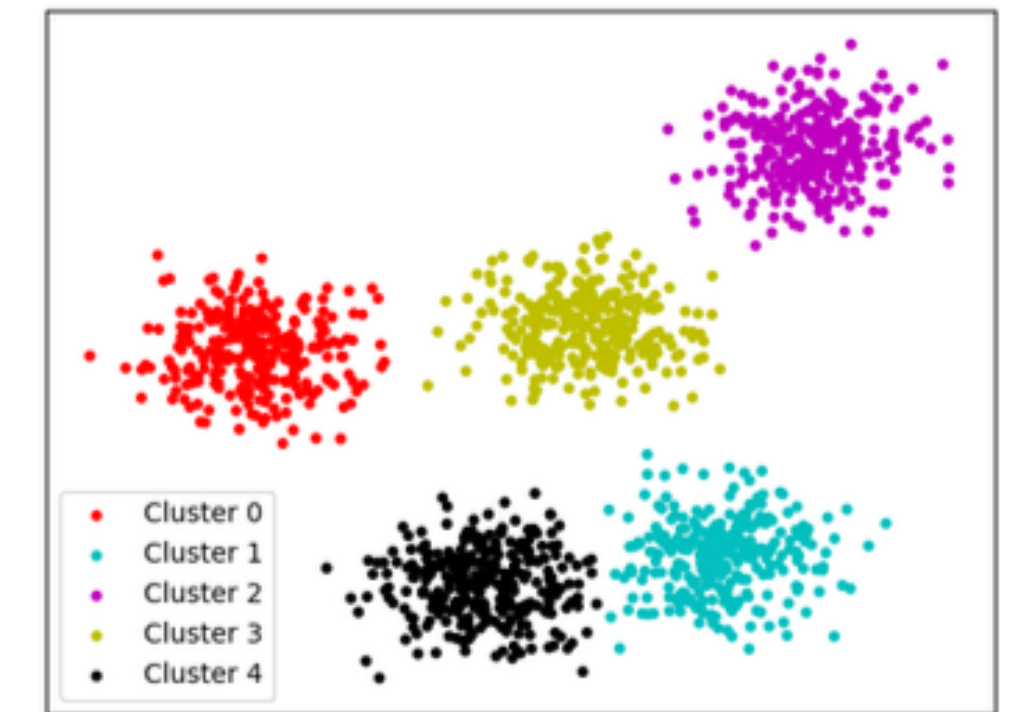
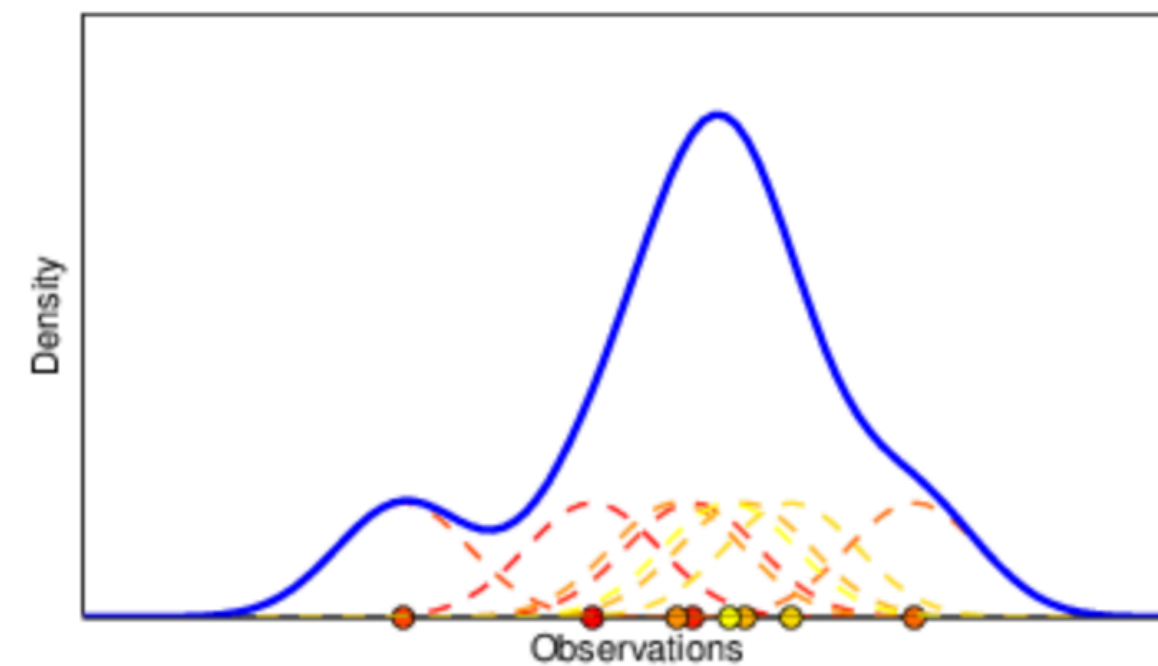


Type of problem: unsupervised learning

- 📌 Data $D = \{x_i\}$, but no labels for classes
- 📌 Learn possible structures or representations from data
- 📌 Examples:

► Density estimation

- Learn $p(x)$
- Can generate new data

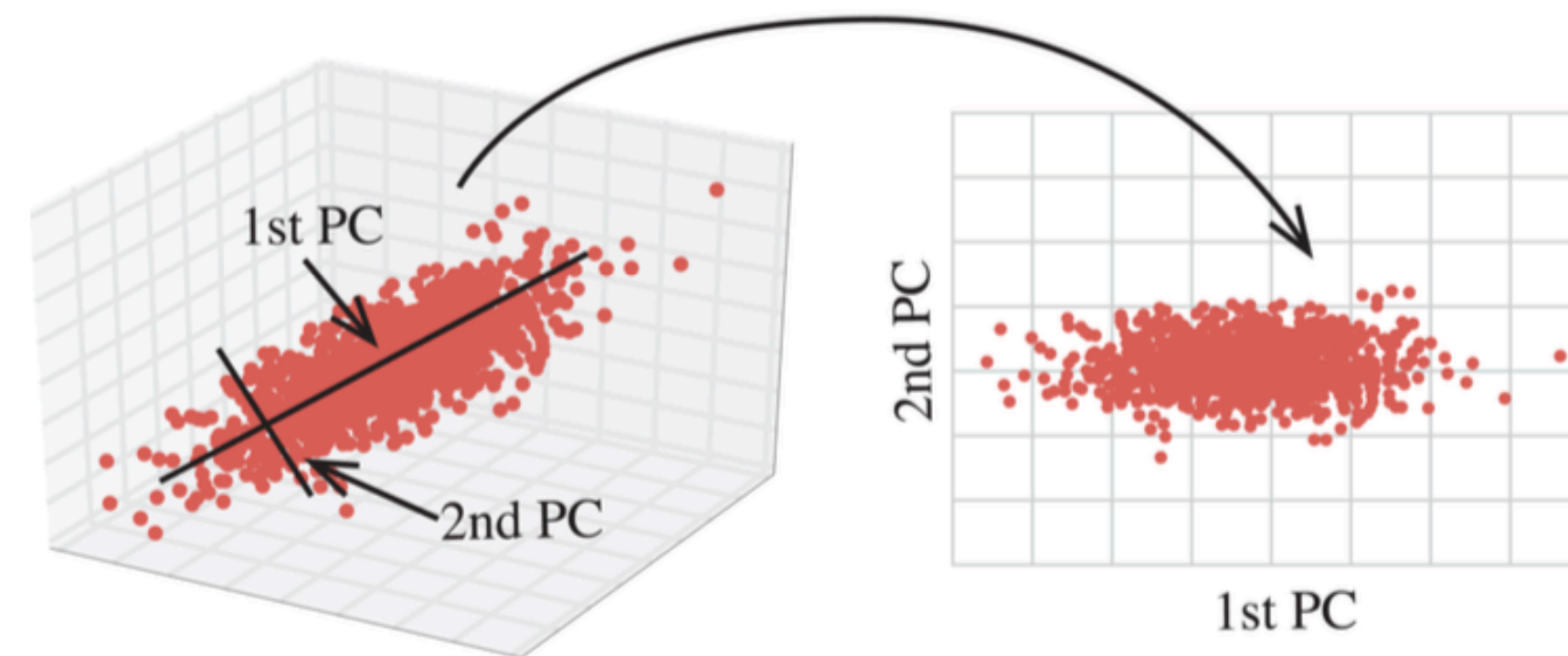


► Clustering

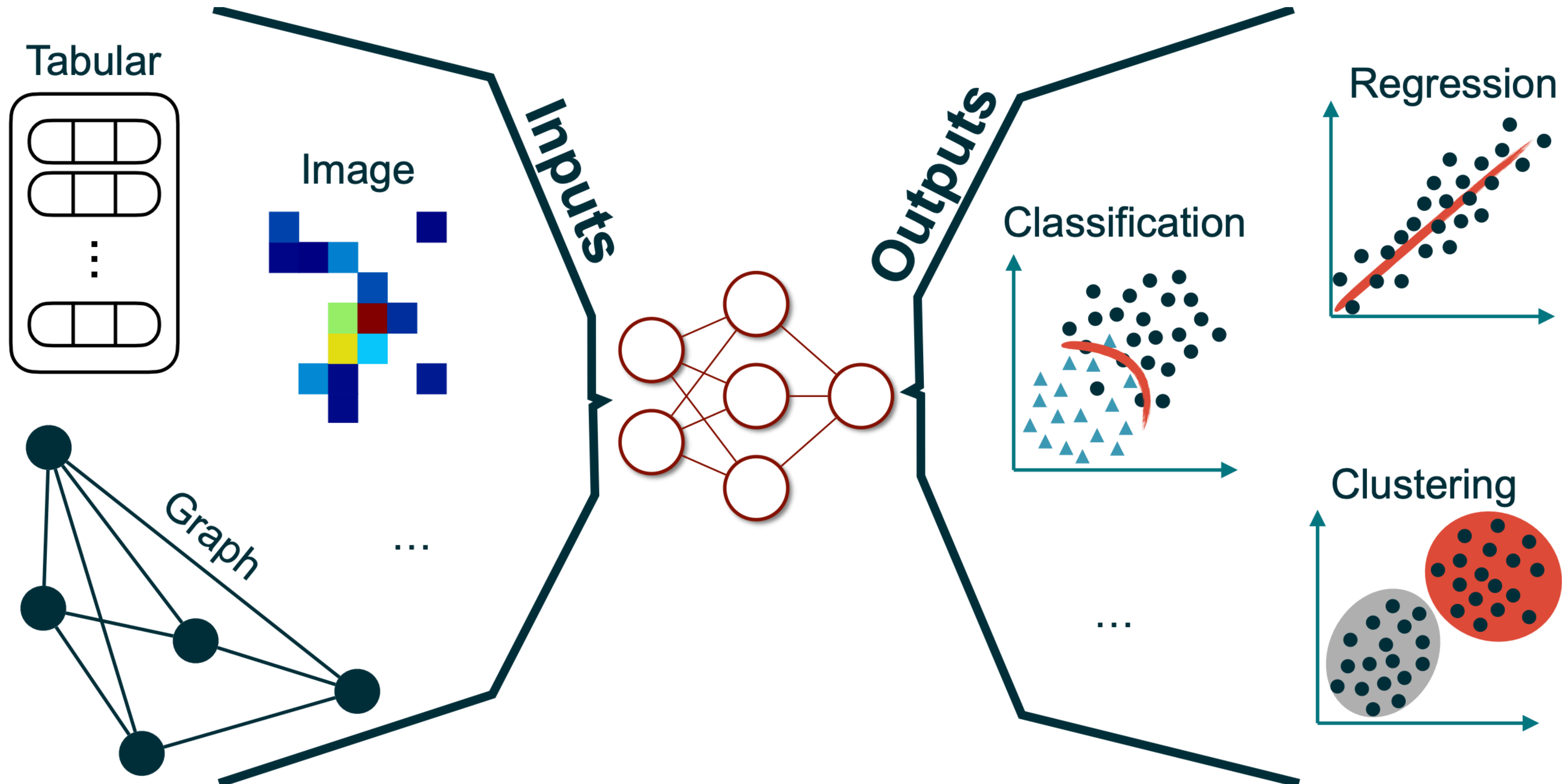
- Group similar data points

► Dimensionality reduction

- Represent data in fewer dimensions



ML tasks



ML tasks in HEP

Classification

- ▶ Particle identification
- ▶ Pattern recognition (tracks)
- ▶ Searches for new physics

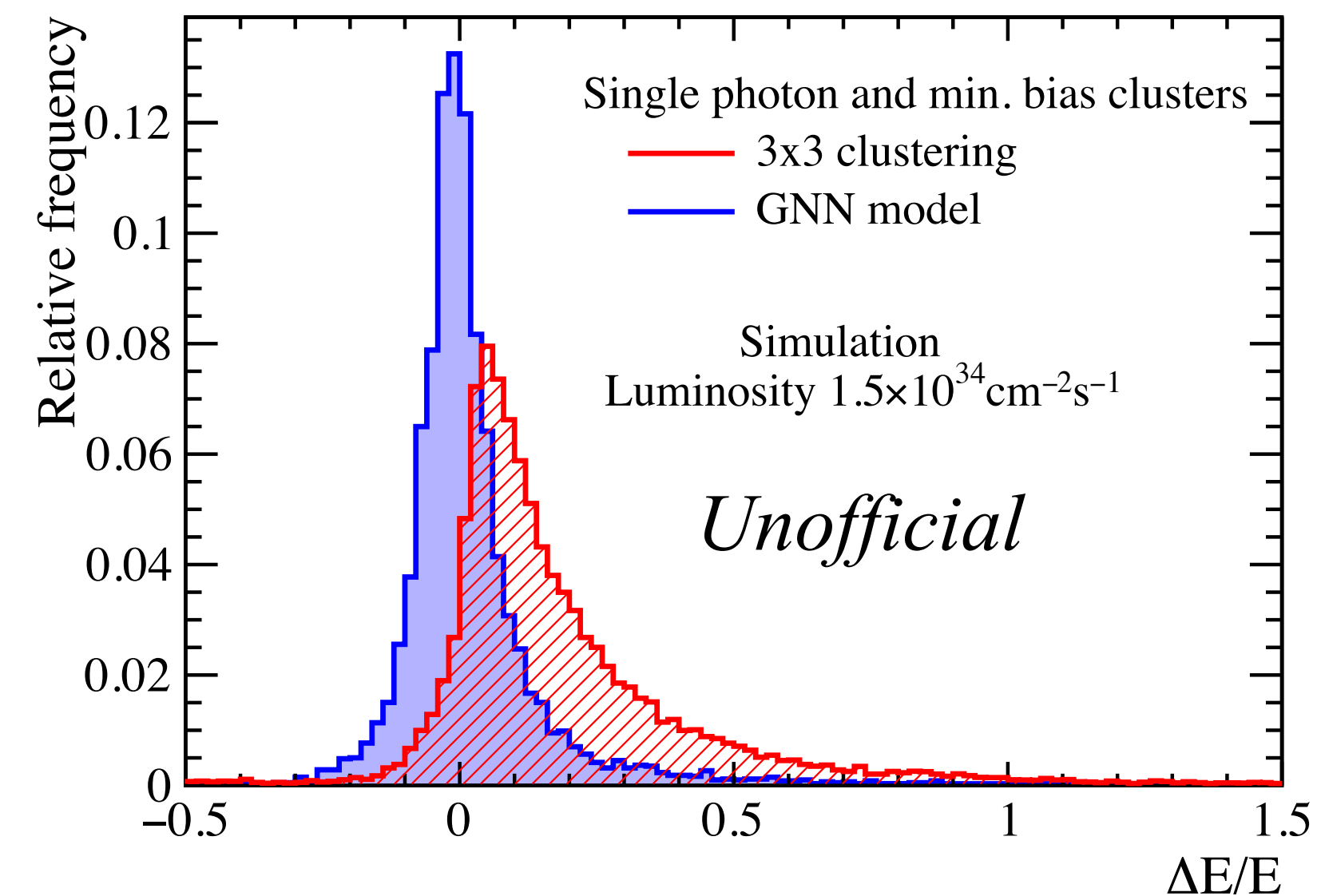
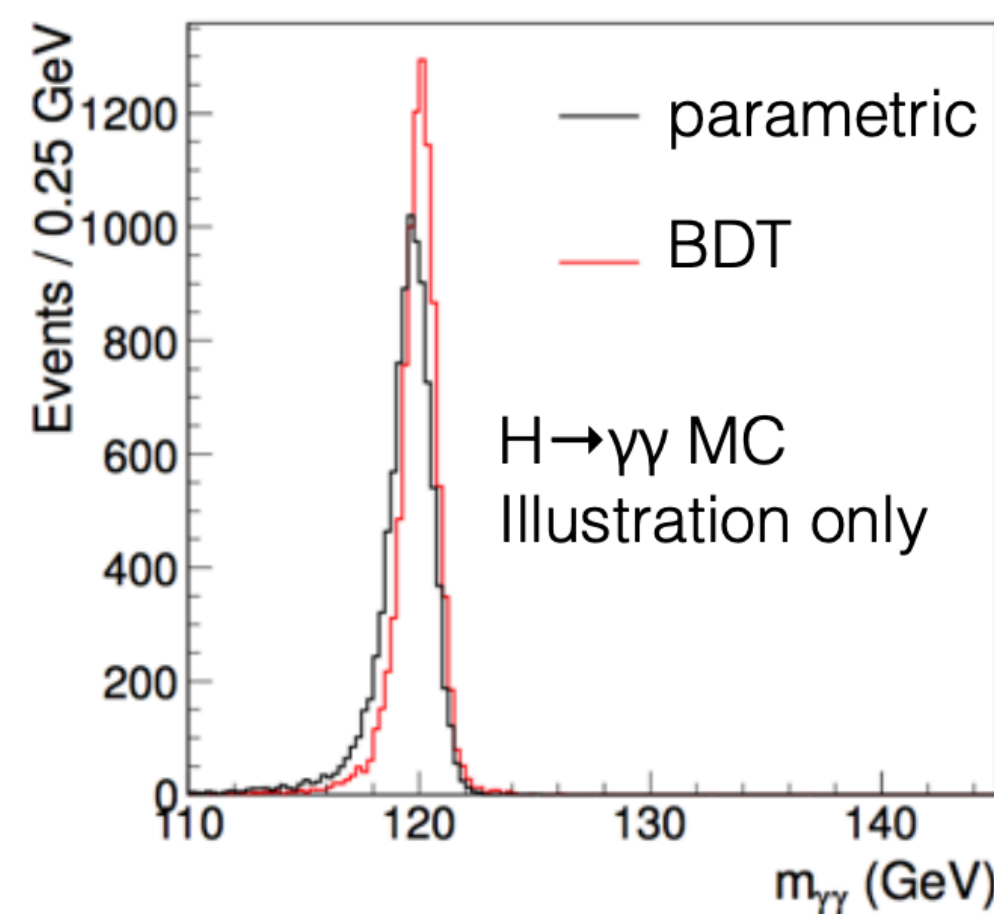
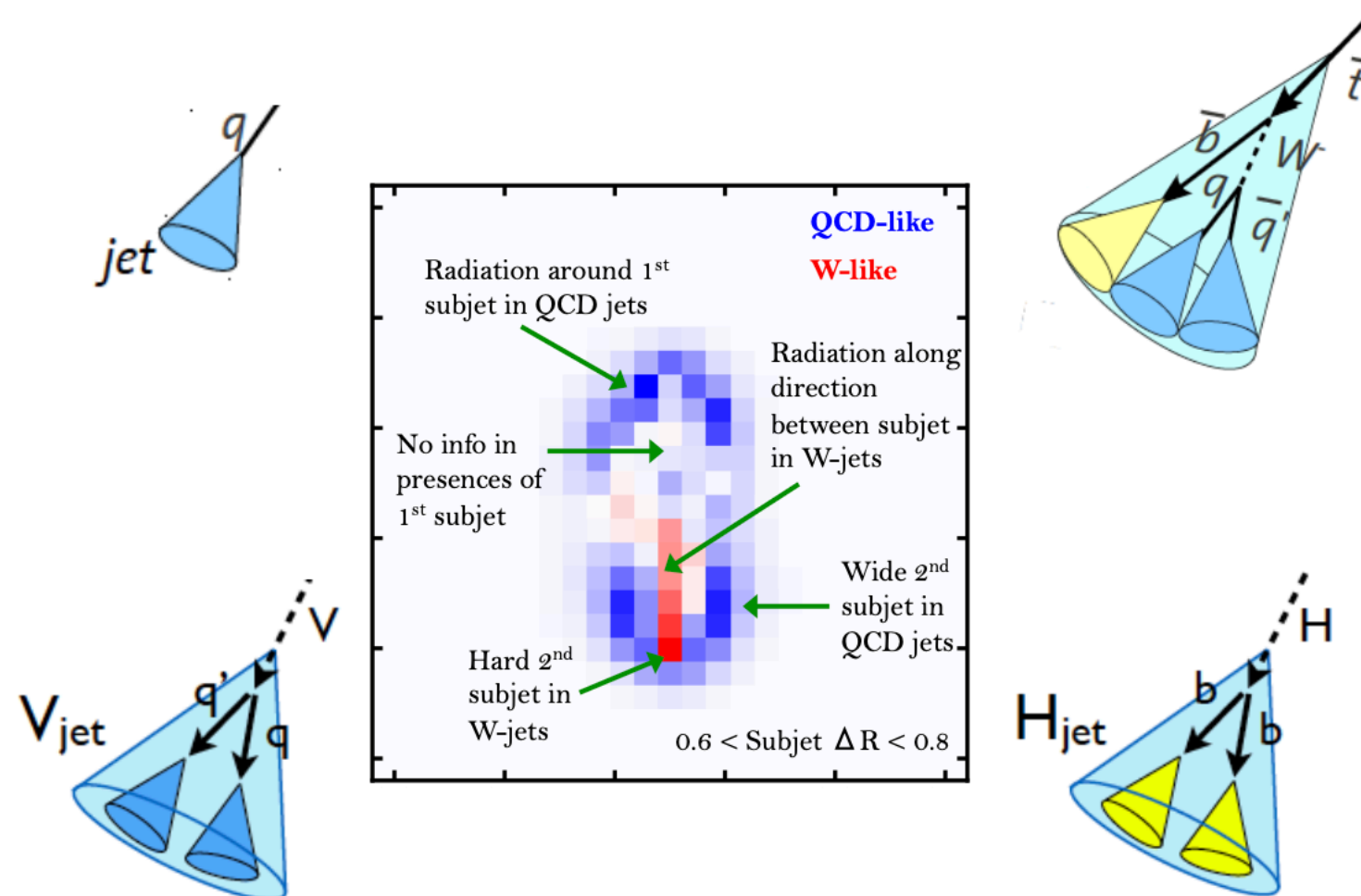
Regression

- ▶ Function estimate
- e.g. particle energy

Density estimation

- ▶ Fast simulation, likelihood

Anomaly detection



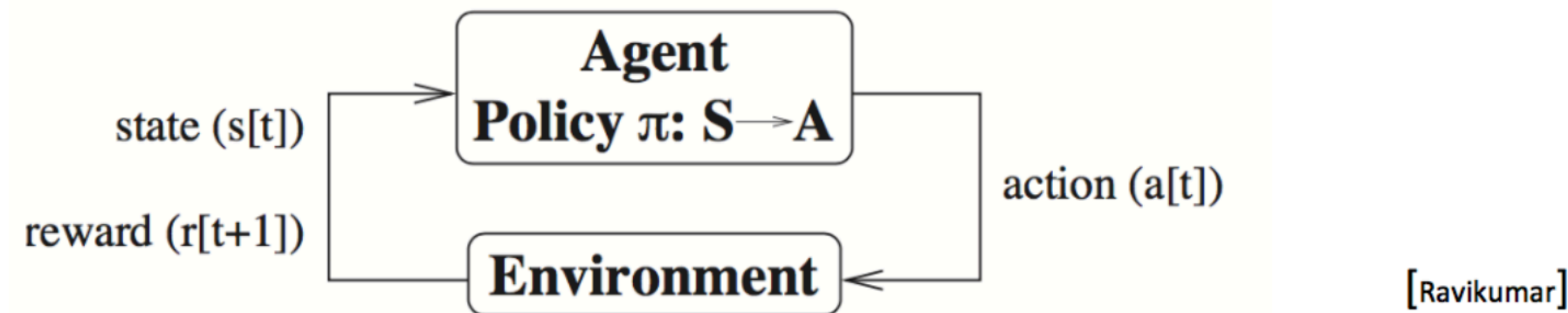


Thank you!



Backup

Reinforcement Learning



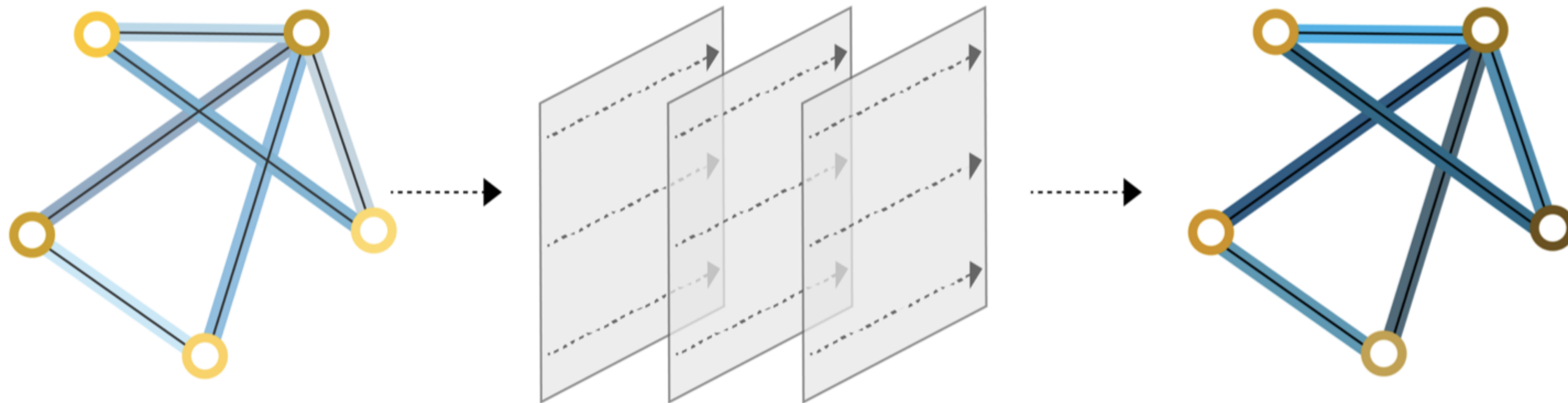
- Models for agents that take actions depending on current state
 - Actions incur rewards, and affect future states (“feedback”)
- Learn to make the best sequence of decisions to achieve a given goal when feedback is often delayed until you reach the goal

[M. Kagan]

📌 Inspired by Belle II: GNN reconstruction [Comput Software Big Sci 7, 13 (2013)]

▶ GNNs do not require the same concessions to handle irregular geometries

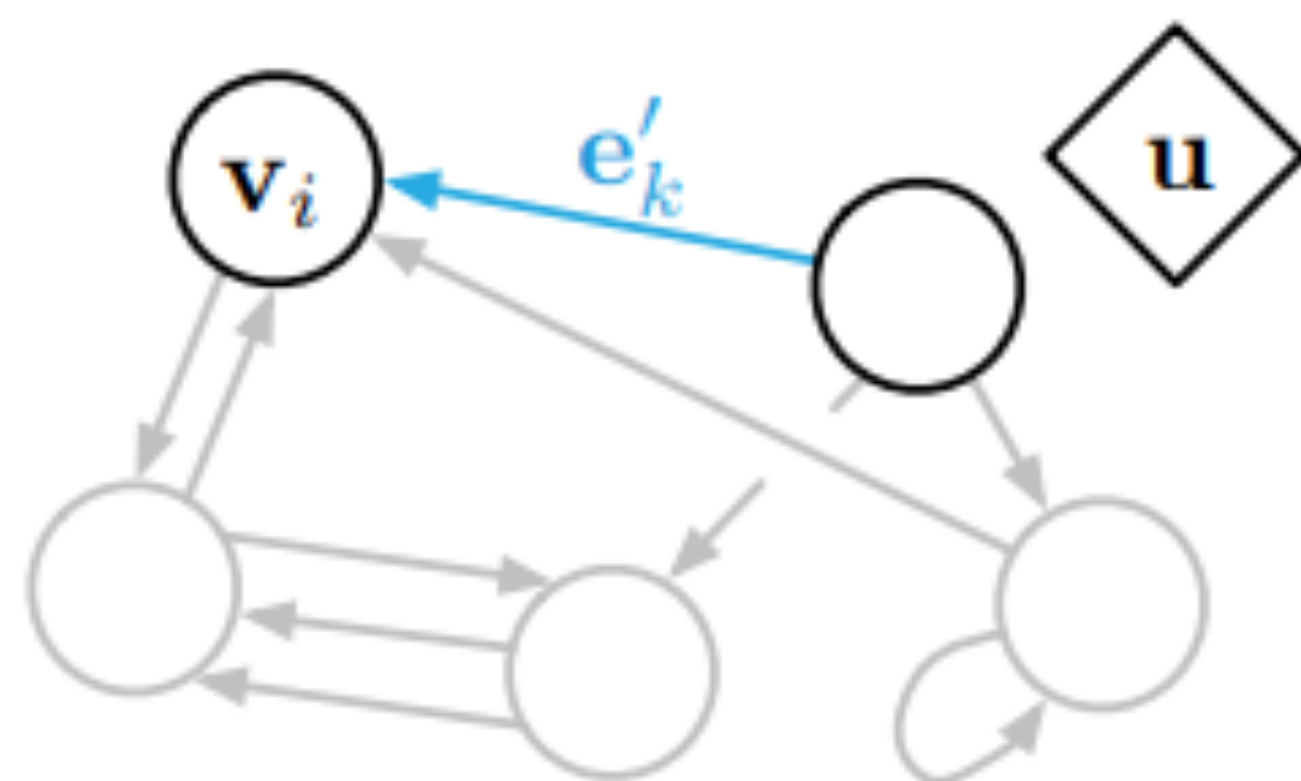
▶ Clusters represented as graphs: **cells as nodes**, **edges describing relationships among cells** (e.g. spatial distance), and **global features for the cluster** (e.g. total energy)



The message passing algorithm

- Nodes, edges, and globals are updated through aggregation and MLPs

BLUE updated by BLACK (not utilizing GREY)



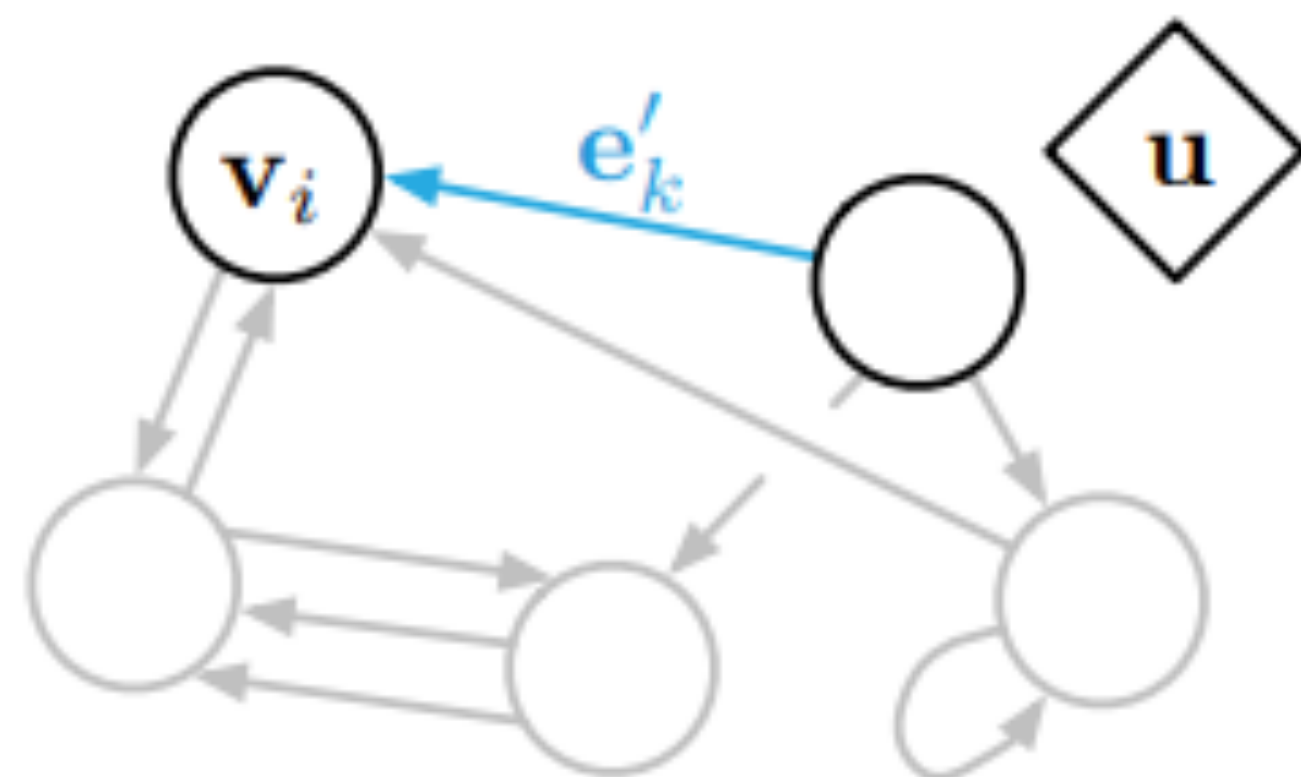
(a) Edge update

Update edges using
connected nodes and globals

The message passing algorithm

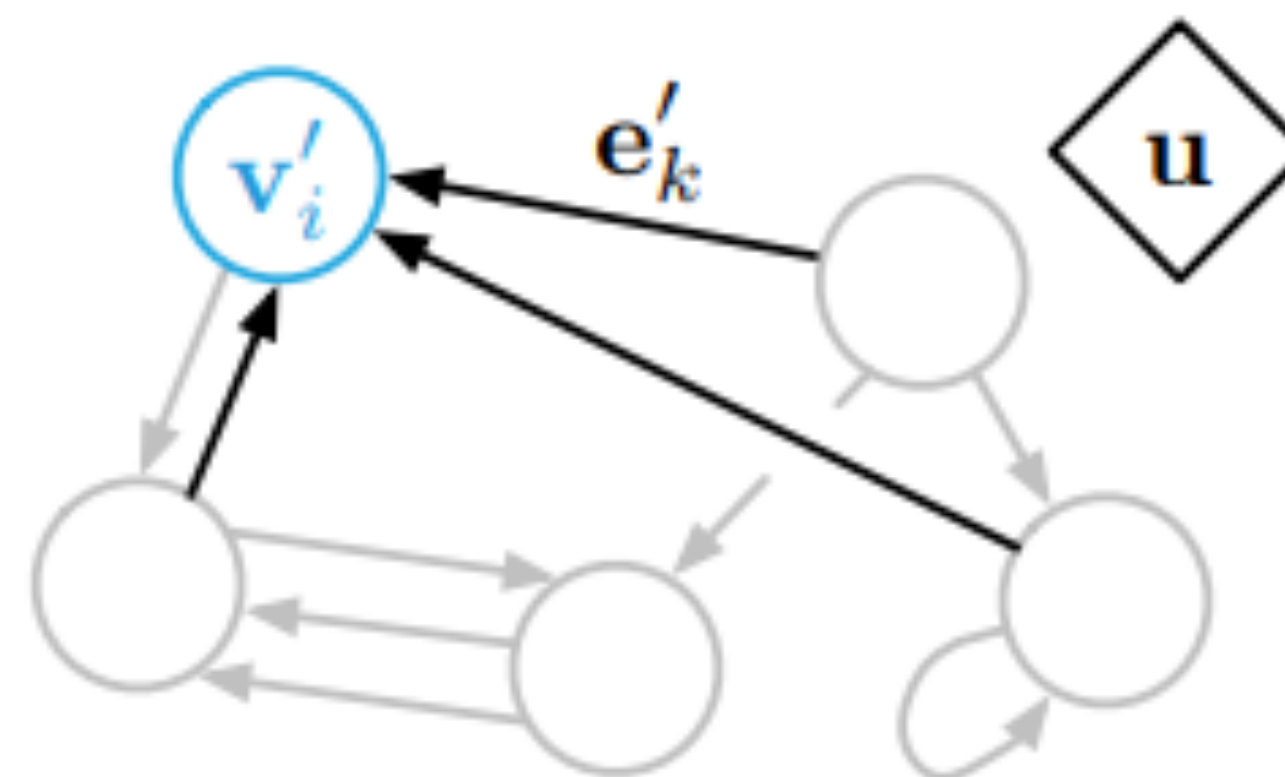
- Nodes, edges, and globals are updated through aggregation and MLPs

BLUE updated by BLACK (not utilizing GREY)



(a) Edge update

Update edges using
connected nodes and globals



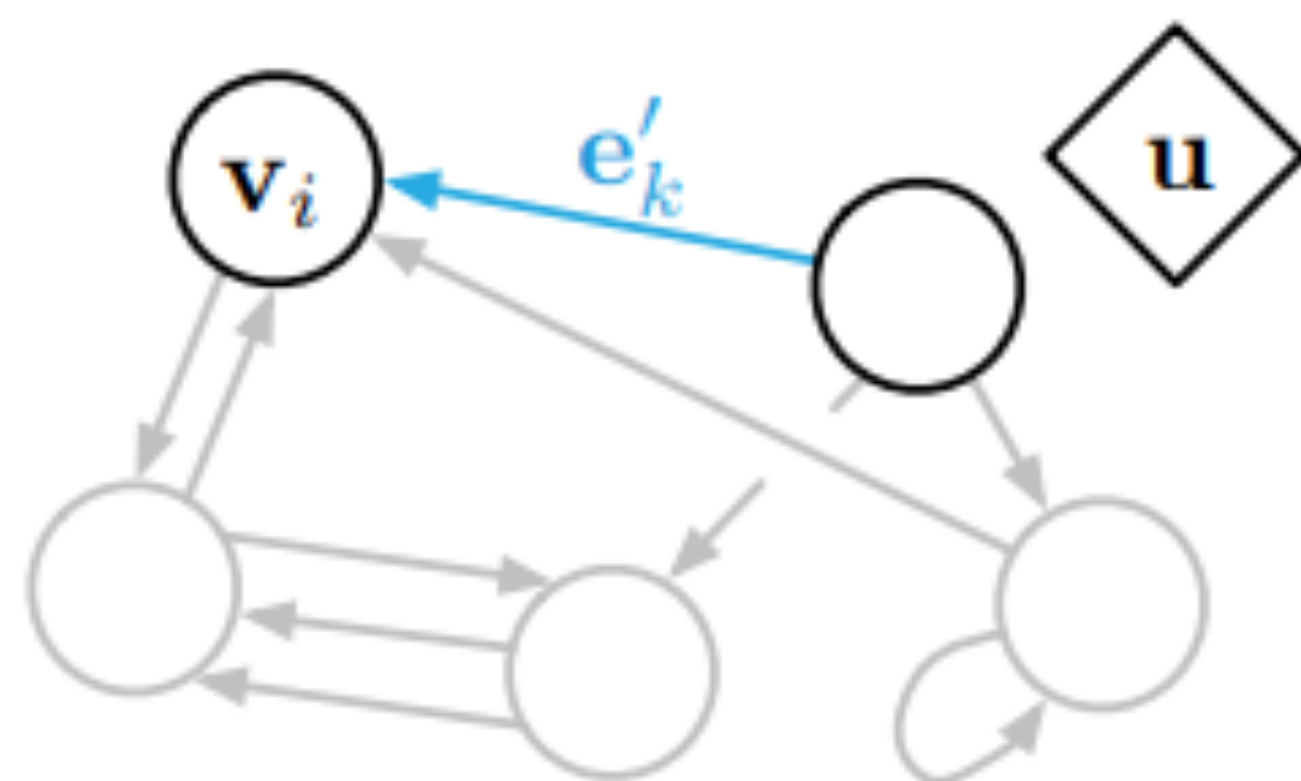
(b) Node update

Update nodes
using (new) edges
and globals

The message passing algorithm

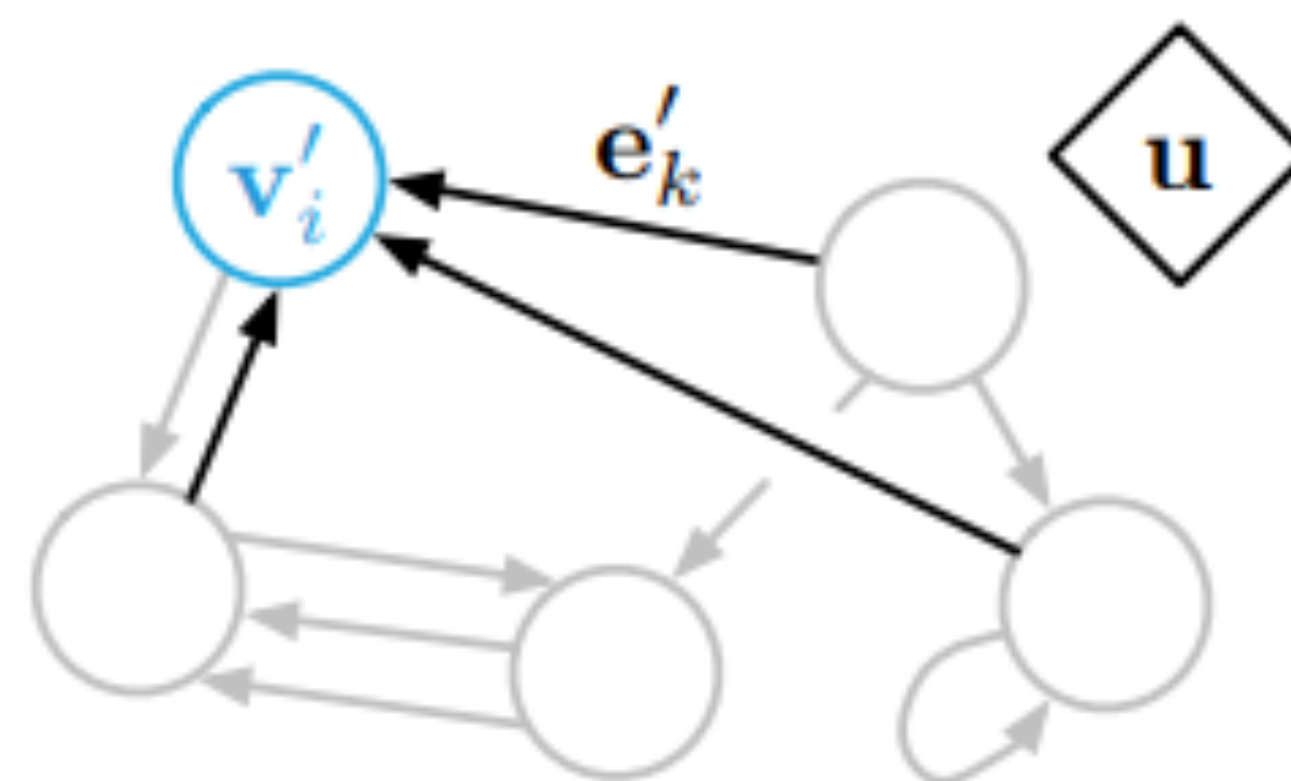
- Nodes, edges, and globals are updated through aggregation and MLPs

BLUE updated by BLACK (not utilizing GREY)



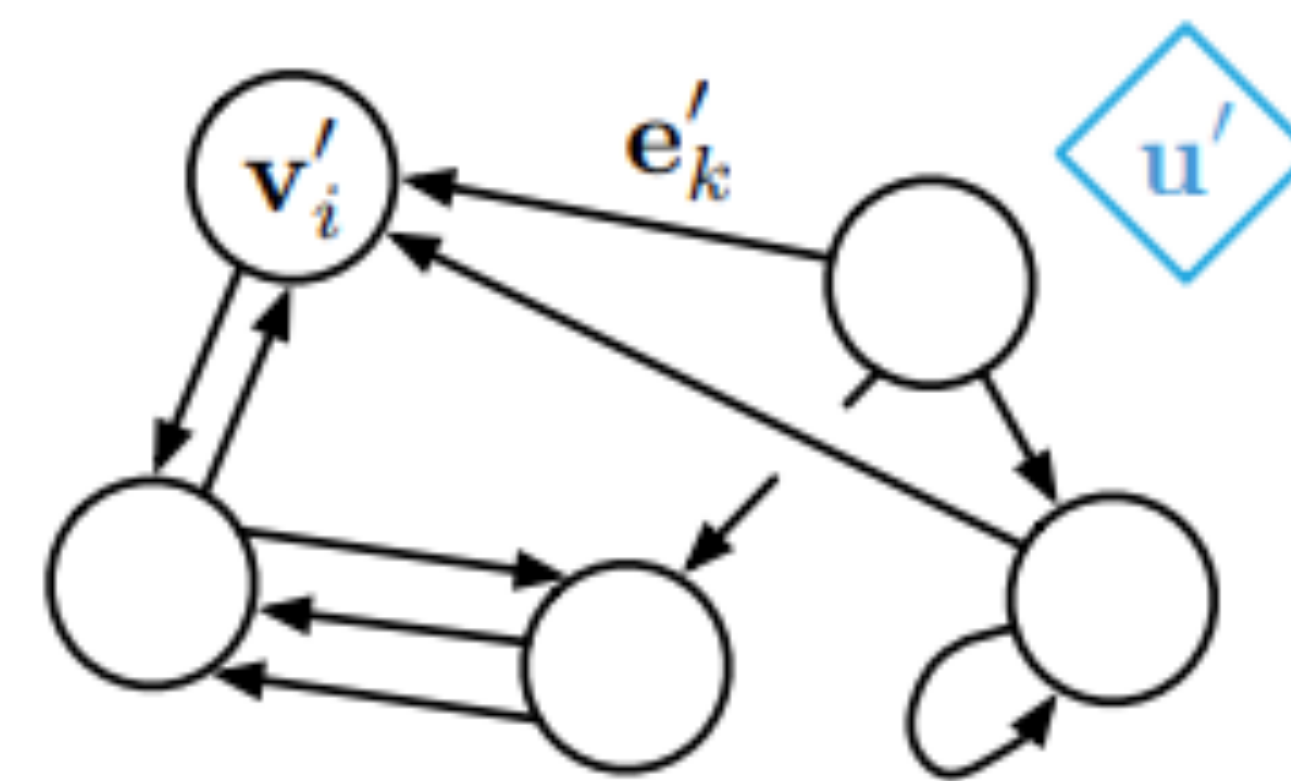
(a) Edge update

Update edges using
connected nodes and globals



(b) Node update

Update nodes
using (new) edges
and globals



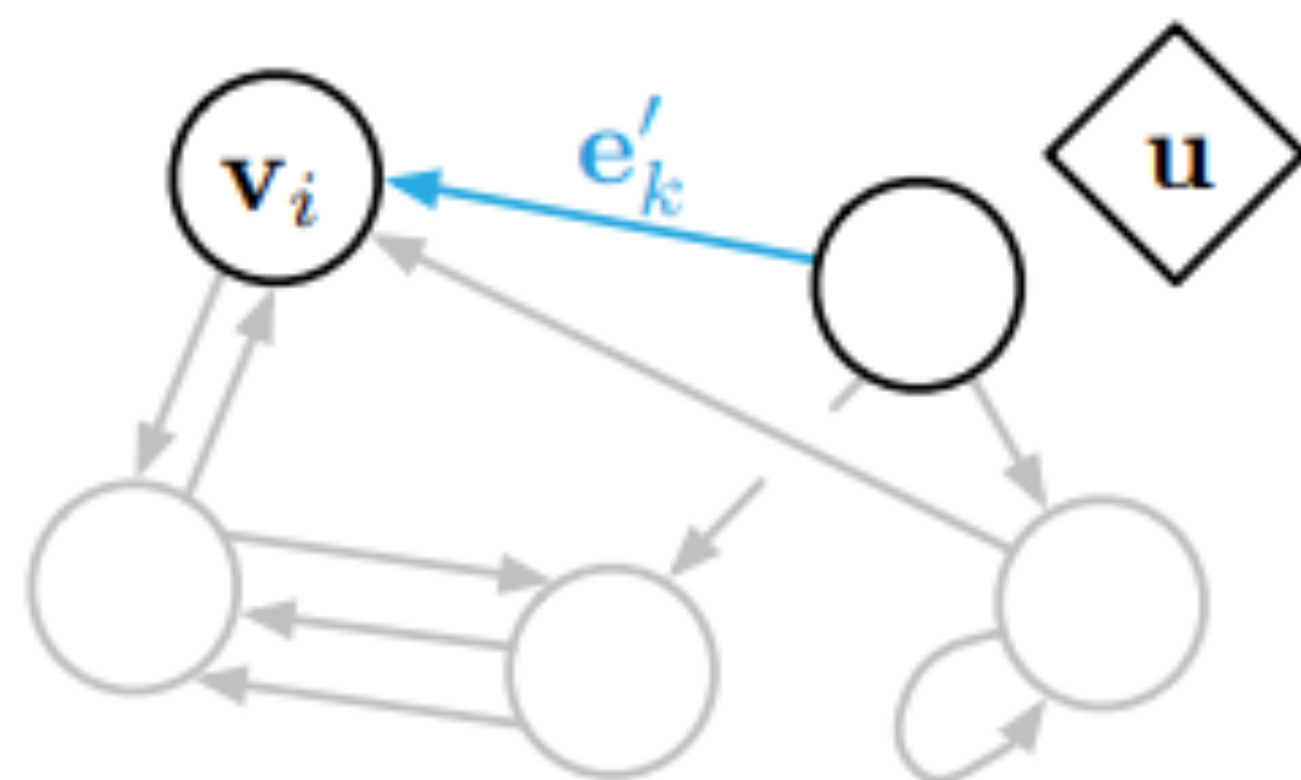
(c) Global update

Update globals
using (new) nodes
and (new) edges

The message passing algorithm

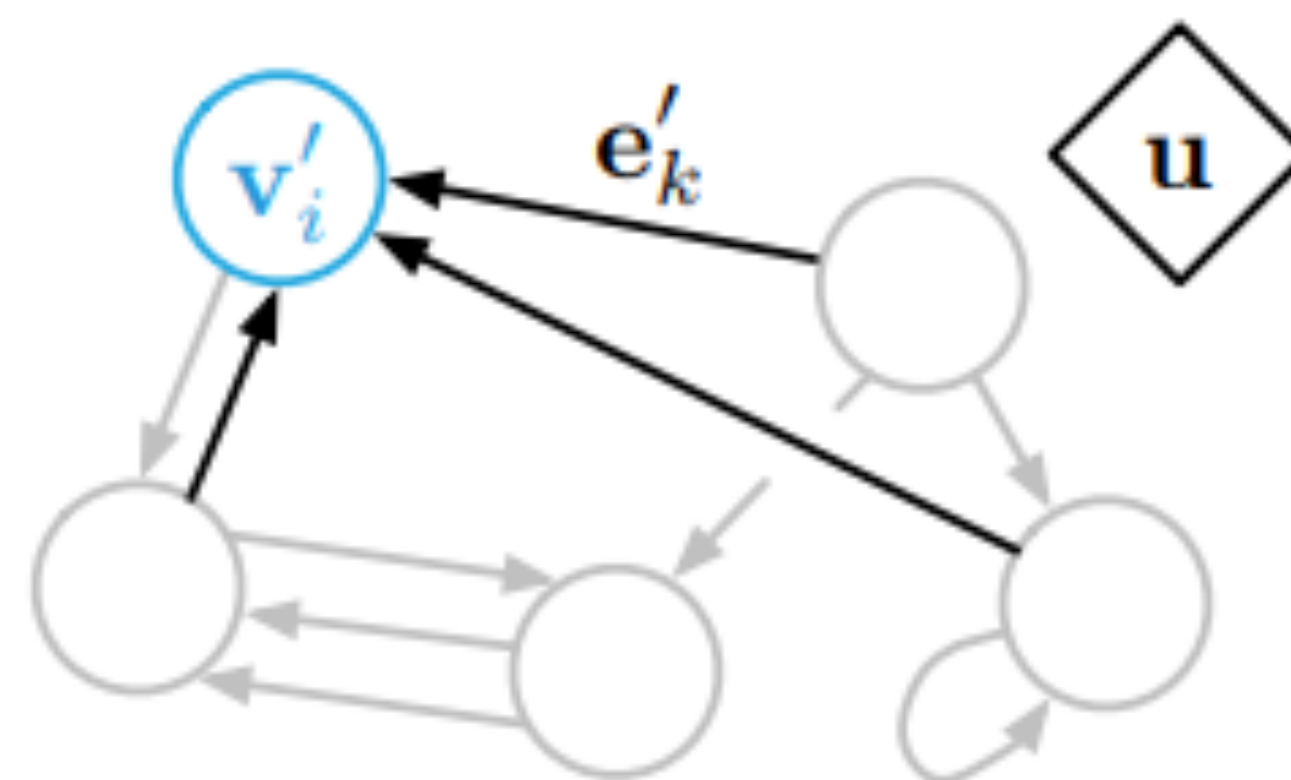
- Nodes, edges, and globals are updated through aggregation and MLPs

BLUE updated by BLACK (not utilizing GREY)



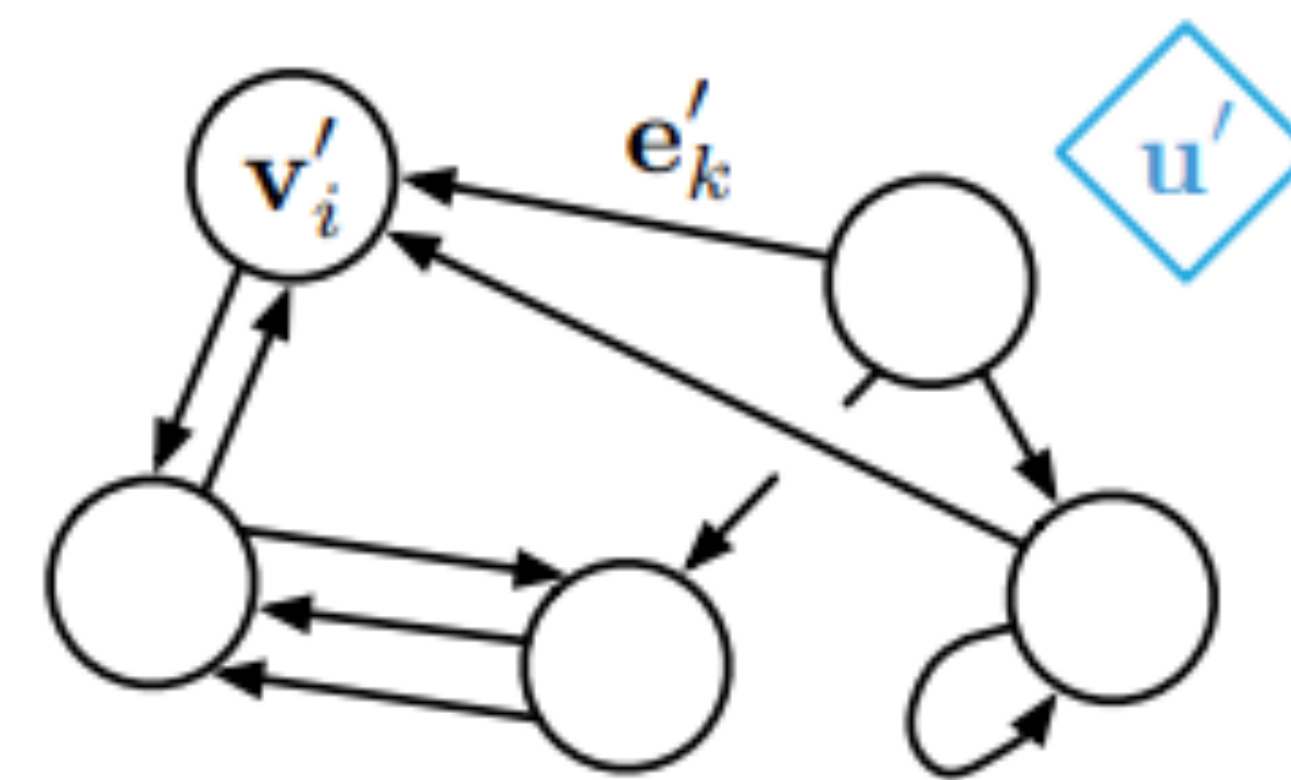
(a) Edge update

Update edges using
connected nodes and globals



(b) Node update

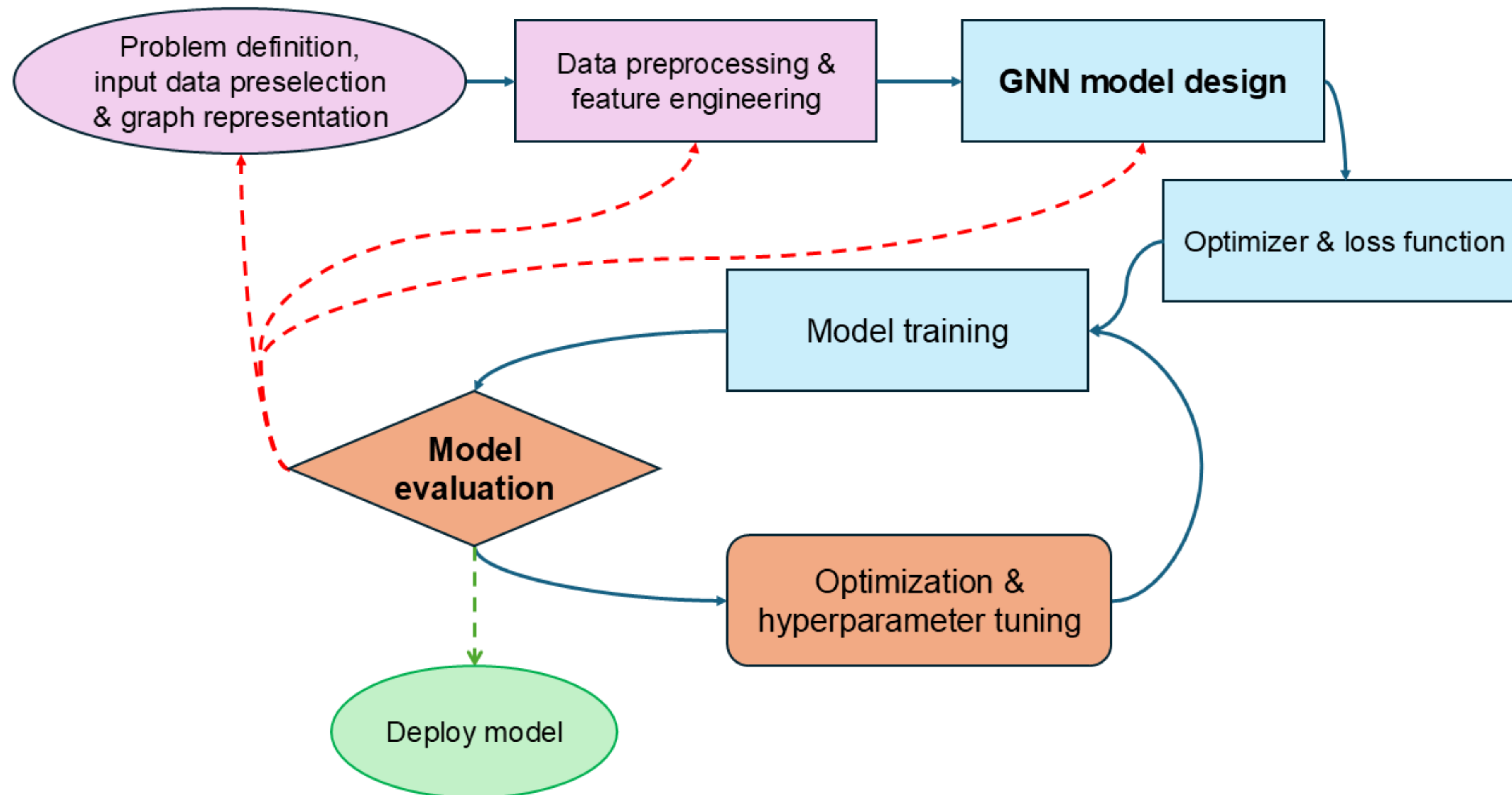
Update nodes
using (new) edges
and globals



(c) Global update

Update globals
using (new) nodes
and (new) edges

This process can be repeated many times
Each iteration can have a unique set of NNs



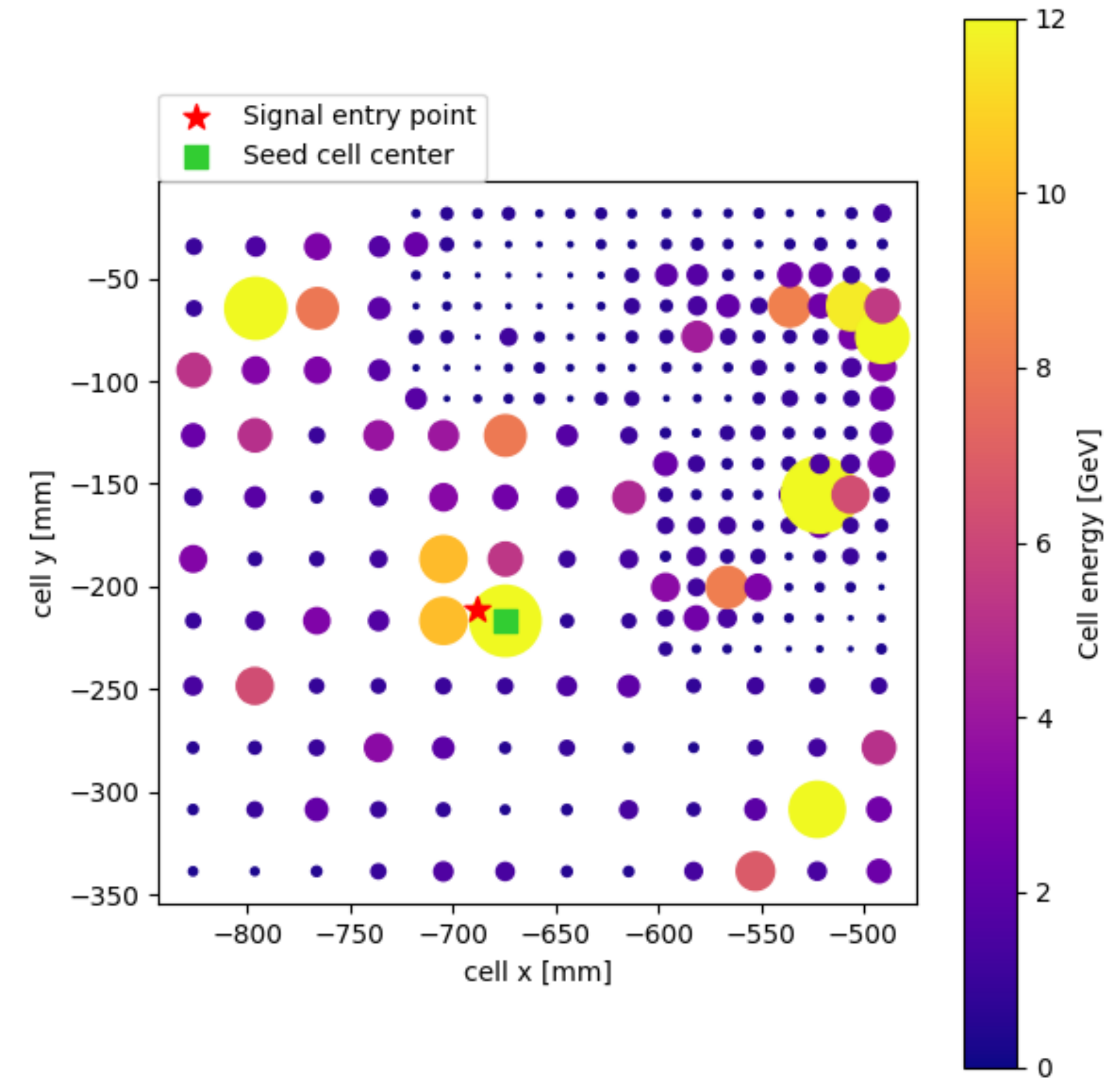
Our approach

📌 Training sample

- ▶ Single photons in the particle-collision conditions expected for LHCb Run 5
- ▶ $0.5 \leq E_T \leq 5$ GeV photons

📌 Preselection

- ▶ Photons produced within 100 mm of interaction point
- ▶ Clusters with seed in **the central innermost region of the ECAL**
- ▶ 5x5 modules around the seed module, reduced to 25 central cells



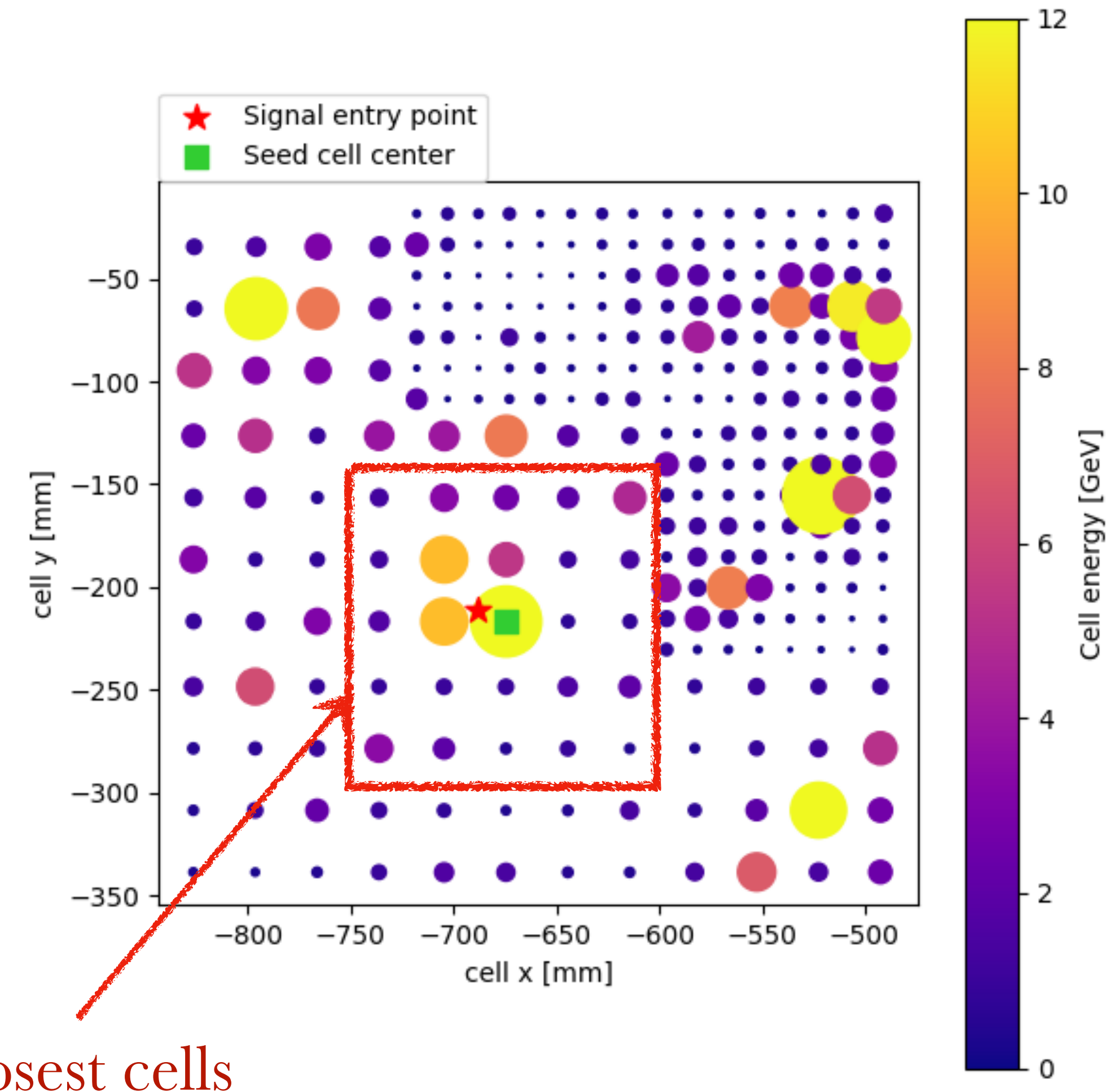
Our approach

📌 Training sample

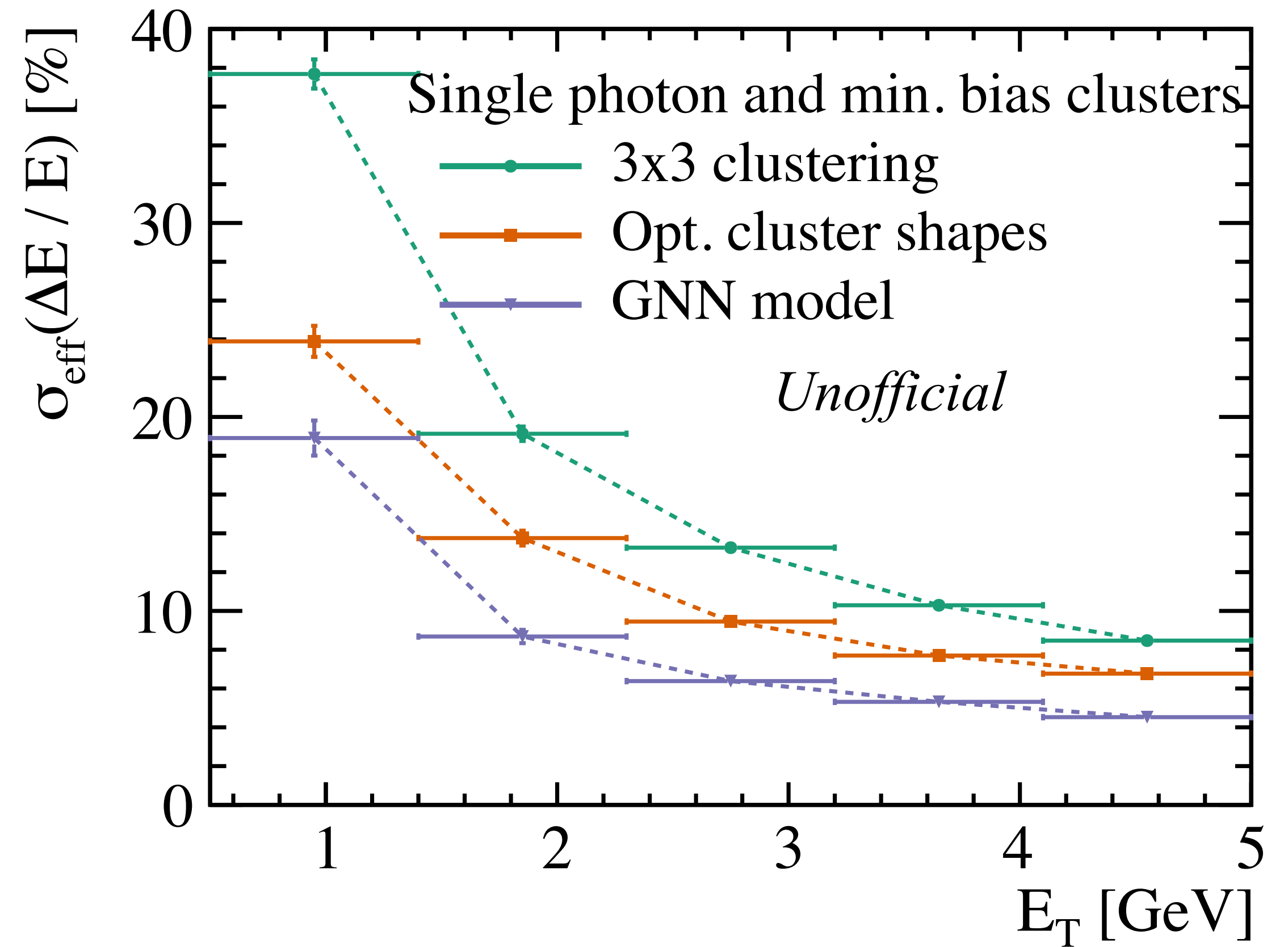
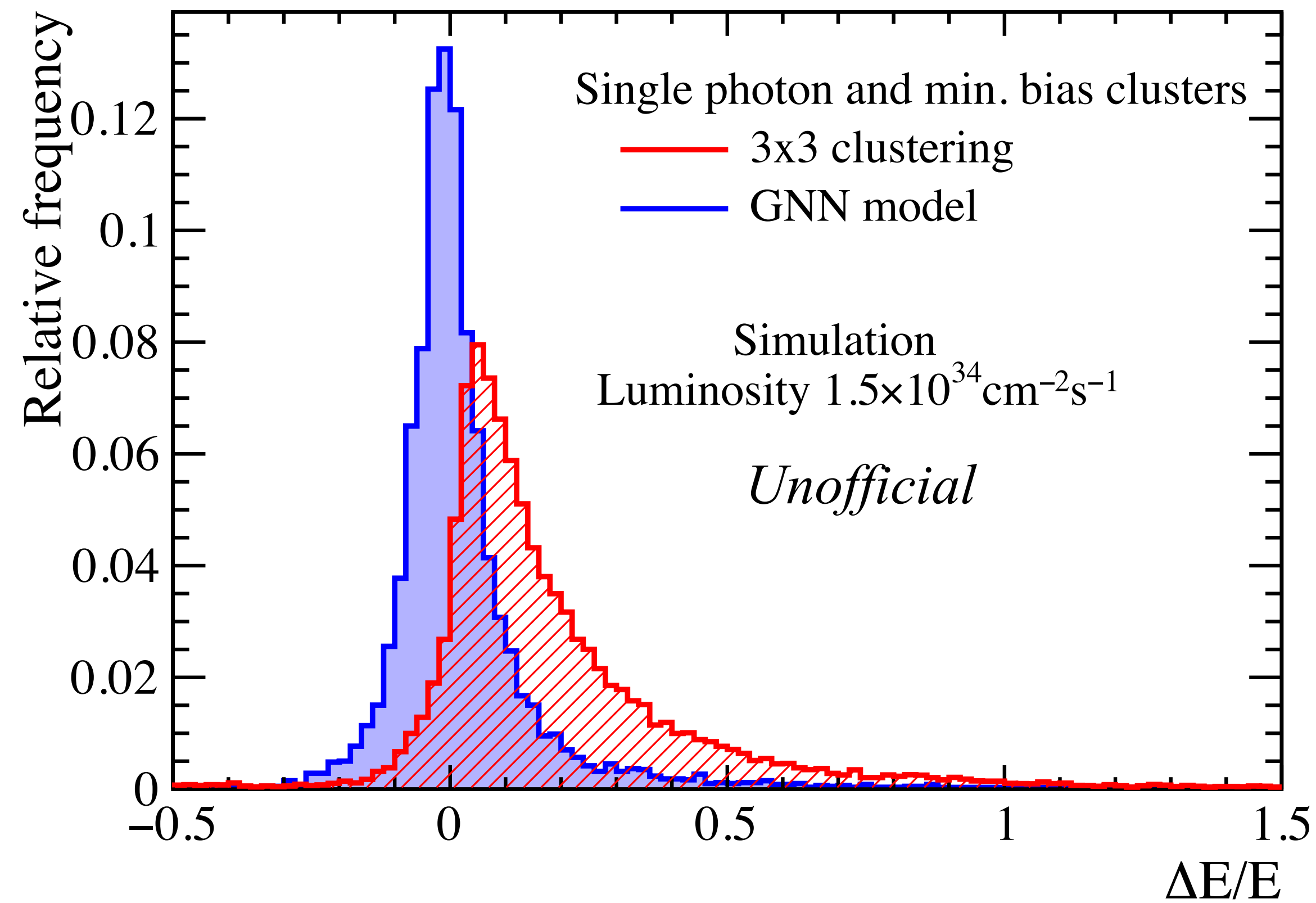
- ▶ Single photons in the particle-collision conditions expected for LHCb Run 5
- ▶ $0.5 \leq E_T \leq 5$ GeV photons

📌 Preselection

- ▶ Photons produced within 100 mm of interaction point
- ▶ Clusters with seed in **the central innermost region of the ECAL**
- ▶ 5x5 modules around the seed module, reduced to 25 central cells

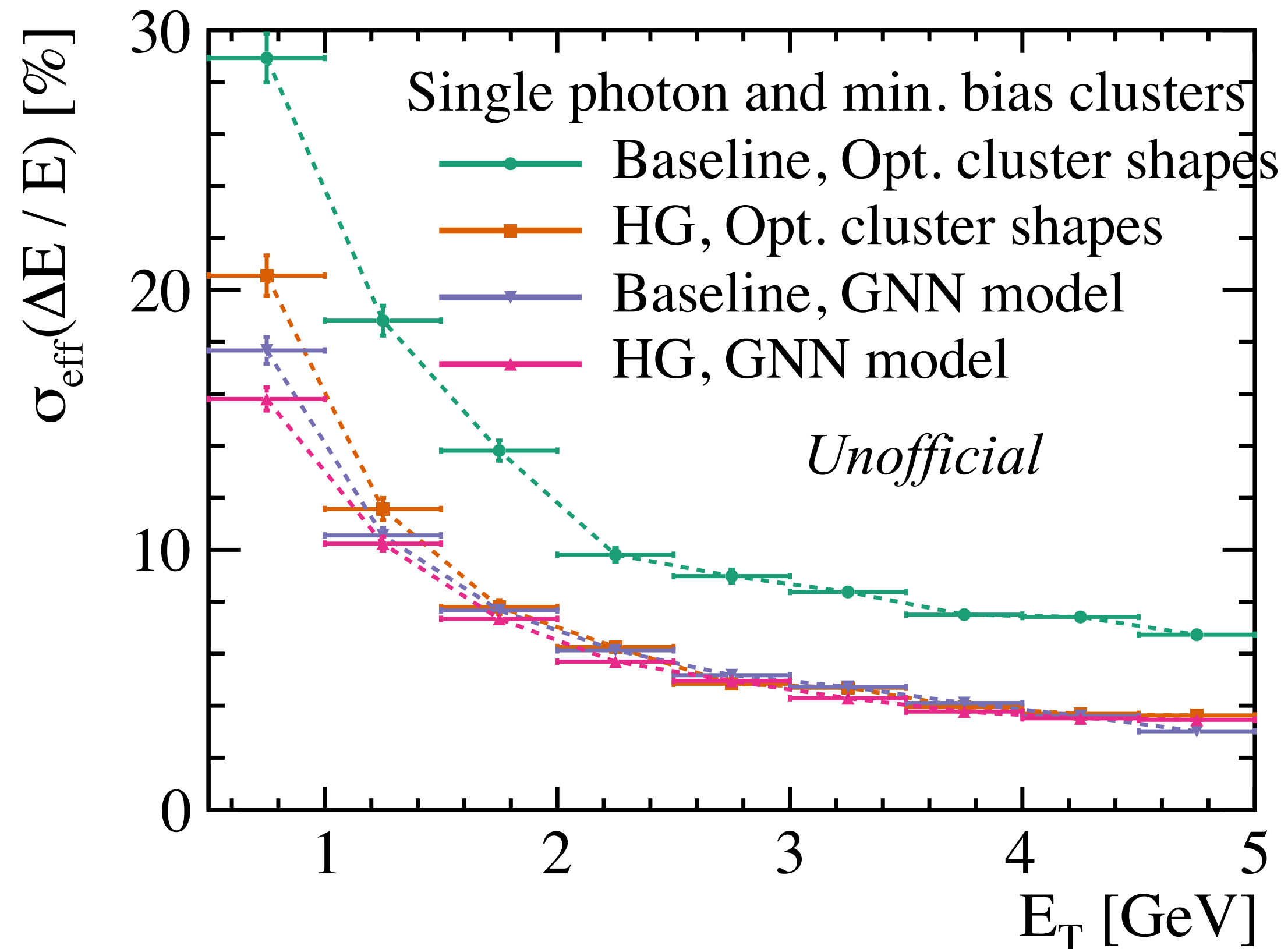


25 closest cells



The GNN outperforms the standard approaches over the full E_T range

Higher-granularity alternative



- 📌 The GNN achieves better performance than the standard approach even with lower granularity
- 📌 Not much improvement observed in the GNN with higher transverse granularity configuration
- 📌 But it further demonstrates that the GNN can handle irregular geometries
- 📌 Further improvements might be achievable with geometry-aware features

Conclusions



📌 We successfully developed novel GNN architectures for PicoCal reconstruction

▶ Challenging yet highly promising development

▶ The approach already outperforms standard reconstruction methods

▶ Further improvements expected as new models, features and strategies are explored

▶ Active ongoing research



📌 Flexible PyTorch implementation

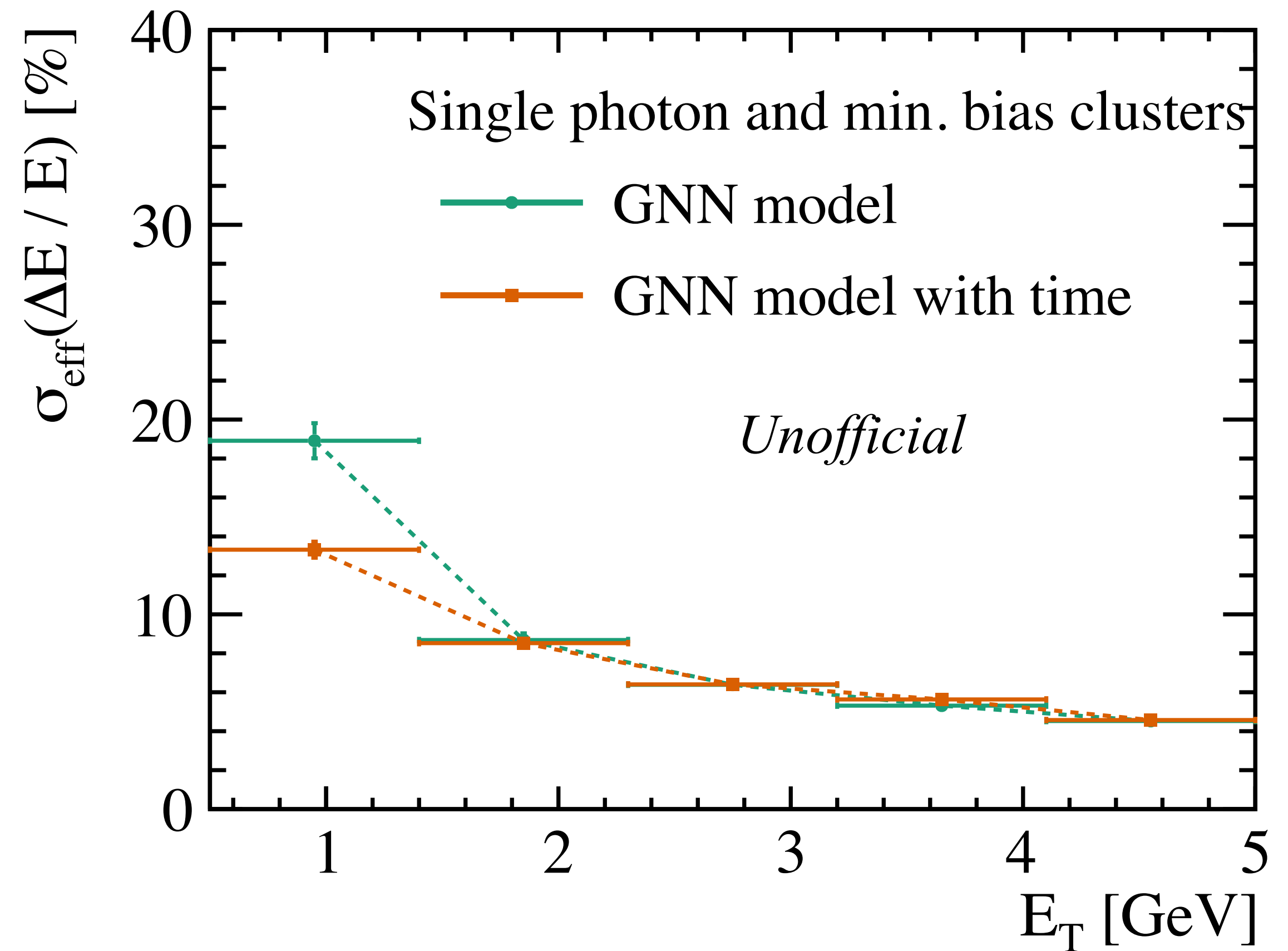
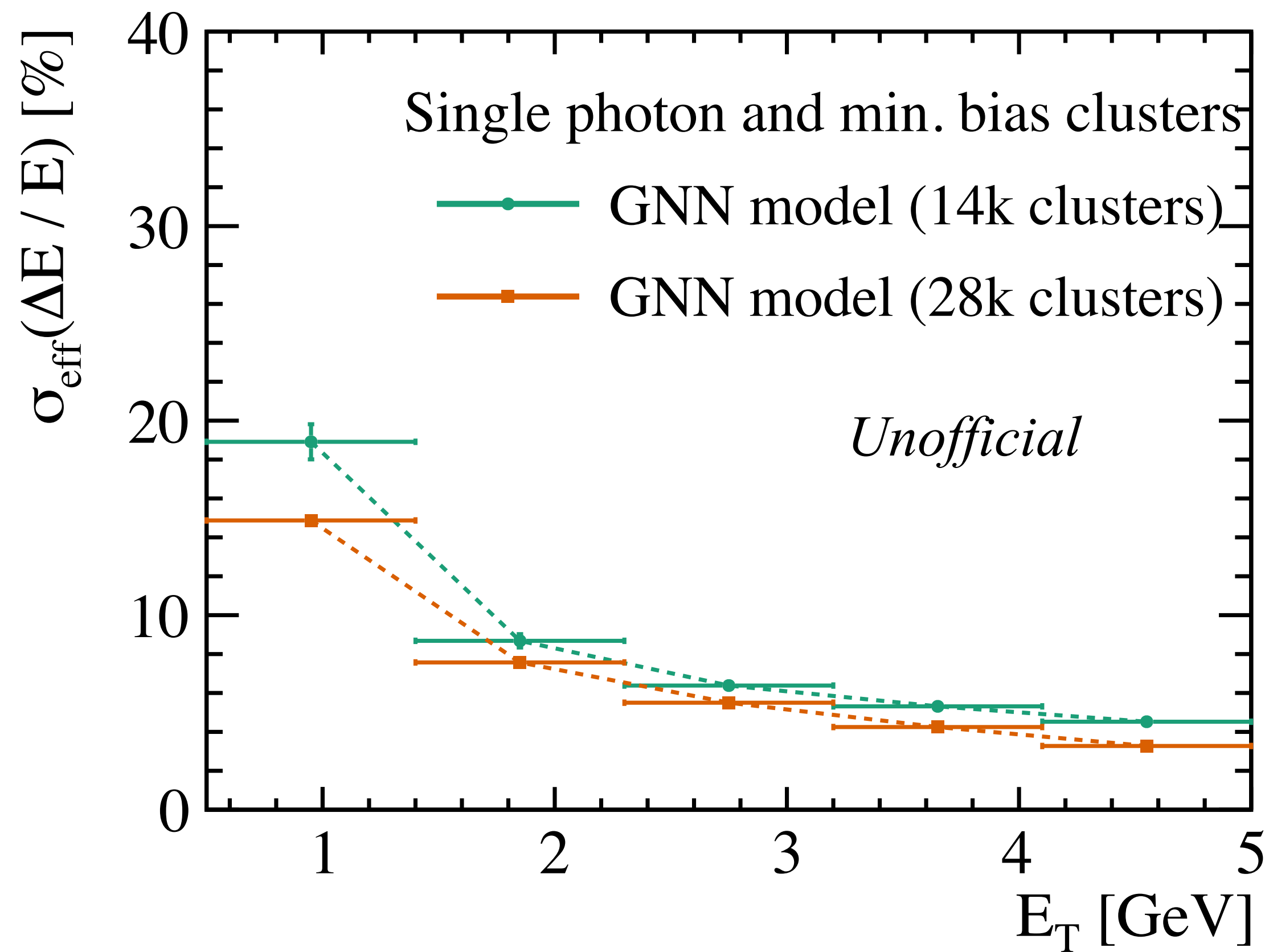
▶ Currently being tested for LHCb Runs 3&4



UNIVERSITAT DE
BARCELONA

▣ Further development required for real-time deployment — [Check Uzziel's talk!](#)

Further improvements

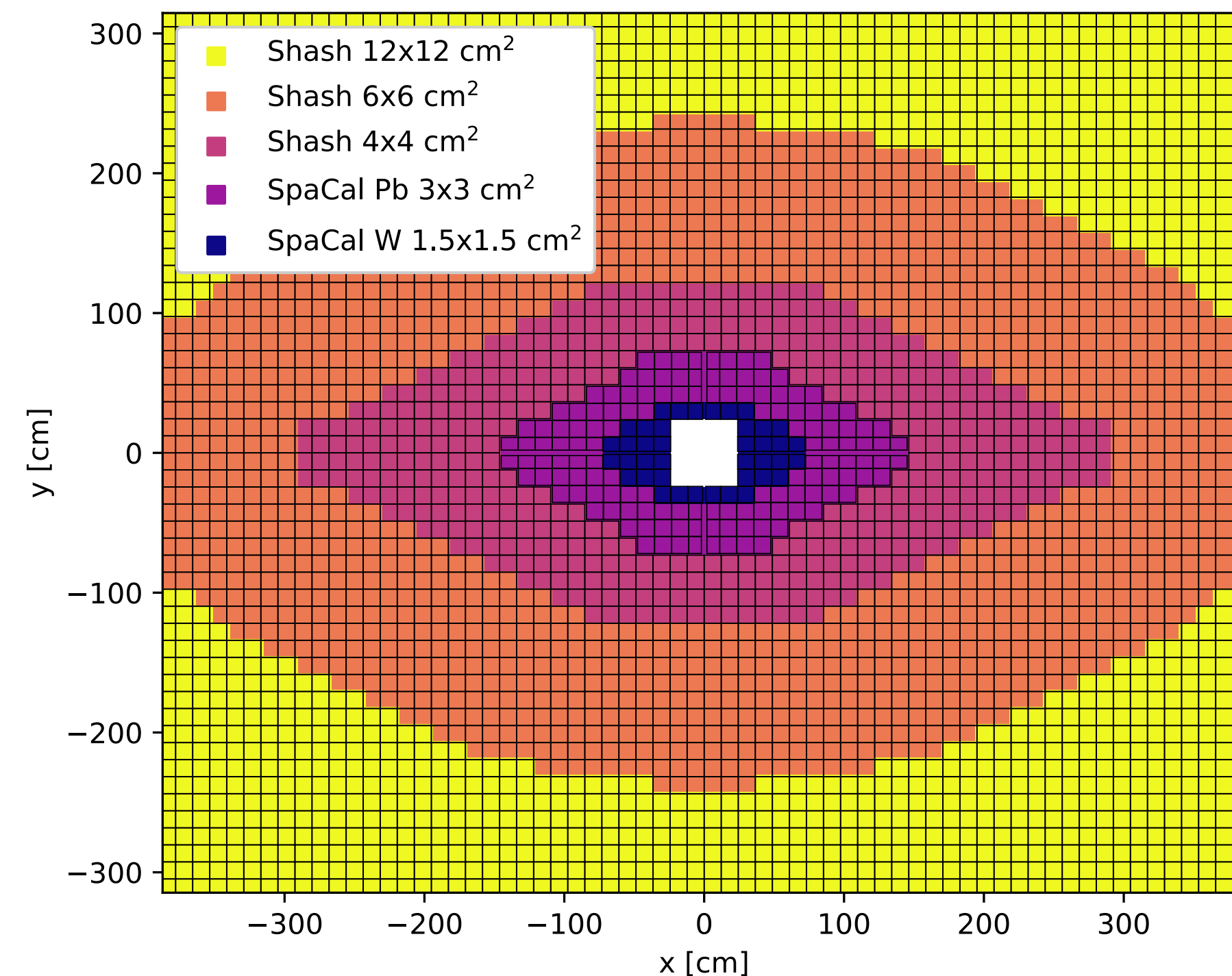


The GNN performance can still be improved with more data, and timing information can also help

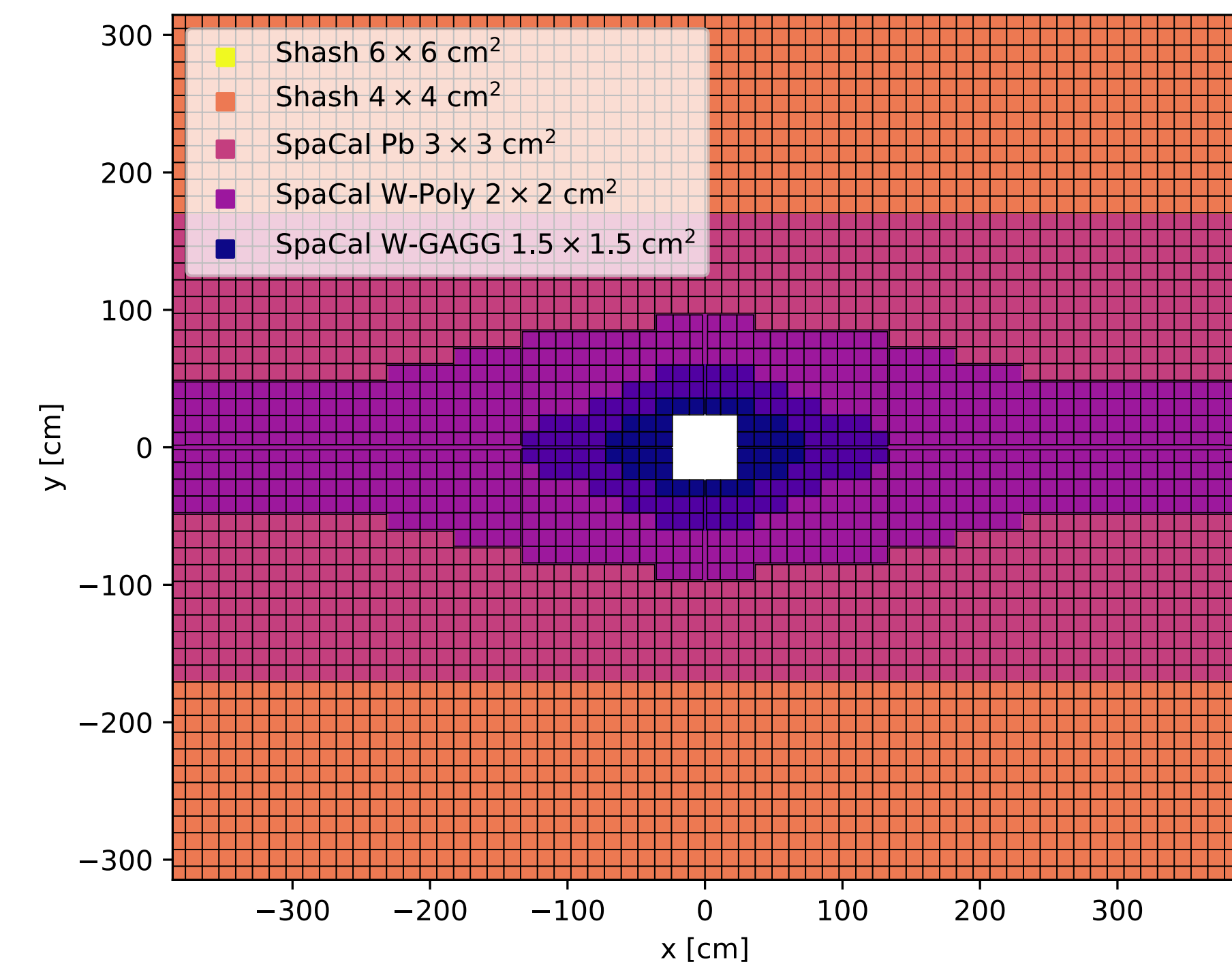
High-granularity single-sided configuration

📌 Designed to keep # of channels $\sim$ constant

▶ Reduce overall occupancy



Baseline



HG

Our approach

Nodes

- ▶ Deposited energy (front, back)
- ▶ Cell index in module
- ▶ Cell position (x, y)
- ▶ Distance to seed cell

Edges

- ▶ ΔE , Δx , Δy , total distance

Globals

- ▶ Total energy
- ▶ Seed cell position (x, y)

Loss

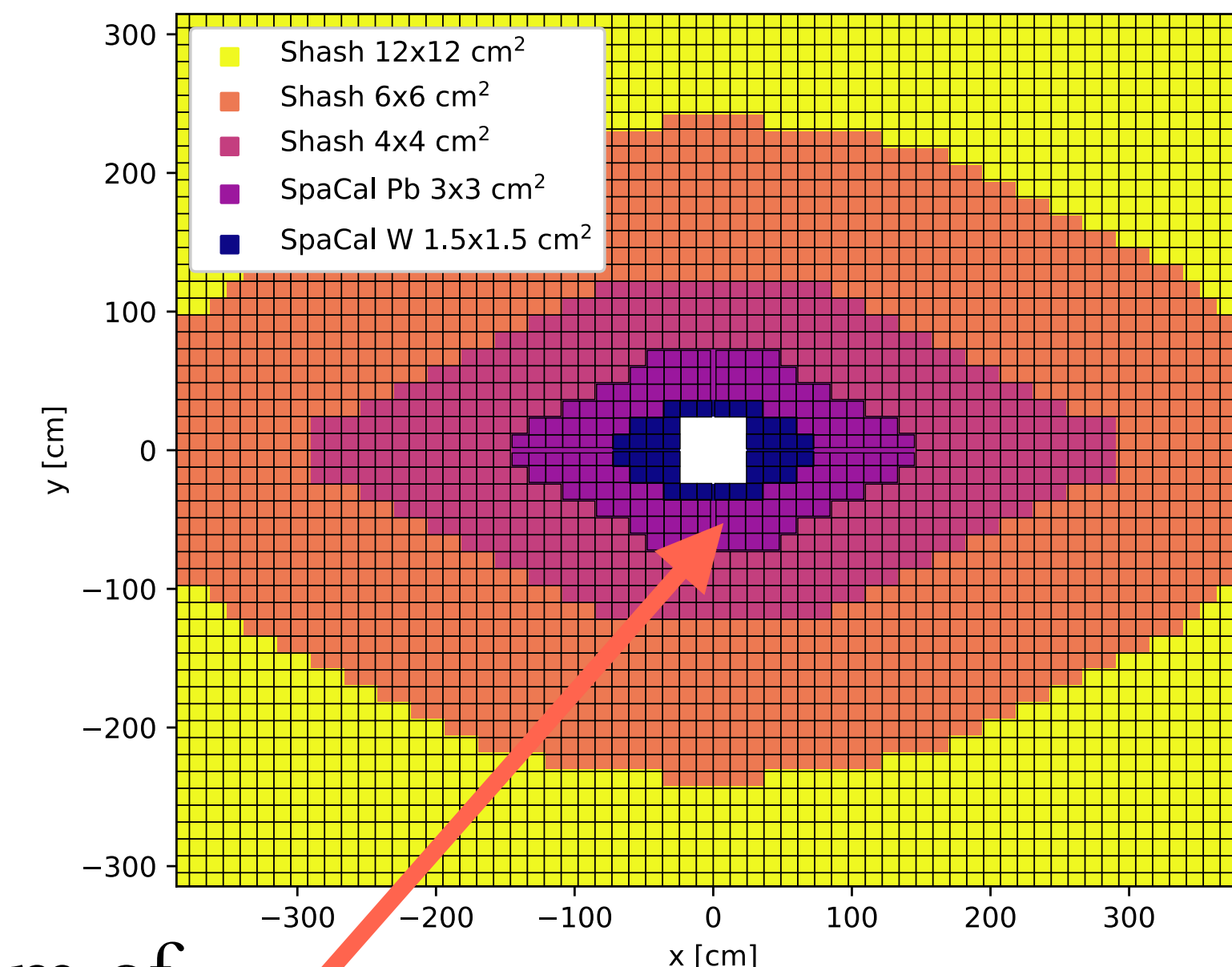
- ▶ MSE

Target

- ▶ True energy

Preselection

- ▶ Photons produced within 100 mm of interaction point
- ▶ Clusters with seed in **SpaCal-Pb** region
- ▶ 3x3 modules around the seed module, reduced to 25 central cells



How to evaluate the performance?

Asymmetric distributions due to pile-up: long right tails

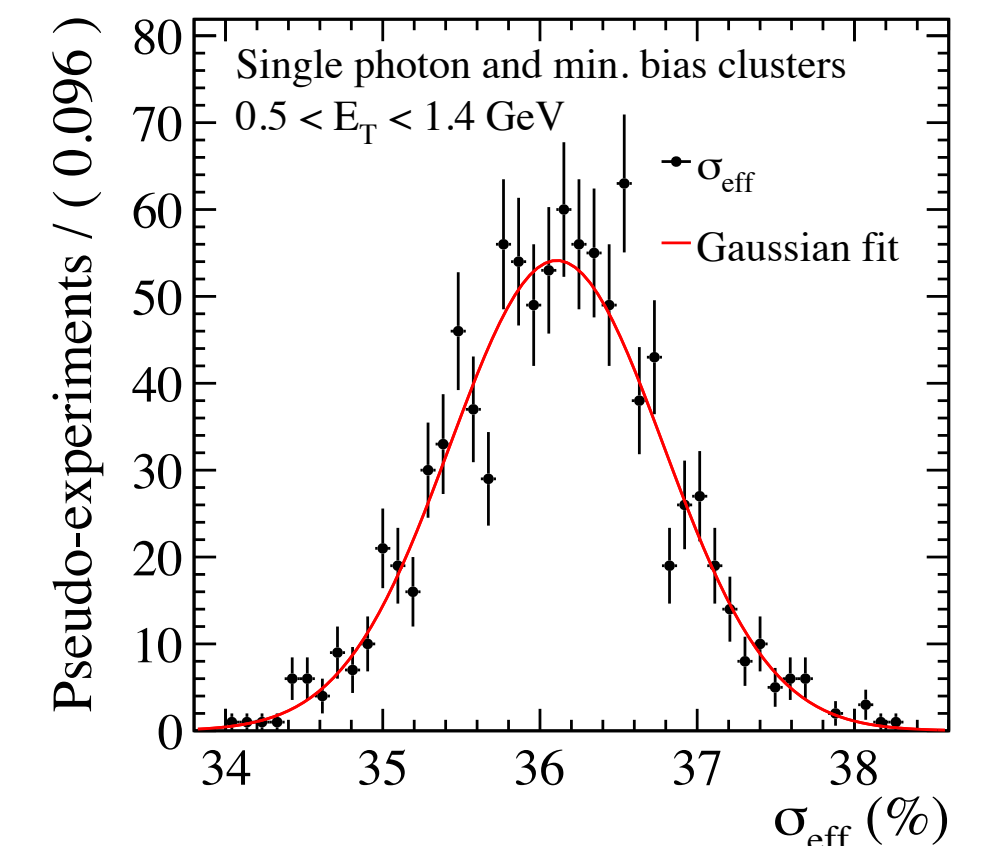
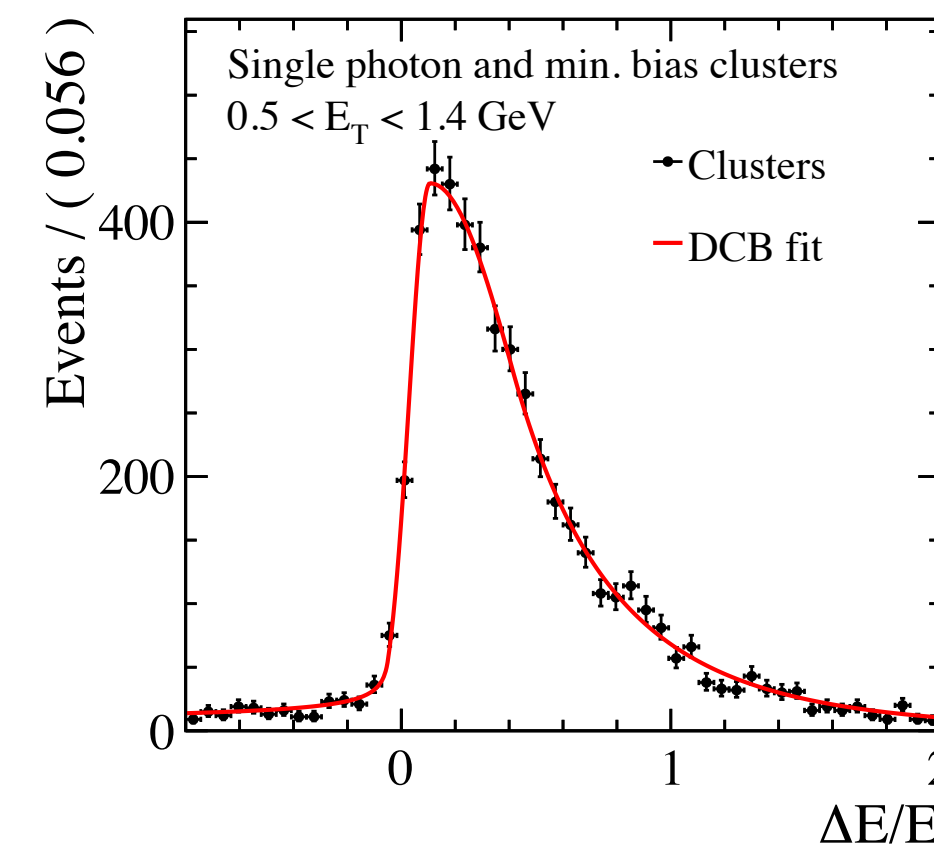
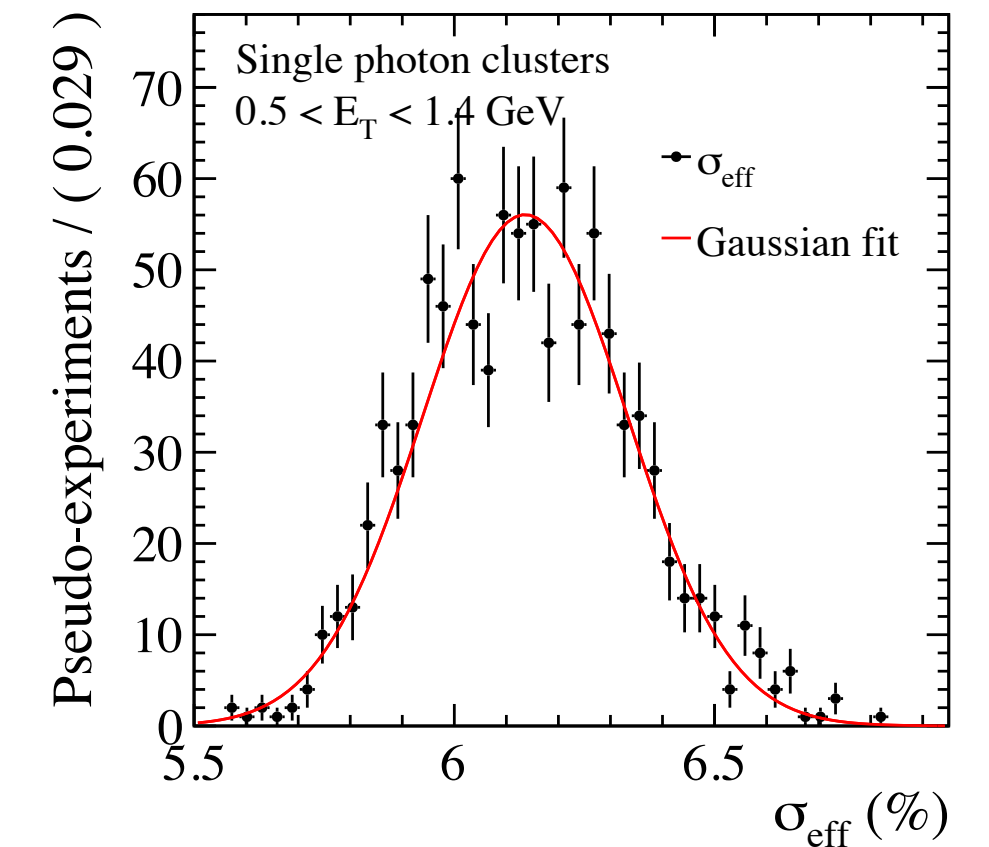
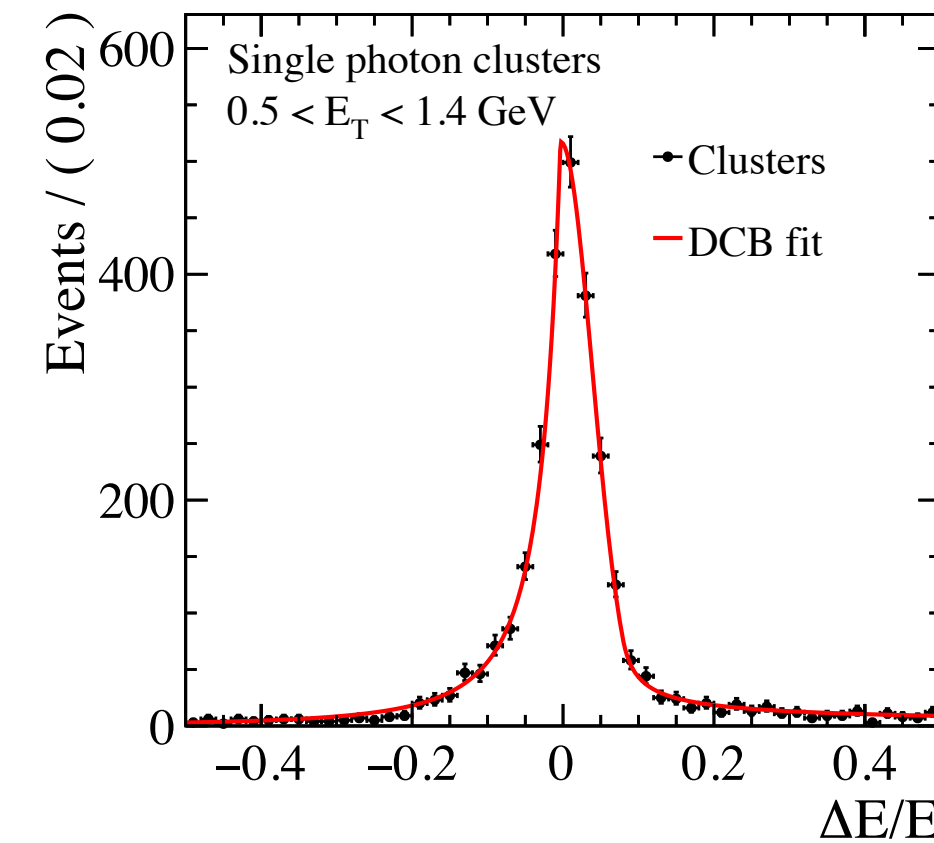
► Defining the ‘resolution’ is not straightforward

► We define the resolution (σ_{eff}) as the half-width of the central 68% interval

Fits to the $\Delta E/E$ distributions

► Generate toys from the p.d.f., recompute metric

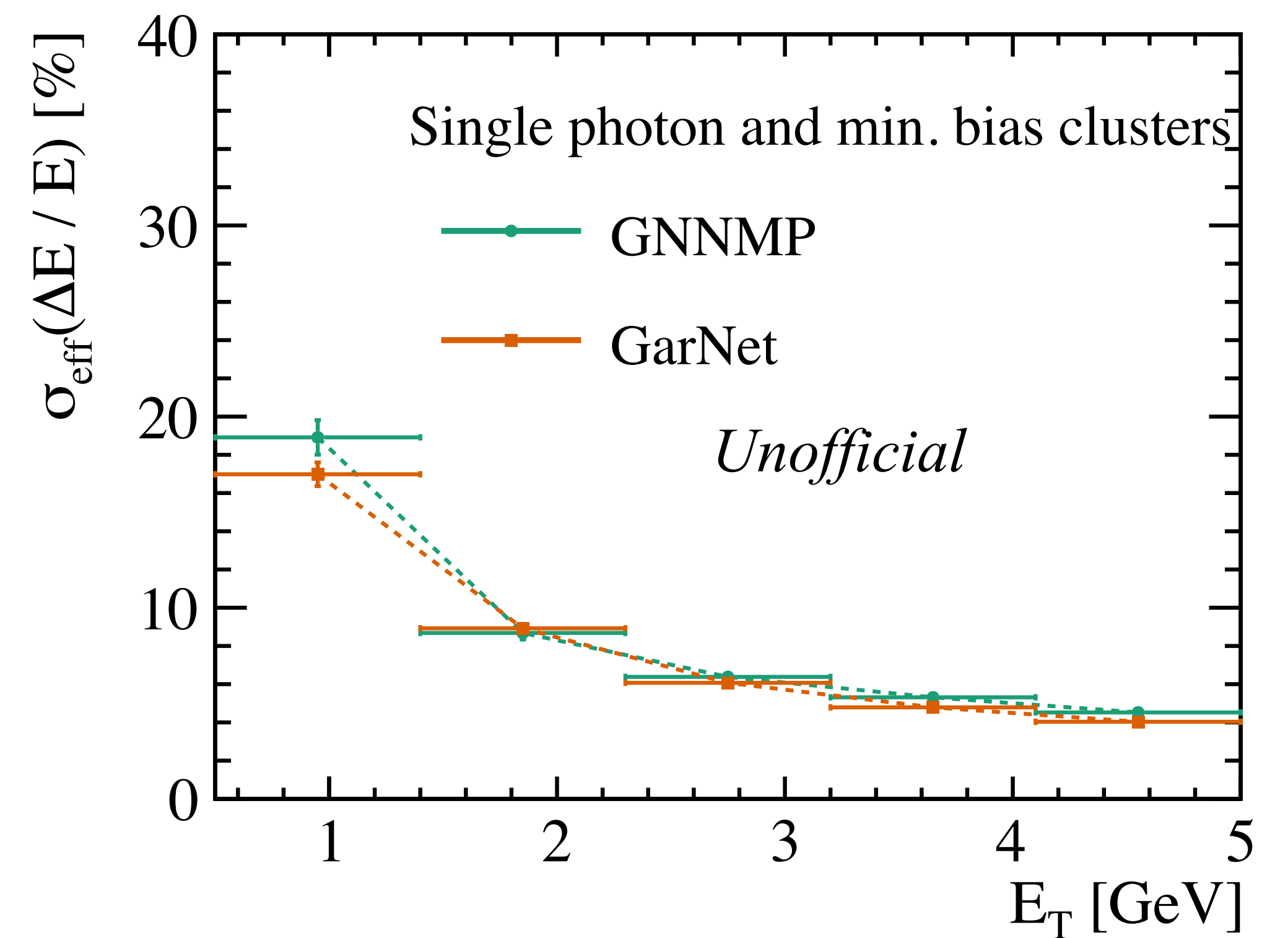
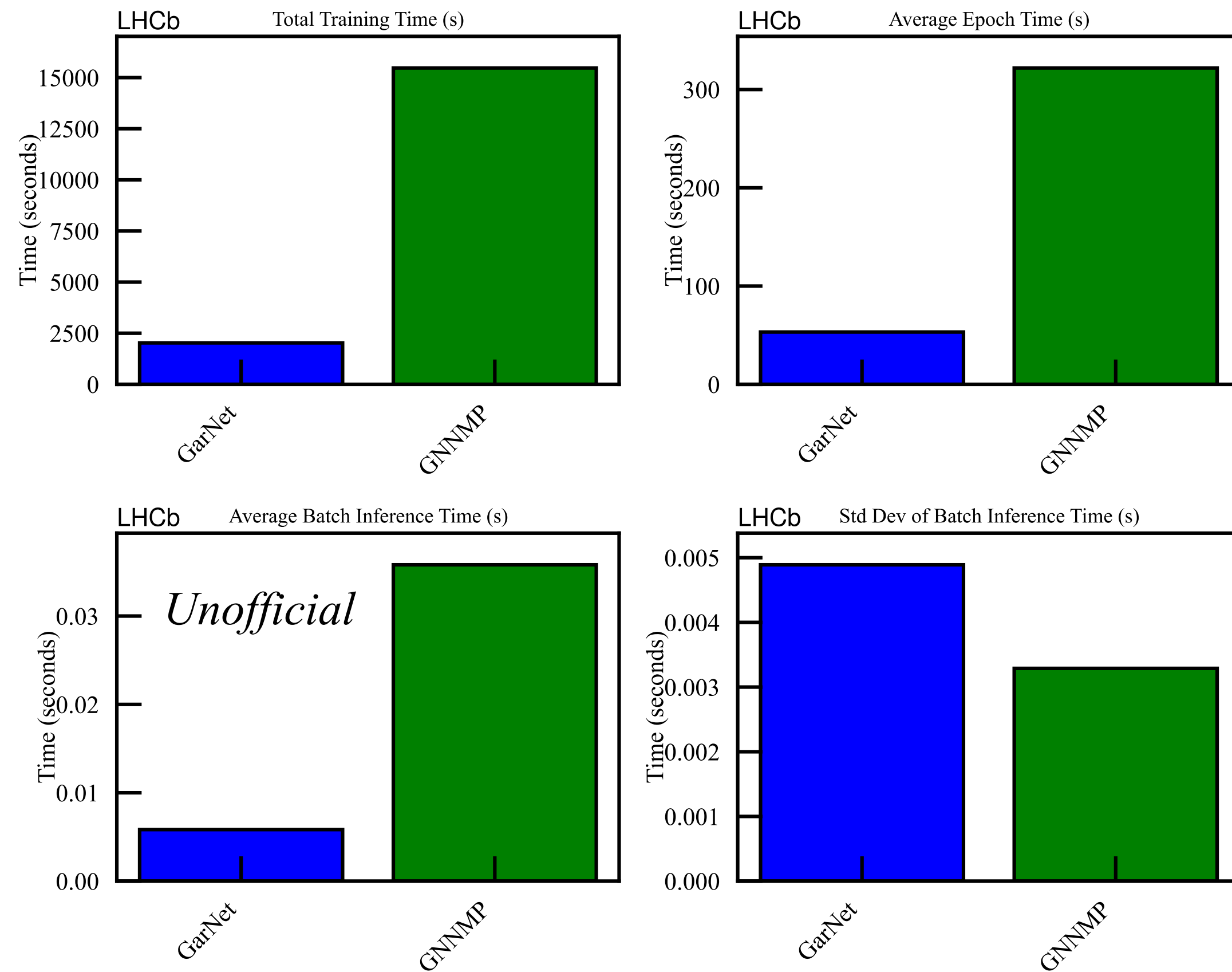
► Gaussian fit to distribution of metric to extract mean and width (uncertainty)



- 📌 A lightweight, attention-enhanced variant GNN architecture
 - ▶ Introduced for real-time particle reconstruction at CMS by Iiyama et al. ([Eur. Phys. J. C 79, 608 \(2019\)](#); [Front. Big Data 3, 598927 \(2021\)](#))
 - ▶ Explicit edges are replaced by learnable aggregators connecting nodes through latent vertices
 - This reduces the number of mathematical operations, decreasing training and inference time
- ☑ GarNet has a known ONNX implementation for FPGA deployment

GarNet results

 GarNet has similar performance, but is 6-7x faster!



Ongoing studies

- 📌 Comparison for performance in and out of boundary regions
- 📌 Alternative strategies for connecting nodes in the full GNNMP
- 📌 Spatial symmetries
- 📌 Node-level targets
- 📌 Curriculum learning
- 📌 GarNet:
 - ▶ Distillation, quantization-aware training, post quantization
 - ▶ FPGA implementation, GPU parallelization

Multitarget regression

Simultaneous regression of energy, time, and position

- ▶ Each target contribution must be properly weighted in the loss
- ▶ Uncertainty-aware losses can help the model dynamically balance target contributions
- However, regularization may be needed to prevent degenerate minima and stabilize training
- ▶ Strategies range from simple weighted MSE to multivariate Gaussian NLL with full covariance and Cholesky decomposition

Multi-target MSE (predefined weights)

$$\mathcal{L} = \sum_i w_i (y_i - \hat{y}_i(x))^2$$

learn one uncertainty per target

Homoscedastic (learned)

$$\mathcal{L} = \sum_i \left(\frac{1}{2\hat{\sigma}_i^2} (y_i - \hat{y}_i(x))^2 + \frac{1}{2} \log \hat{\sigma}_i^2 \right)$$

make uncertainty input-dependent

Heteroscedastic (diagonal)

$$\mathcal{L} = \sum_i \left(\frac{1}{2\hat{\sigma}_i^2(x)} (y_i - \hat{y}_i(x))^2 + \frac{1}{2} \log \hat{\sigma}_i^2(x) \right)$$

include target correlations

Multivariate Gaussian NLL

$$\mathcal{L} = \frac{1}{2} (\mathbf{y} - \hat{\mathbf{y}}(x))^T \hat{\mathbf{\Sigma}}(x)^{-1} (\mathbf{y} - \hat{\mathbf{y}}(x)) + \frac{1}{2} \log \det \hat{\mathbf{\Sigma}}(x)$$