LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Smart Background Simulation with Graph Attention Networks and Weighting Methods

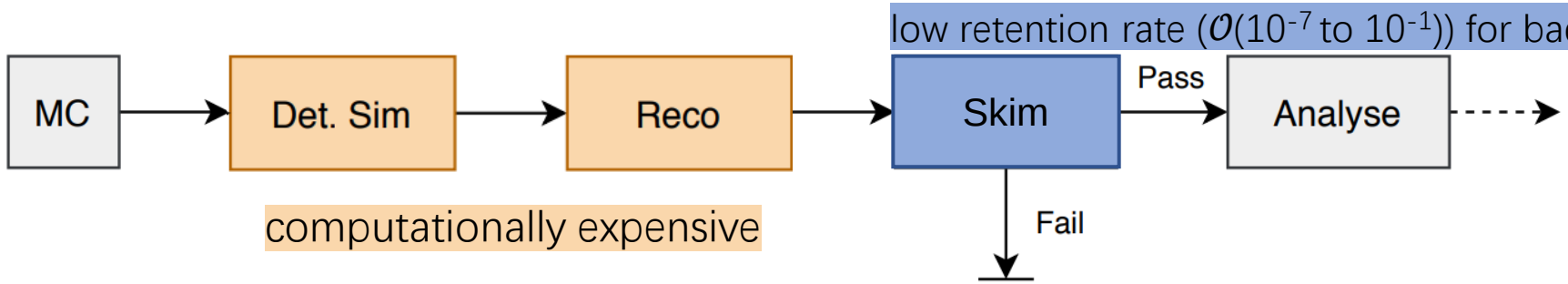
Boyang Yu, Nikolai Hartmann, Luca Schinnerl, Thomas Kuhr

Joint Seminar of Particle Physics Groups, July 13th, 2022

Belle II

Contents

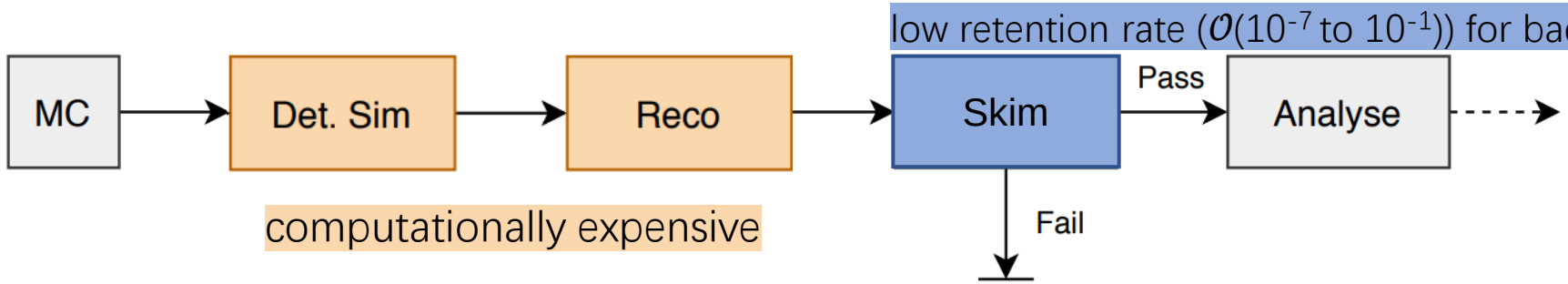
Current Monte Carlo Simulation data flow



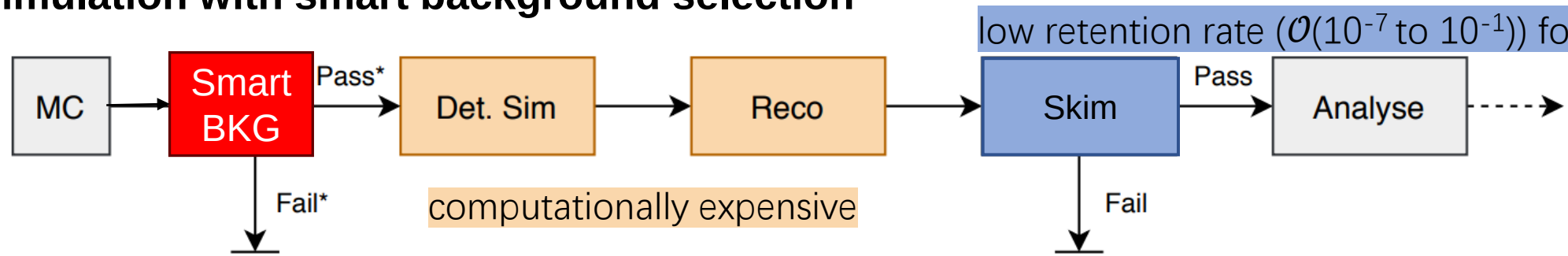
Smart Background Simulation with Graph Attention Networks and Weighting Methods

Motivation

Current Monte Carlo Simulation data flow



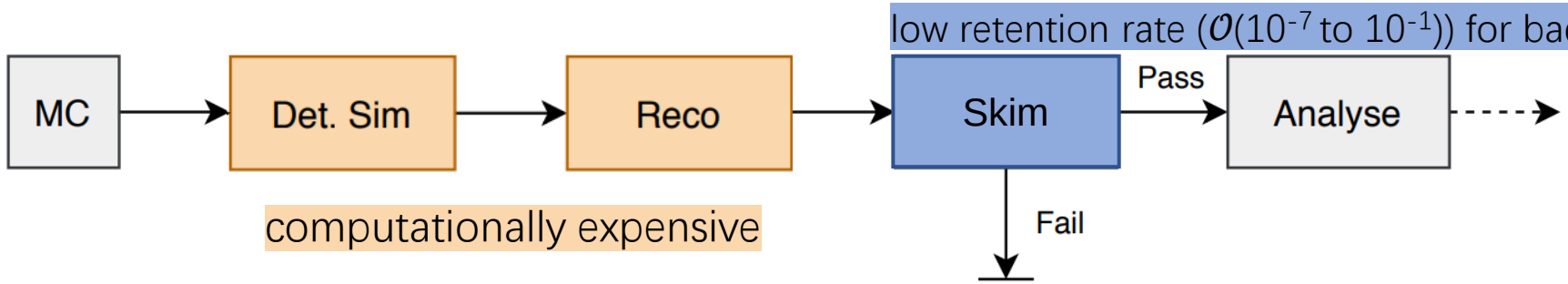
Simulation with smart background selection



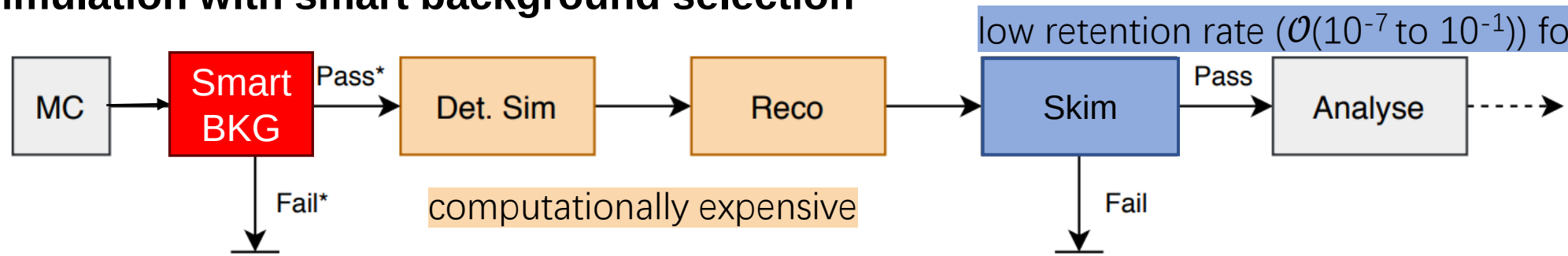
Smart Background Simulation with Graph Attention Networks and Weighting Methods

Motivation

Current Monte Carlo Simulation data flow



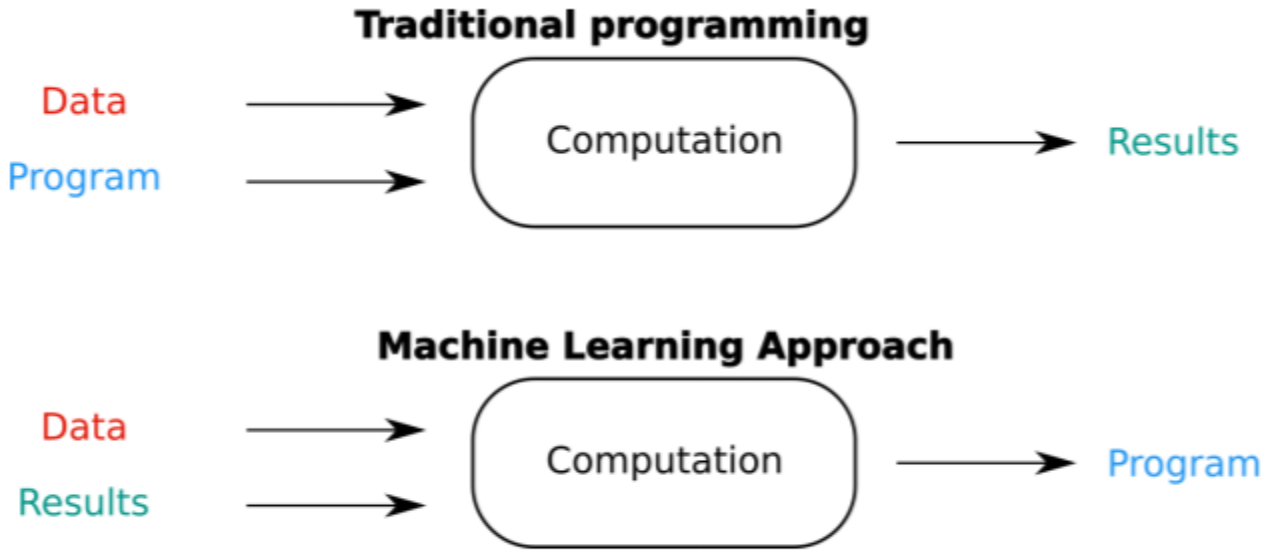
Simulation with smart background selection



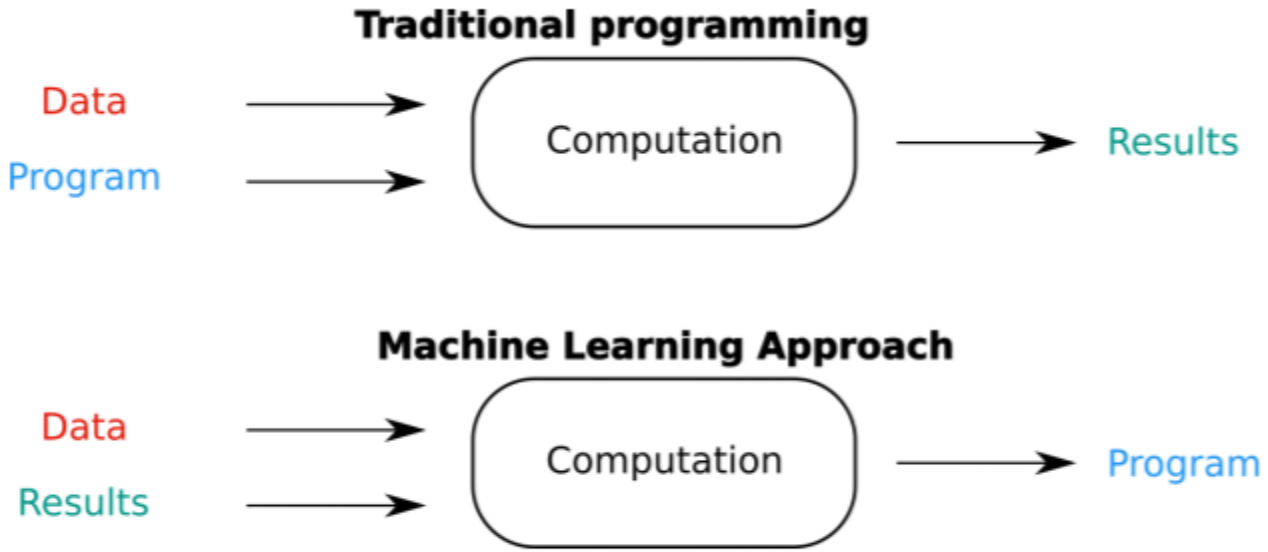
Previous Works:

- **PhD Thesis**: *Hadronic Tag Sensitivity Study of $B \to K^{(*)}\nu\bar{\nu}$ and Selective Background Monte Carlo Simulation at Belle II*, James Kahn, 2019
- **Talk**: *Selective background Monte Carlo simulation at Belle II*, James Kahn, CHEP 2019

Why Machine Learning

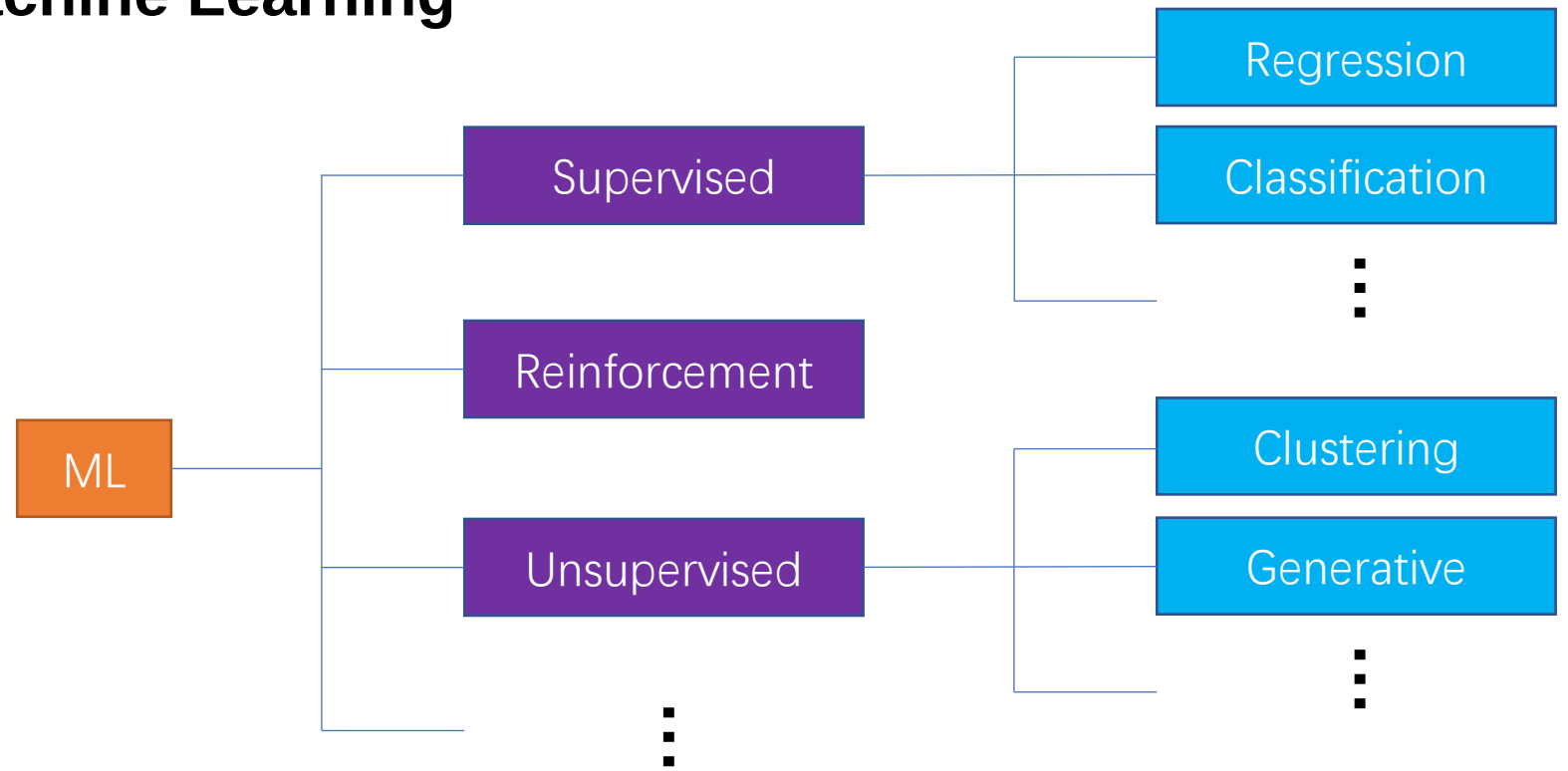


Why Machine Learning

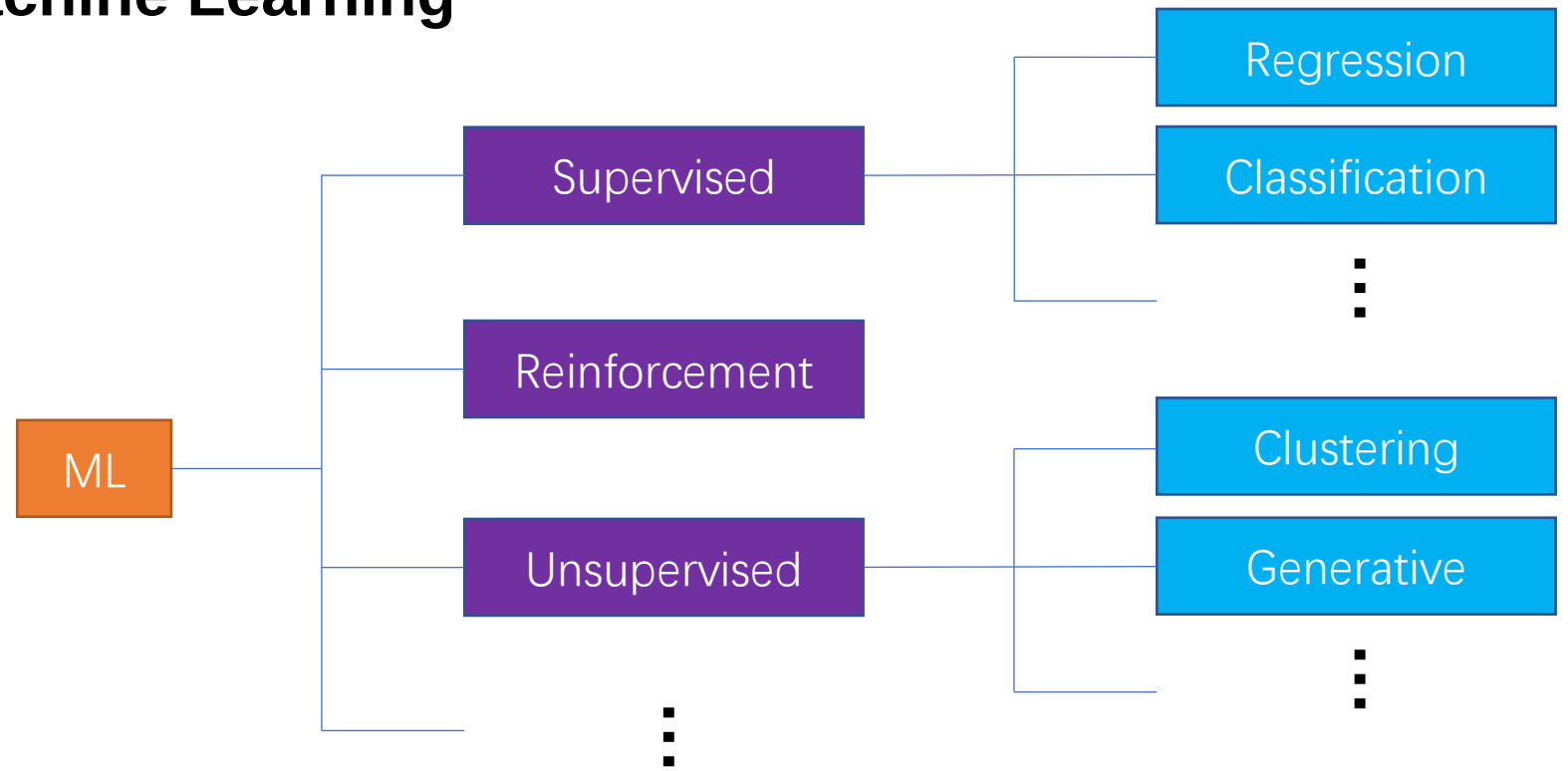


- No need for understanding and describing the detailed mechanisms
  - Easier to achieve
  - Beyond human limits of knowledge and observation
- Simulation with high efficiency
- Large datasets available

Which Machine Learning



Which Machine Learning

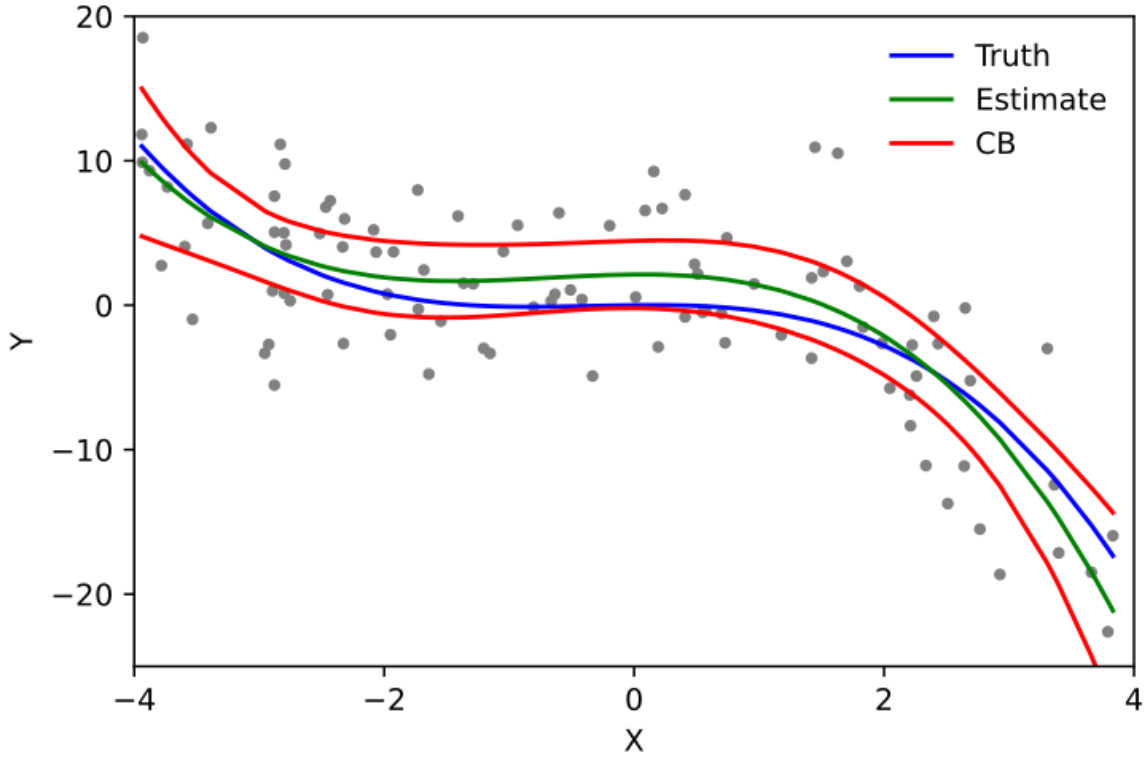


Our Task:

- **In:** Particle decay generated by Monte Carlo
- **Out:** Whether it will pass the skim afterwards

→ Supervised, Classification

Start with Linear Regression

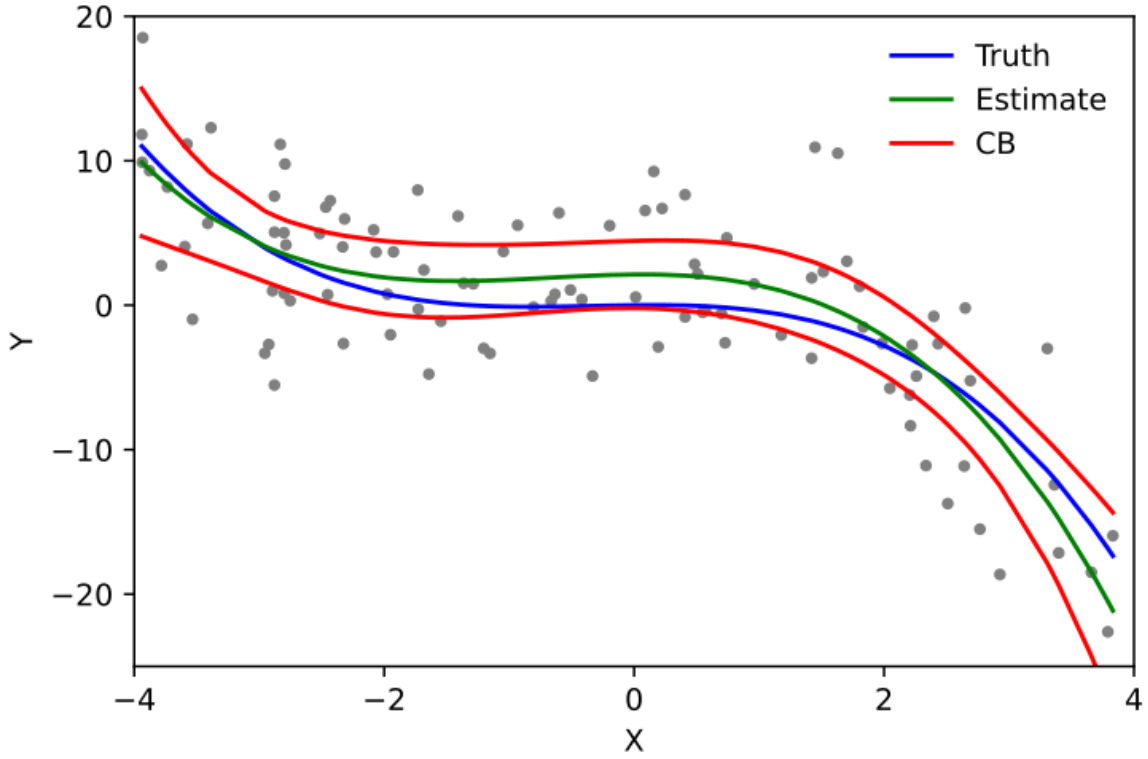


Estimation:

$$\hat{y} \overset{\text{def}}{=} b + w_1 x + w_2 x^2 + w_3 x^3 \overset{\text{def}}{=} \vec{W} \cdot \vec{X} + b = W^i X_i + b$$

with $\vec{X} = \begin{pmatrix} x \\ x^2 \\ x^3 \end{pmatrix}$

Start with Linear Regression



Estimation:

$$\hat{y} \overset{\text{def}}{=} b + w_1 x + w_2 x^2 + w_3 x^3 \overset{\text{def}}{=} \vec{W} \cdot \vec{X} + b = W^i X_i + b$$

with $\vec{X} = \begin{pmatrix} x \\ x^2 \\ x^3 \end{pmatrix}$

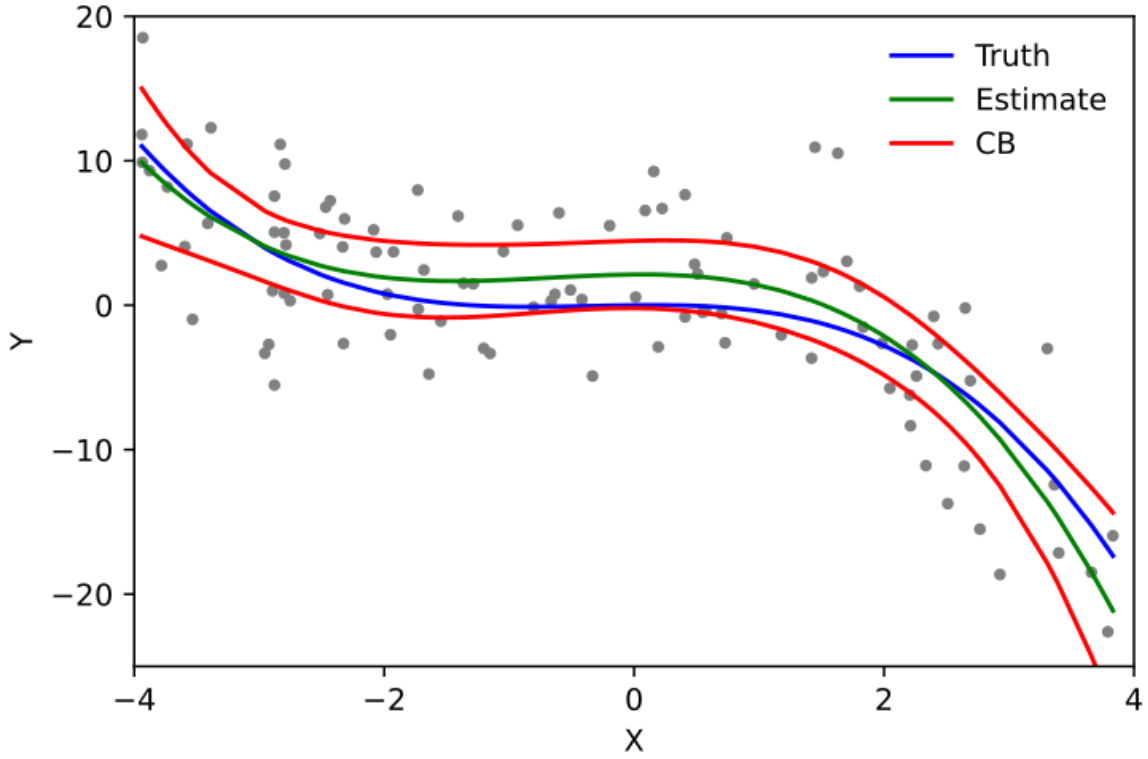
Loss: the difference between estimation and truth

$$L \overset{\text{def}}{=} (\hat{y} - y)^2$$

Cost: the average of losses over all the samples

$$J(W, b) \overset{\text{def}}{=} \frac{1}{2m} \sum_{i=1}^{m} L_i = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}(x_i) - y(x_i))^2$$

Start with Linear Regression



Estimation:

$$\hat{y} \overset{\text{def}}{=} b + w_1 x + w_2 x^2 + w_3 x^3 \overset{\text{def}}{=} \vec{W} \cdot \vec{X} + b = W^i X_i + b$$

with $\vec{X} = \begin{pmatrix} x \\ x^2 \\ x^3 \end{pmatrix}$

Loss: the difference between estimation and truth

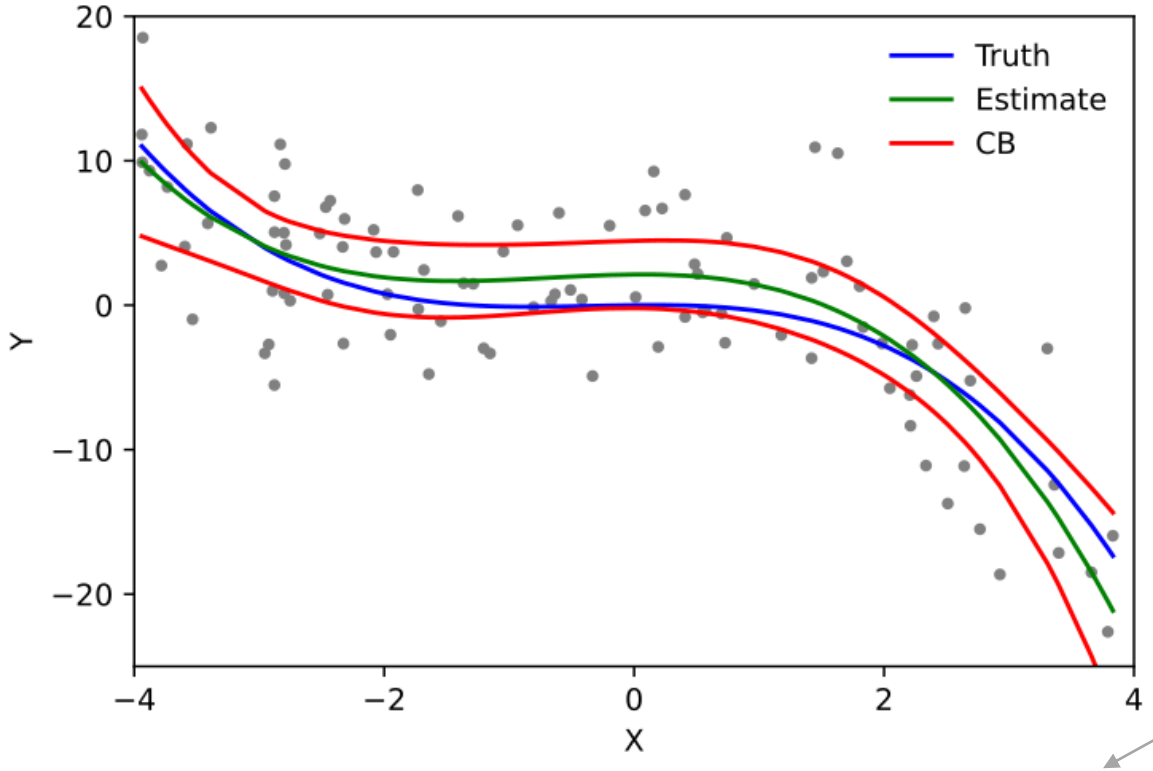
$$L \overset{\text{def}}{=} (\hat{y} - y)^2$$

Cost: the average of losses over all the samples

$$J(W, b) \overset{\text{def}}{=} \frac{1}{2m} \sum_{i=1}^{m} L_i = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}(x_i) - y(x_i))^2$$

Training: find proper parameters $\mathbf{W}, b$ to minimize the cost
-> Best model

Start with Linear Regression



Estimation:

$$\hat{y} \overset{\text{def}}{=} b + w_1 x + w_2 x^2 + w_3 x^3 \overset{\text{def}}{=} \vec{W} \cdot \vec{X} + b = W^i X_i + b$$

with $\vec{X} = \begin{pmatrix} x \\ x^2 \\ x^3 \end{pmatrix}$

Loss: the difference between estimation and truth

$$L \overset{\text{def}}{=} (\hat{y} - y)^2$$

Cost: the average of losses over all the samples

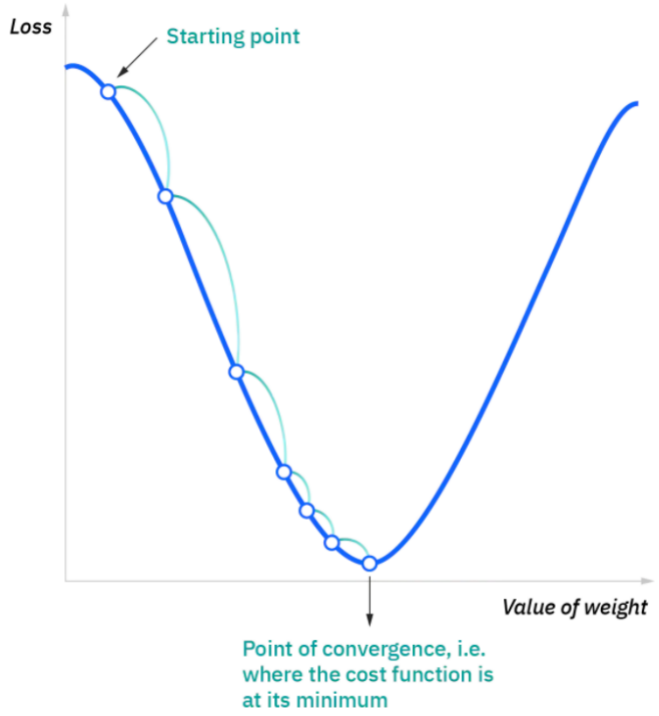
$$J(W, b) \overset{\text{def}}{=} \frac{1}{2m} \sum_{i=1}^{m} L_i = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}(x_i) - y(x_i))^2$$

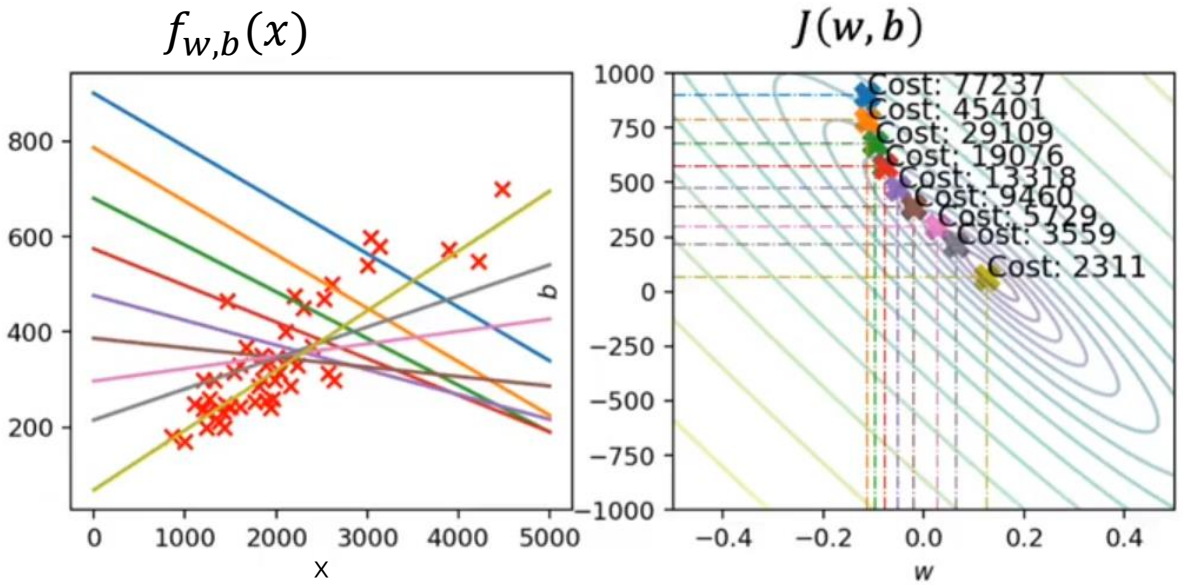
Training: find proper parameters W, b to minimize the cost
-> Best model
Complexity: $O(n^2)$

Analytic solution: $W^* = (X^T X)^{-1} X^T y$
Complexity: $O(n^3)$

Training: Gradient Descent

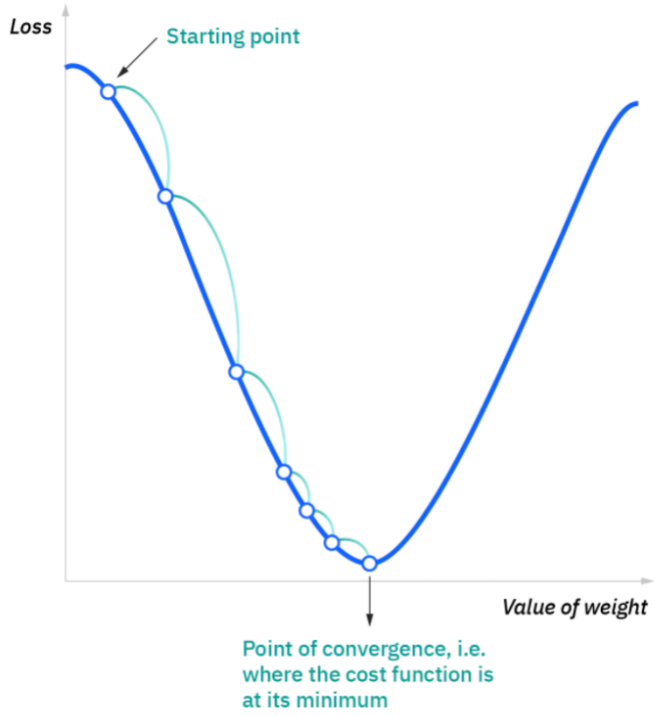
Update parameters in the opposite direction of the gradient of loss function in each step

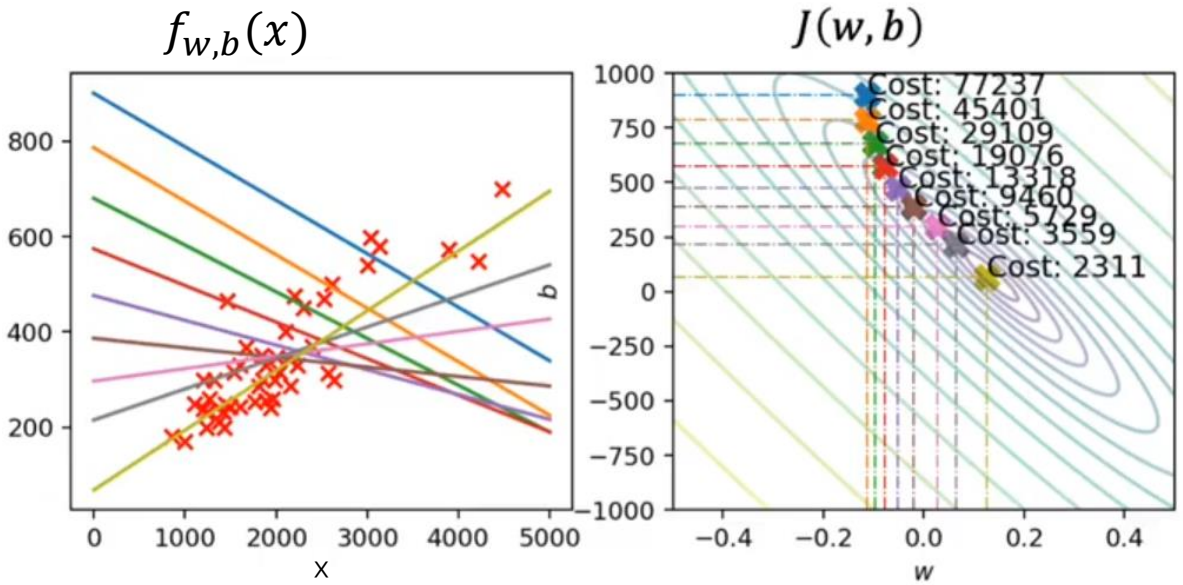




Training: Gradient Descent

Update parameters in the opposite direction of the gradient of loss function in each step

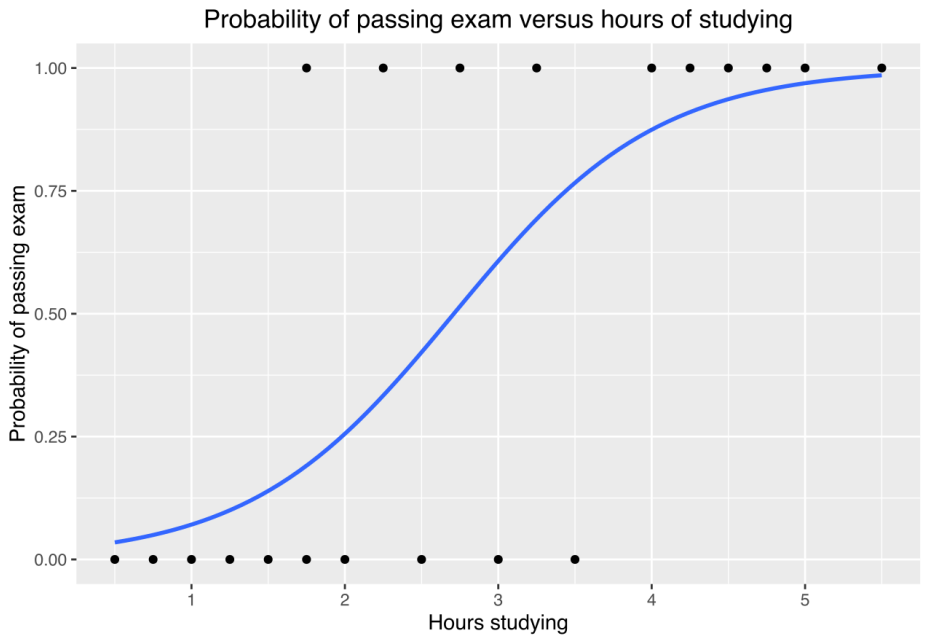




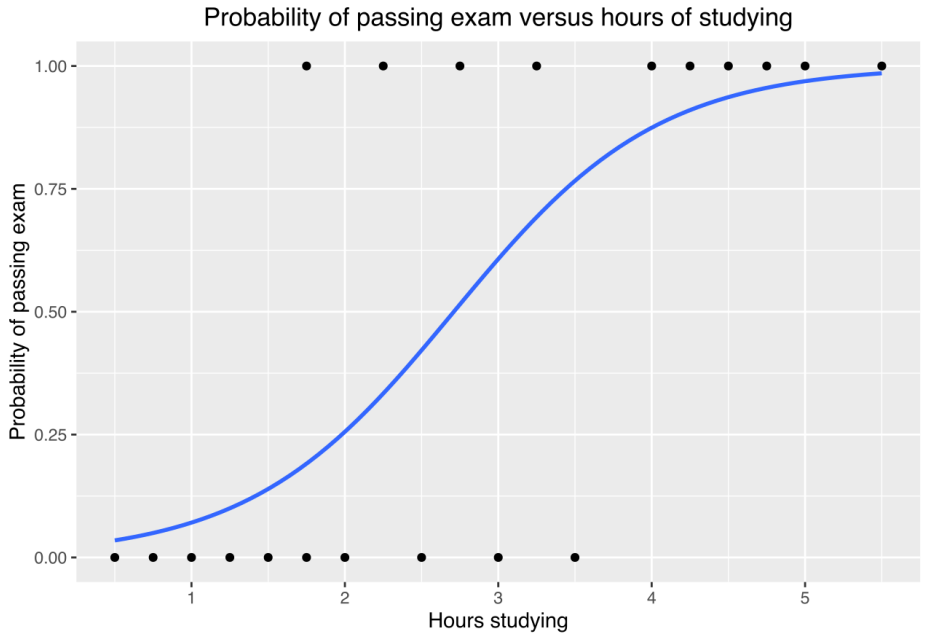
Various of algorithms based on GD:

- Batching: MiniBatchGD, StochasticGD, ⋯
- Momentum: Nesterov, **Adam**, ⋯

Classification: Logistic Regression

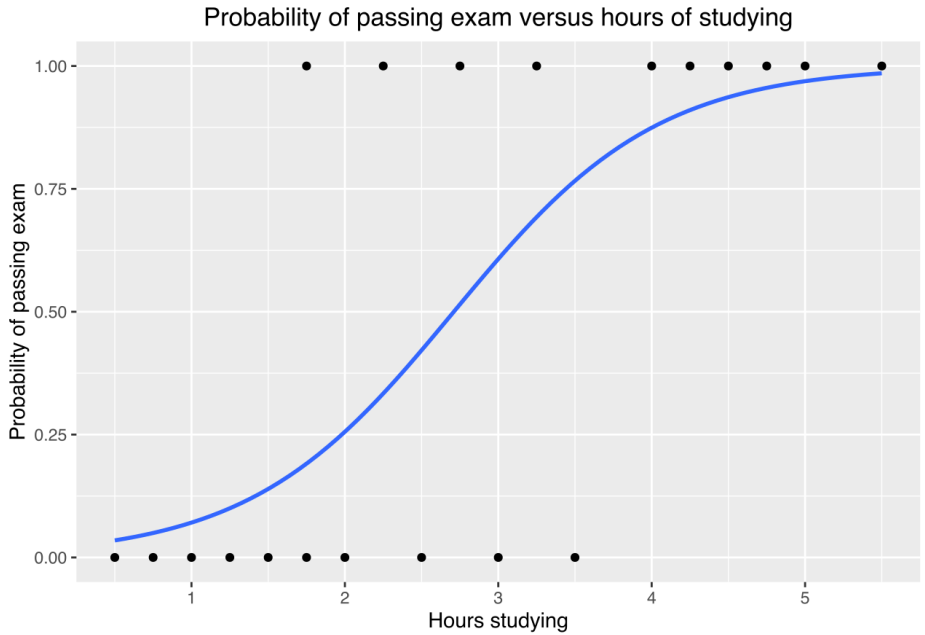


Classification: Logistic Regression



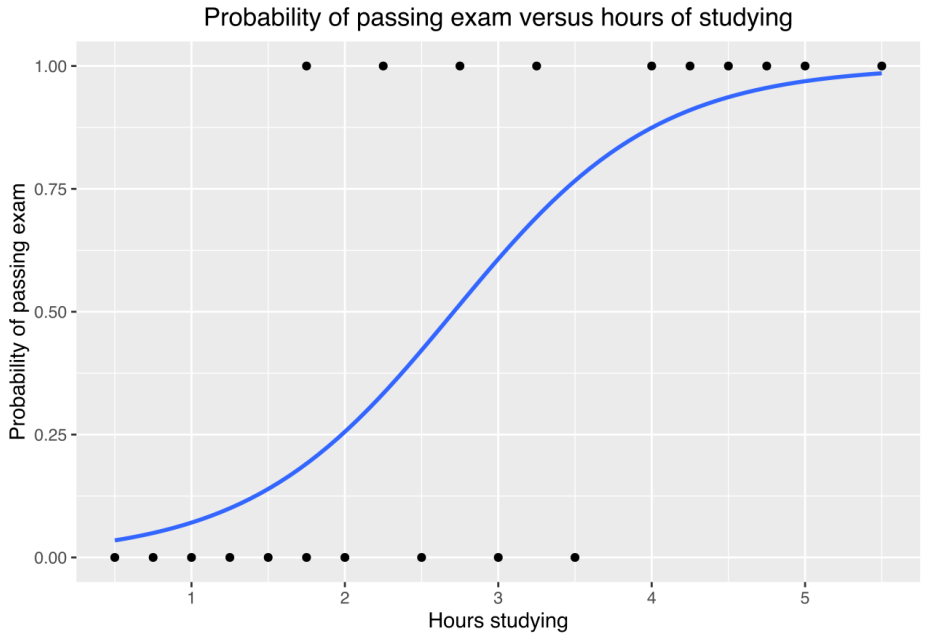
- Regression with boundary [0,1]

Classification: Logistic Regression



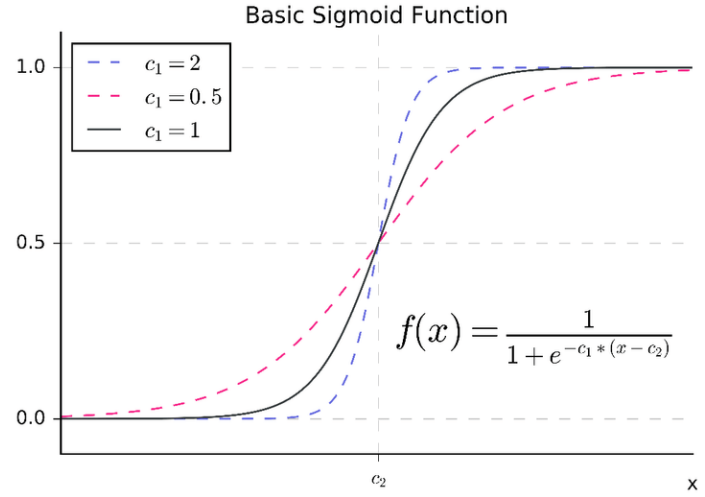
- Regression with boundary [0,1]
- Represent the probability of an event occurring

Classification: Logistic Regression

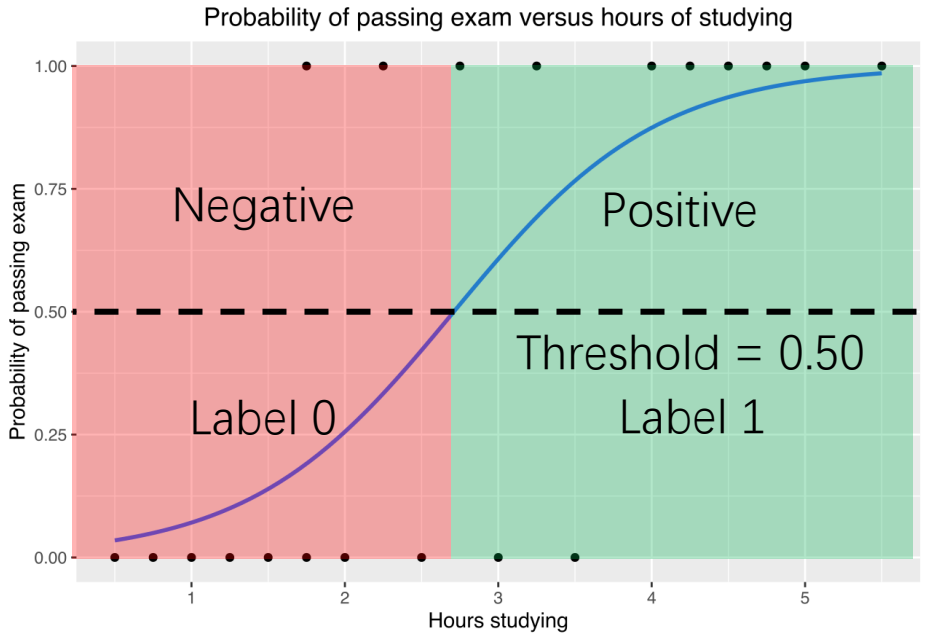


- Regression with boundary [0,1]
- Represent the probability of an event occurring
- Generated by logistic function:

$$g(z) = \frac{1}{1+e^{-z}} \text{ with } z = \vec{w} \cdot \vec{x} + b \text{ called "logit"}$$

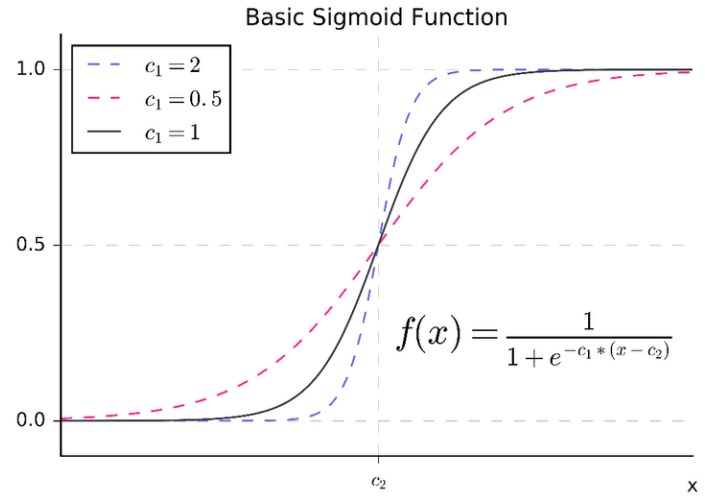


Classification: Logistic Regression



- Regression with boundary [0,1]
- Represent the probability of an event occurring
- Generated by logistic function:

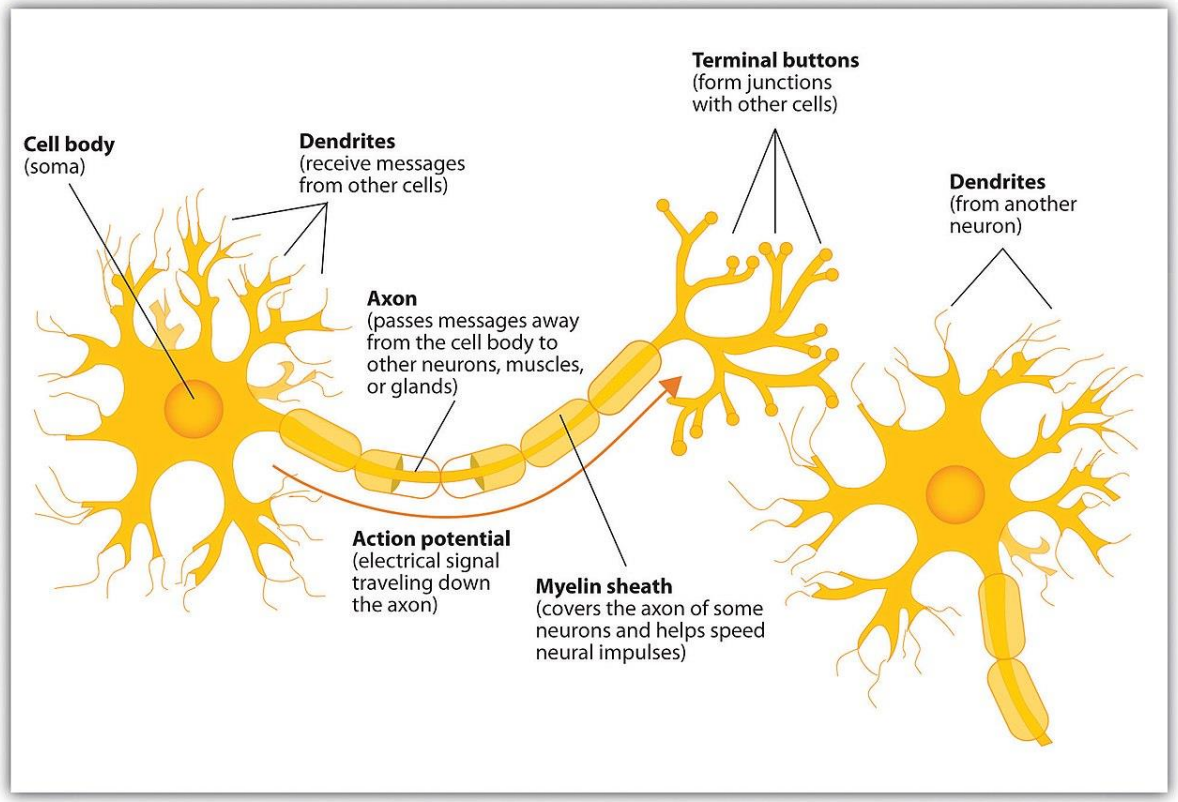
$$g(z) = \frac{1}{1+e^{-z}} \text{ with } z = \vec{w} \cdot \vec{x} + b \text{ called "logit"}$$



- Classification achieved by introducing a threshold

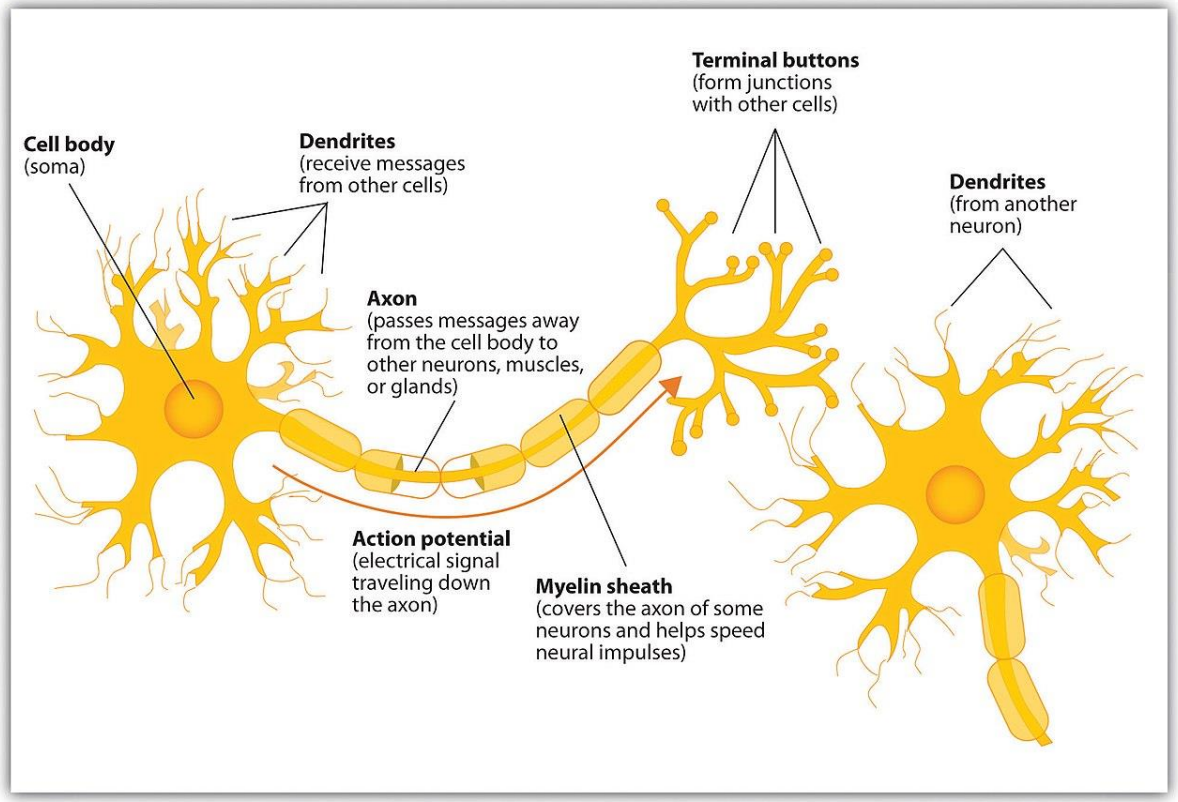
Neural network

Biological neuron

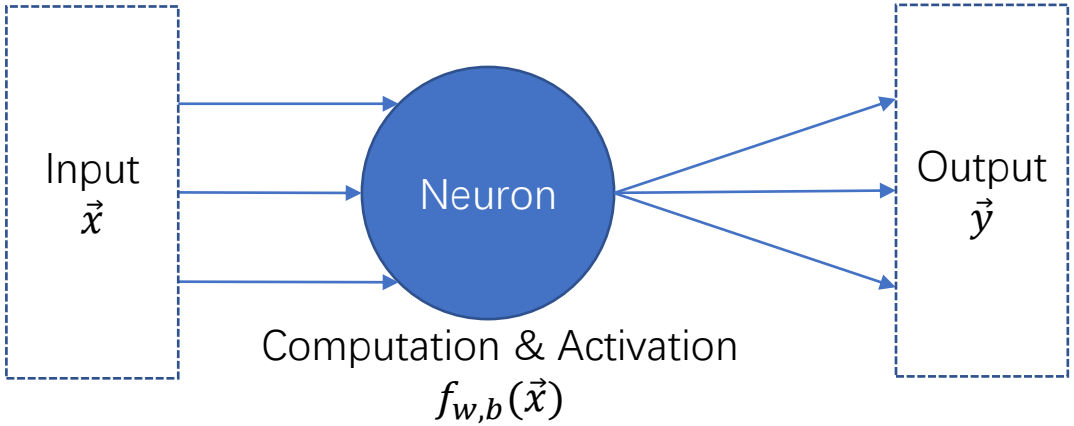


Neural network

Biological neuron

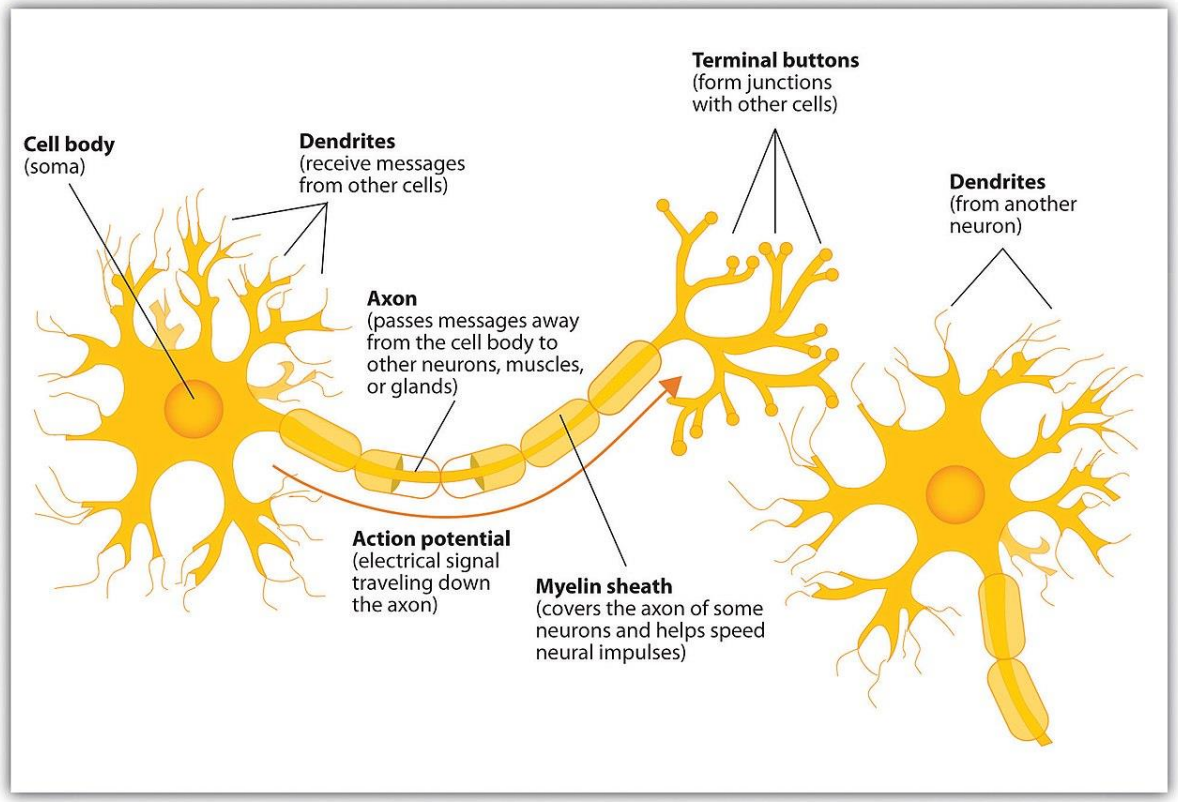


Mathematical simplification

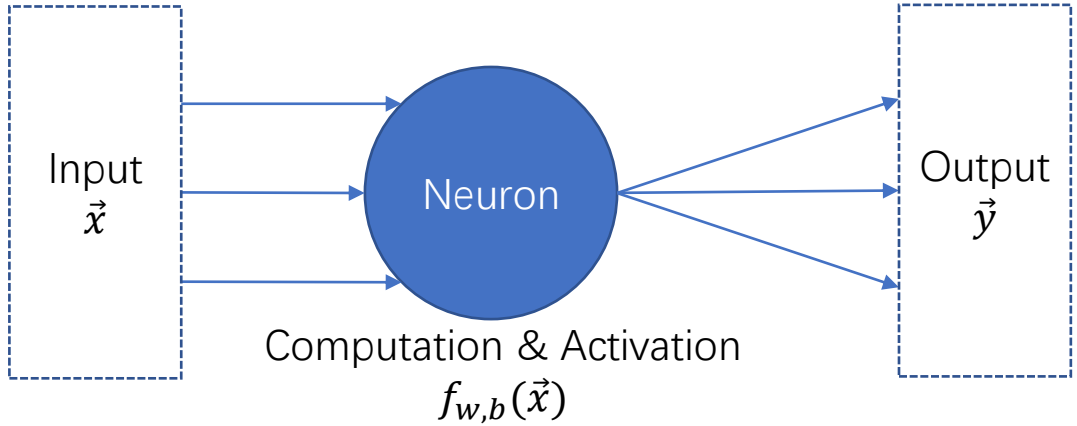


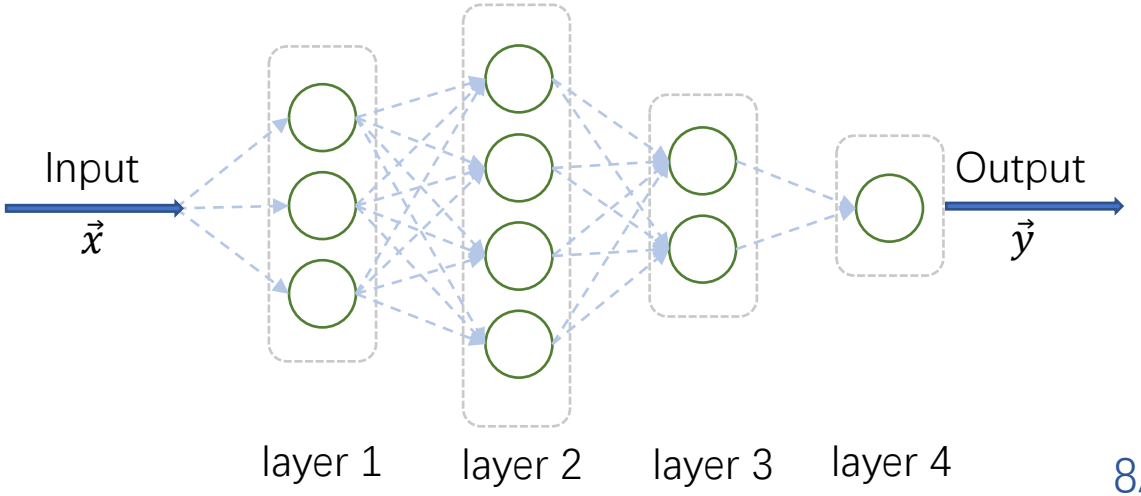
Neural network

Biological neuron



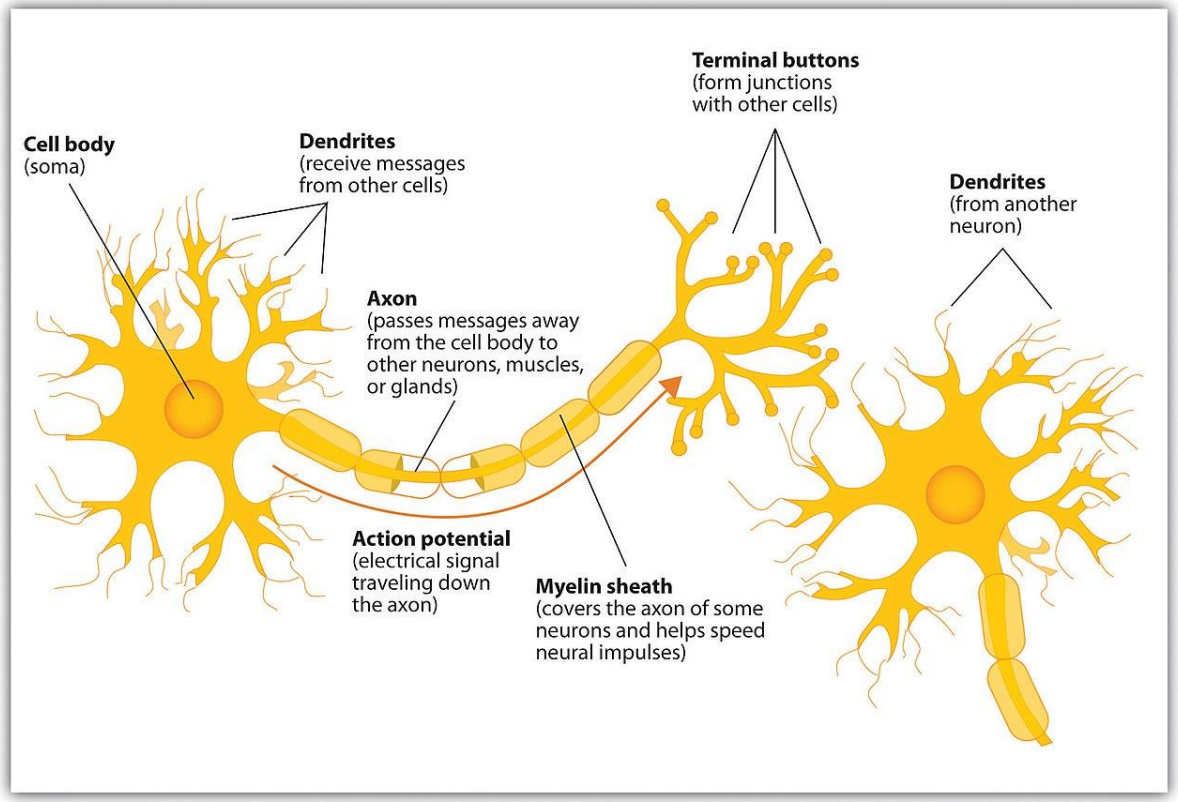
Mathematical simplification



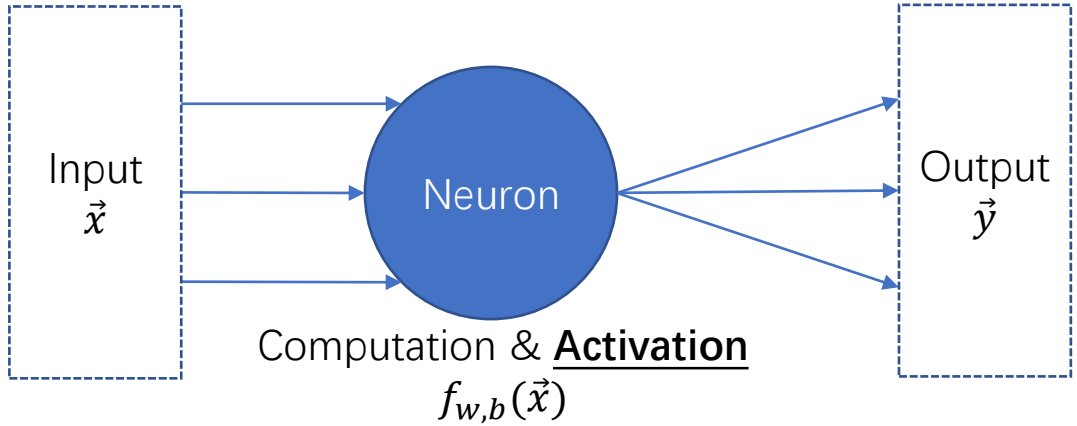


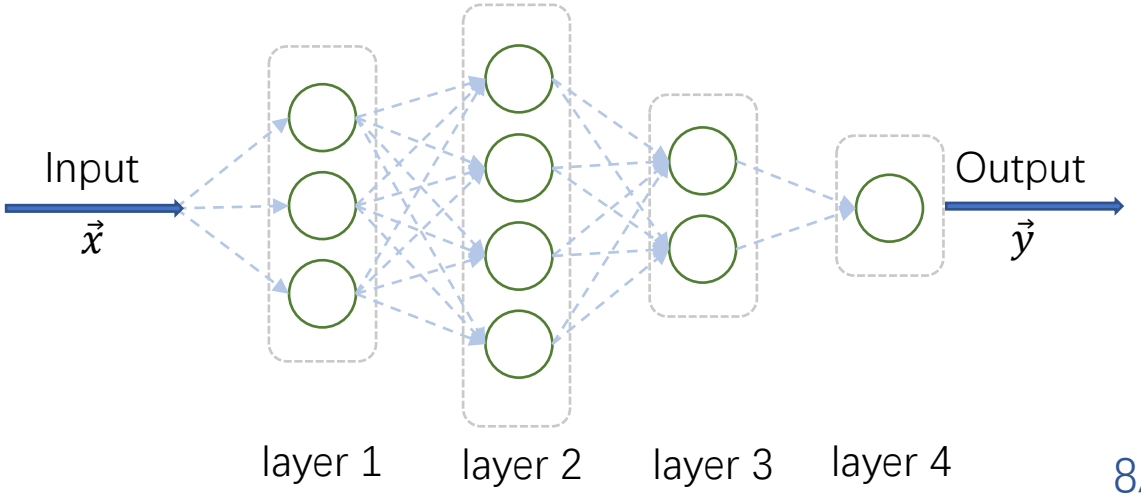
Neural network

Biological neuron



Mathematical simplification

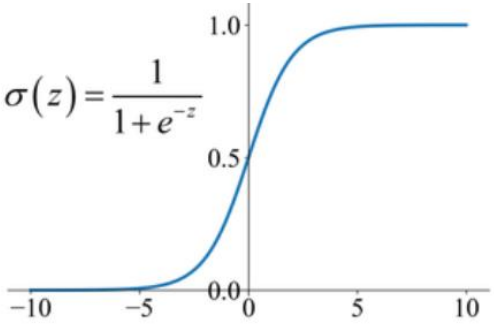




Activation Functions: bring non-linearities

Logistic

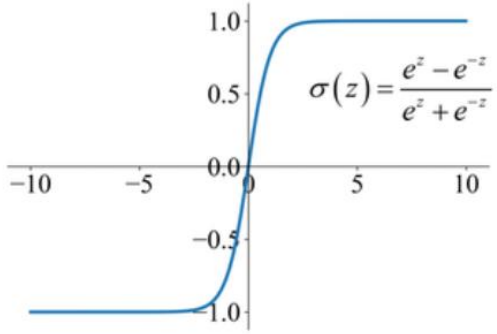
$$\sigma(z) = \frac{1}{1+e^{-z}}$$



(a)

Tanh

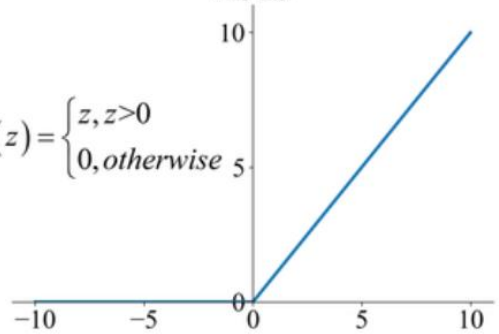
$$\sigma(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$$



(b)

ReLU

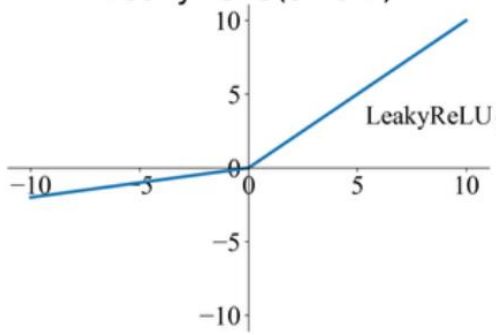
$$\mathrm{ReLU}(z) = \begin{cases} z, z>0 \\ 0, otherwise \end{cases}$$



(c)

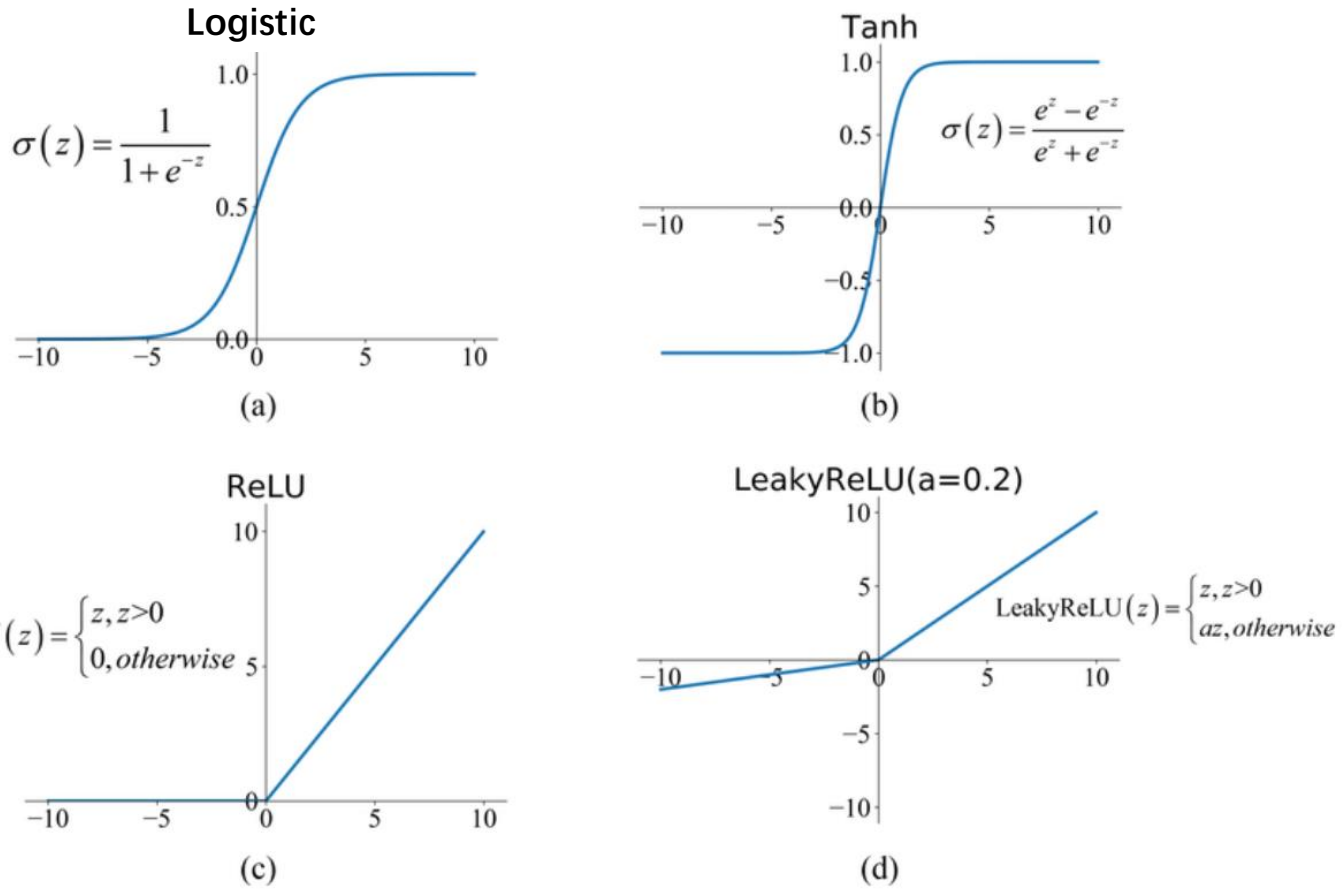
LeakyReLU(a=0.2)

$$\mathrm{LeakyReLU}(z) = \begin{cases} z, z>0 \\ az, otherwise \end{cases}$$



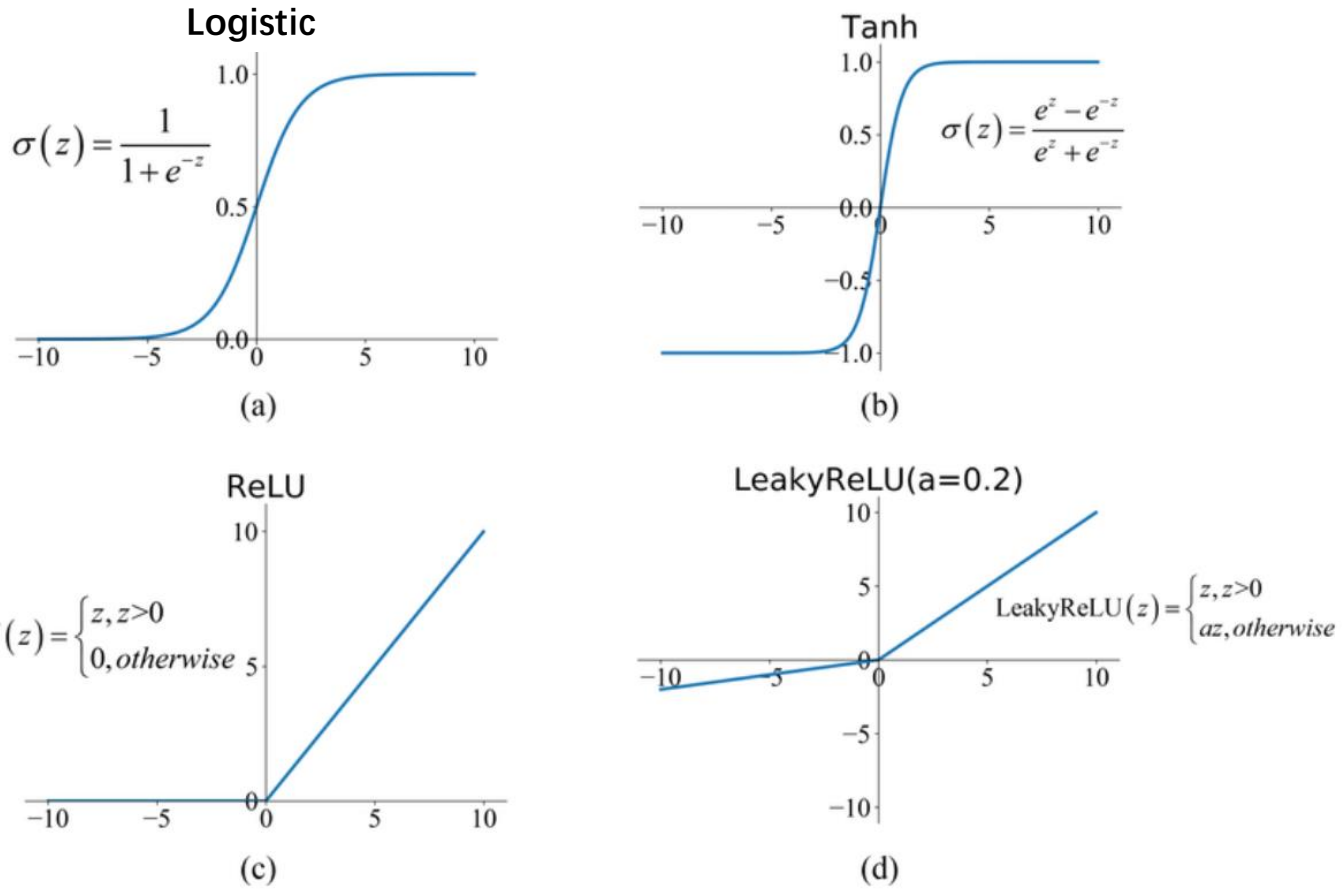
(d)

Activation Functions: bring non-linearities



- Sigmoid functions (Logistic, Tanh)
  - Bounded – ideal for classification
  - Gradient vanish easily – harmful for hidden layers

Activation Functions: bring non-linearities

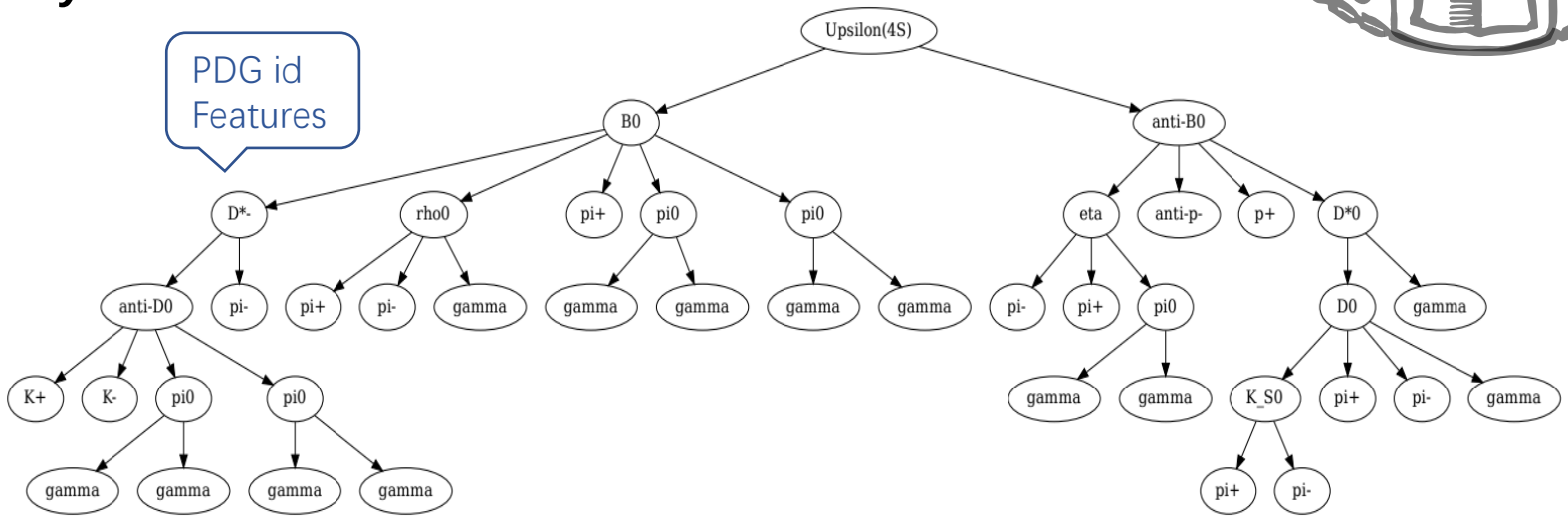


- Sigmoid functions (Logistic, Tanh)
  - Bounded – ideal for classification
  - Gradient vanish easily – harmful for hidden layers
- (Partially) Linear activations
  - Fast to calculate
  - Allowing large values
  - Not differentiable at 0 – some solutions available

Tree Structures of Particle Decay

⇅

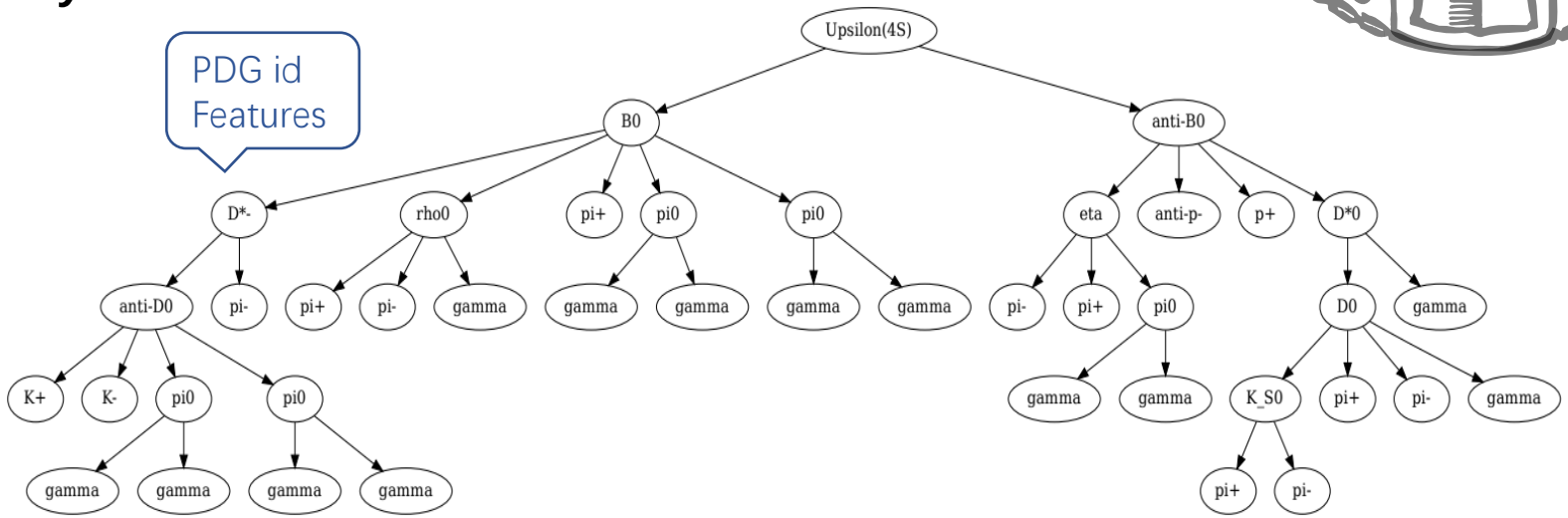
Graph Neural Network



Tree Structures of Particle Decay

⇅

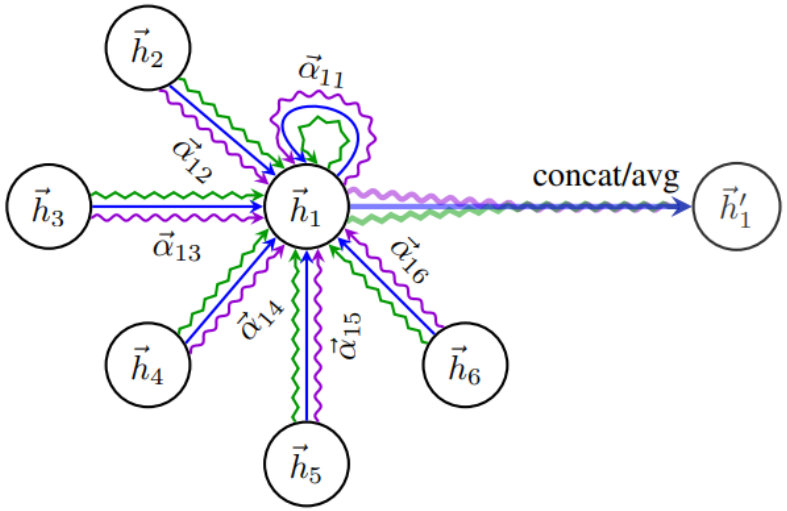
Graph Neural Network

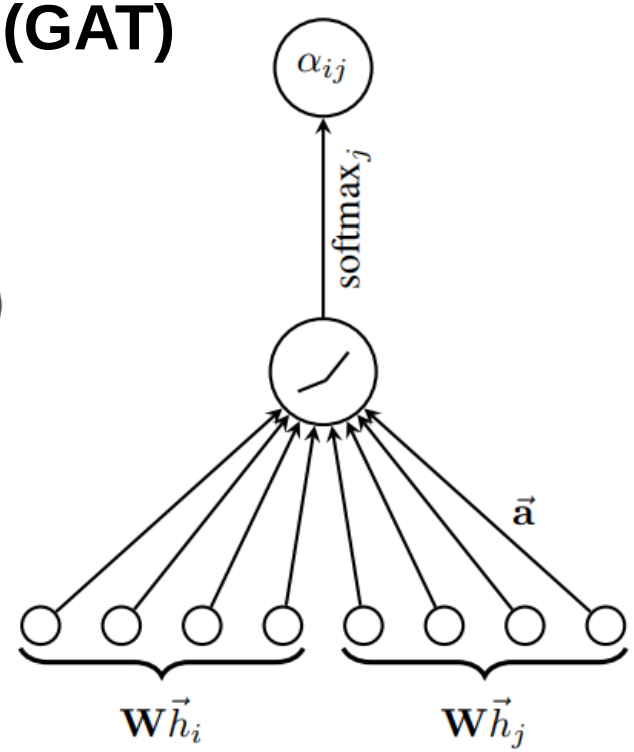


Dataset:

- Each event (each **Graph**):
  - Decay of $\Upsilon(4S) \rightarrow B^0\bar{B}^0$
  - Particles (**Nodes**)
  - Mother/Daughter relations (two way **Edges**) + self loops
  - Each particle (each **Node**)
    - **PDG** ids
    - 8 **Features**: **Production time**, **Energy**, **Position** (3d), **Momentum** (3d)
- **Labels** per event: Pass/Fail after the reconstruction of B decays (FEI skims)
- Other event level **attributions** for further analysis: e.g. $\boldsymbol{M_{bc}}$, etc.

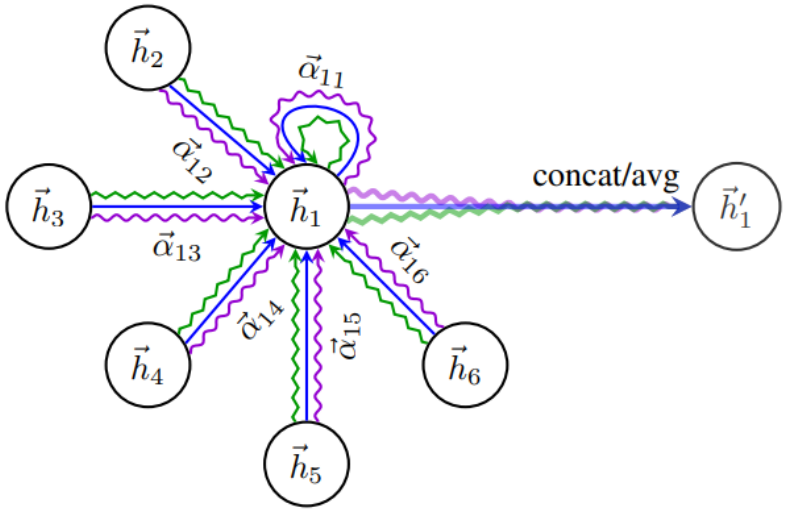
Graph Attention Network (GAT)

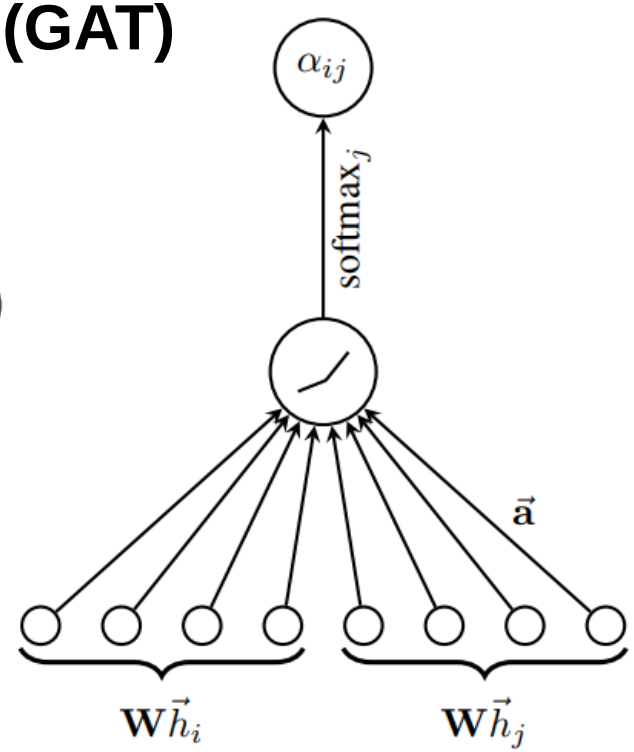




- Update the node $\vec{h}_1$ by adding up the information from all of its neighbours and itself

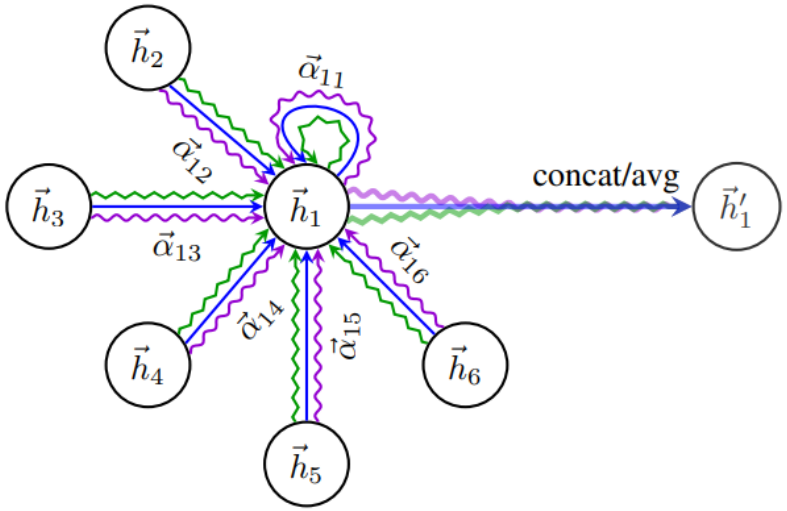
Graph Attention Network (GAT)

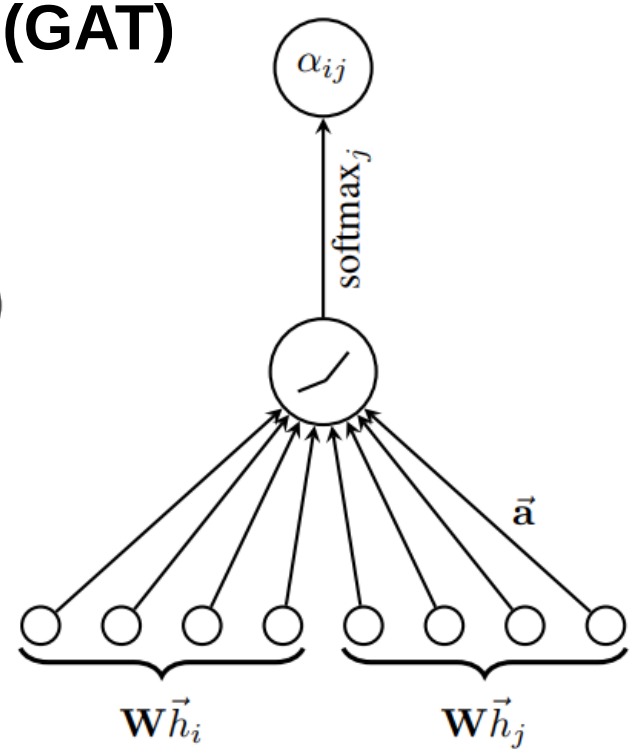




- Update the node $\vec{h}_1$ by adding up the information from all of its neighbours and itself
- Each pair has an attention weight α_{ij}, calculated from the information of both nodes $\vec{h}_i$ and $\vec{h}_j$

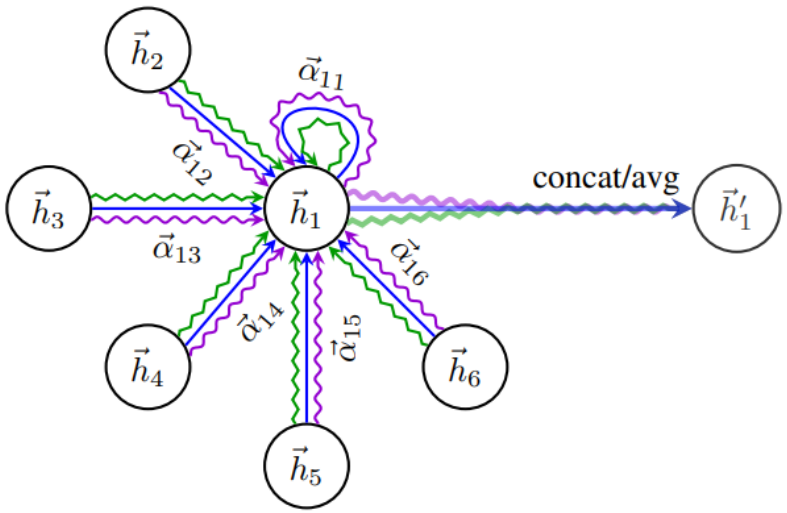
Graph Attention Network (GAT)

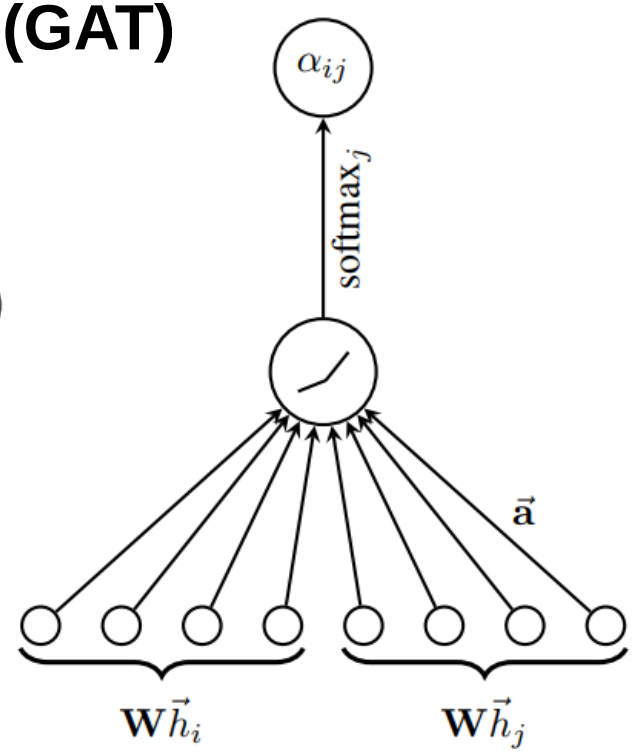




- Update the node $\vec{h}_1$ by adding up the information from all of its neighbours and itself
- Each pair has an attention weight α_{ij}, calculated from the information of both nodes $\vec{h}_i$ and $\vec{h}_j$
- Several sets of attention weights are trained parallelly to extract different aspects of correlation between nodes

Graph Attention Network (GAT)



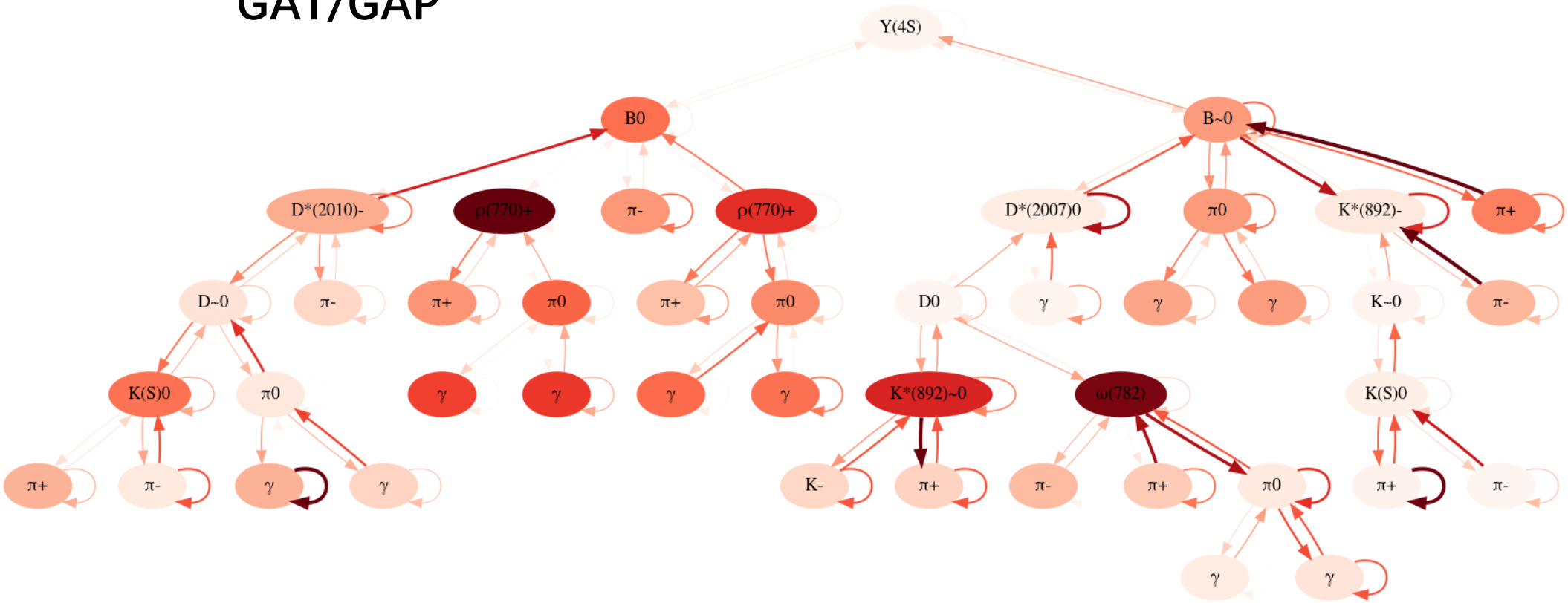


- Update the node $\vec{h}_1$ by adding up the information from all of its neighbours and itself
- Each pair has an attention weight α_{ij}, calculated from the information of both nodes $\vec{h}_i$ and $\vec{h}_j$
- Several sets of attention weights are trained parallelly to extract different aspects of correlation between nodes

Global Attention Pooling (GAP)

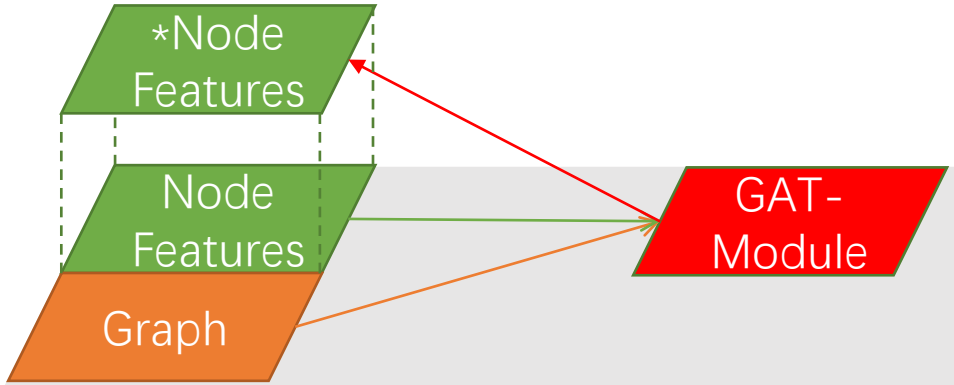
- Aggregate the information from all the nodes with learned attention weight for each node
  -> Deconstruction of graph structure

Visualization of Edge/Node-Attentions for GAT/GAP

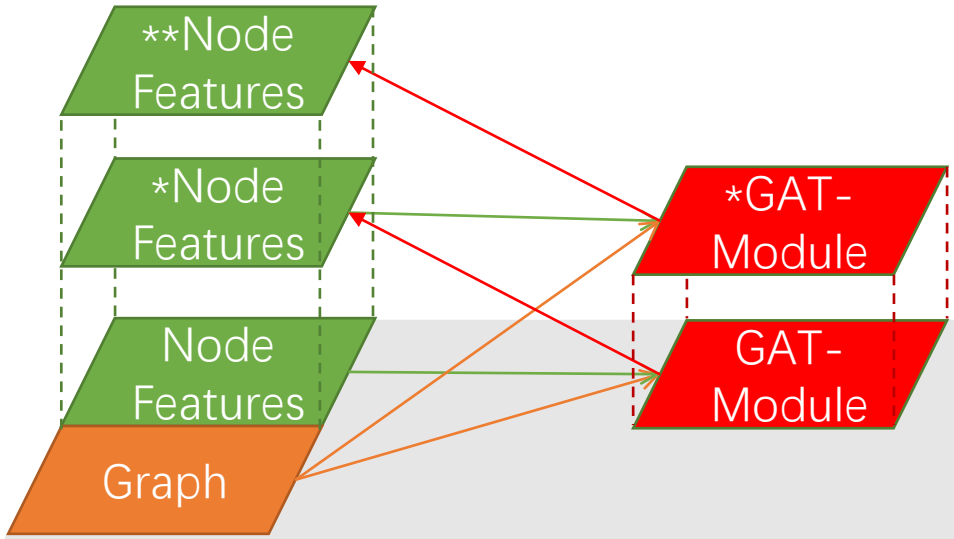


Darker ⇄ More attention

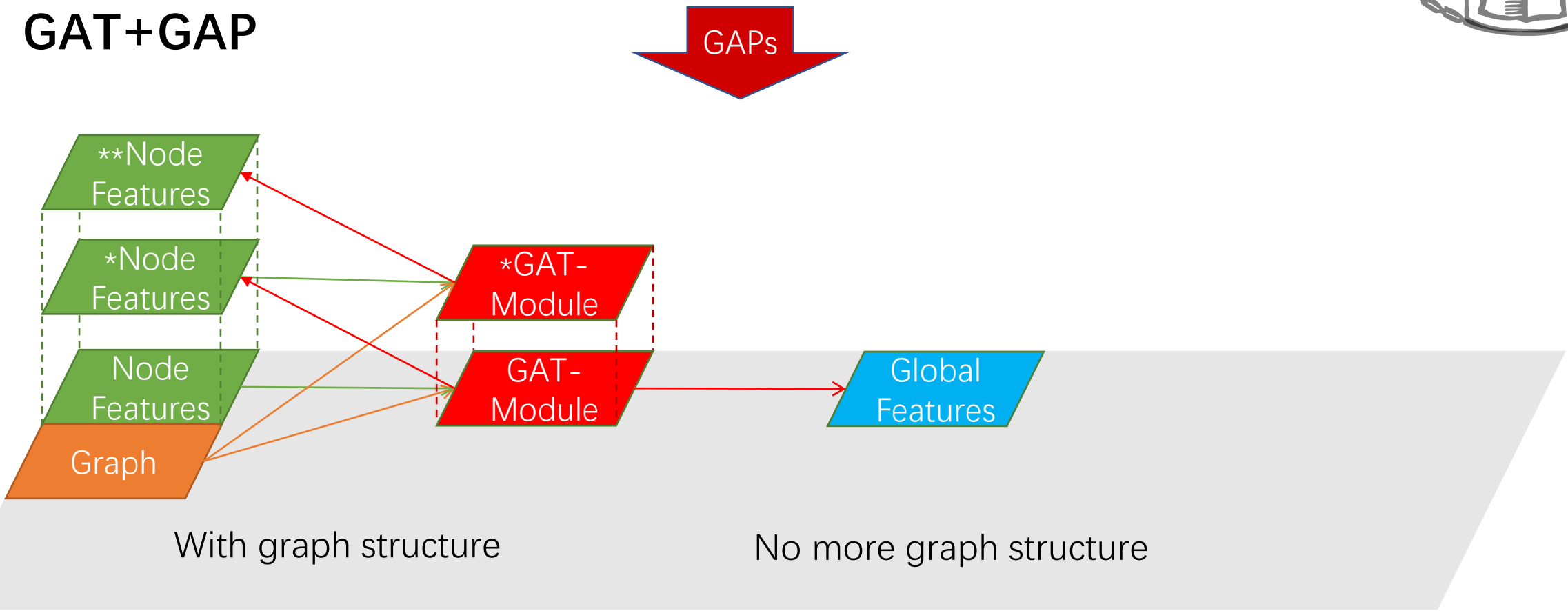
Final Architecture: GAT …



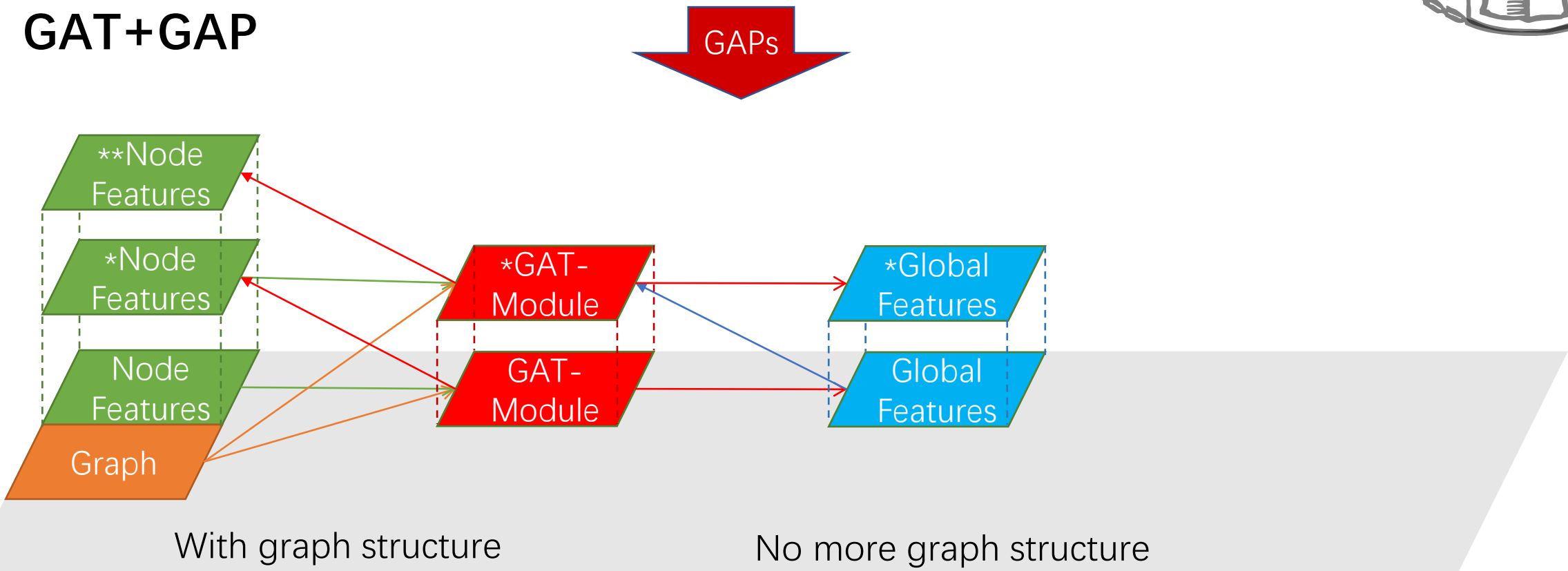
Final Architecture: GAT …



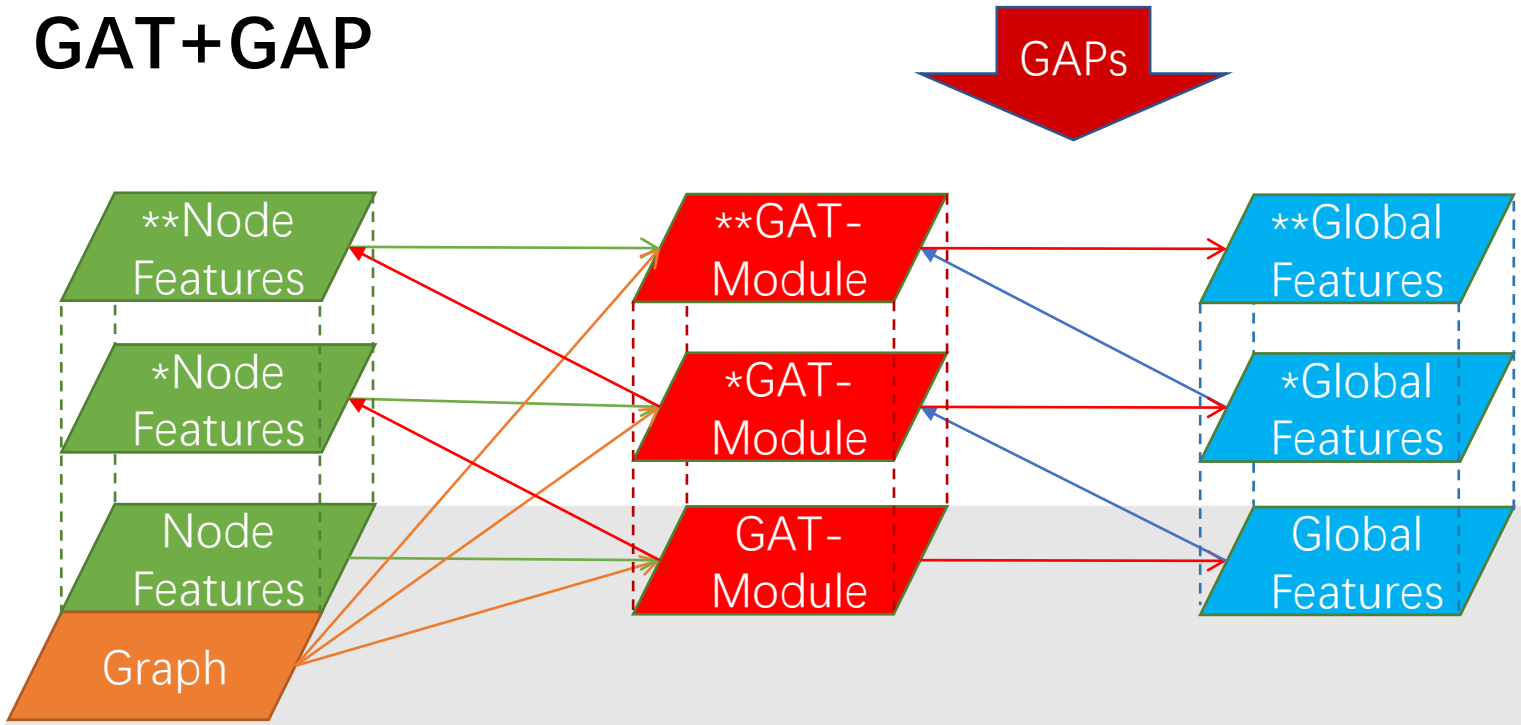
Final Architecture: GAT+GAP



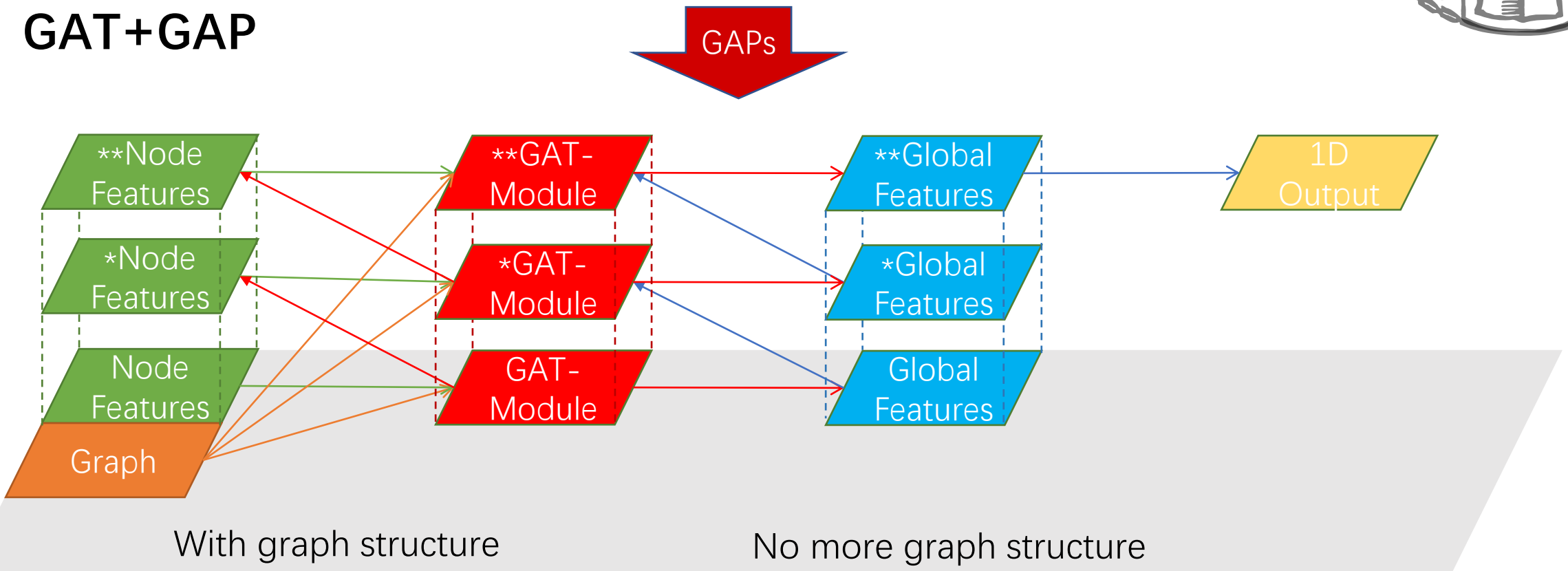
Final Architecture: GAT+GAP



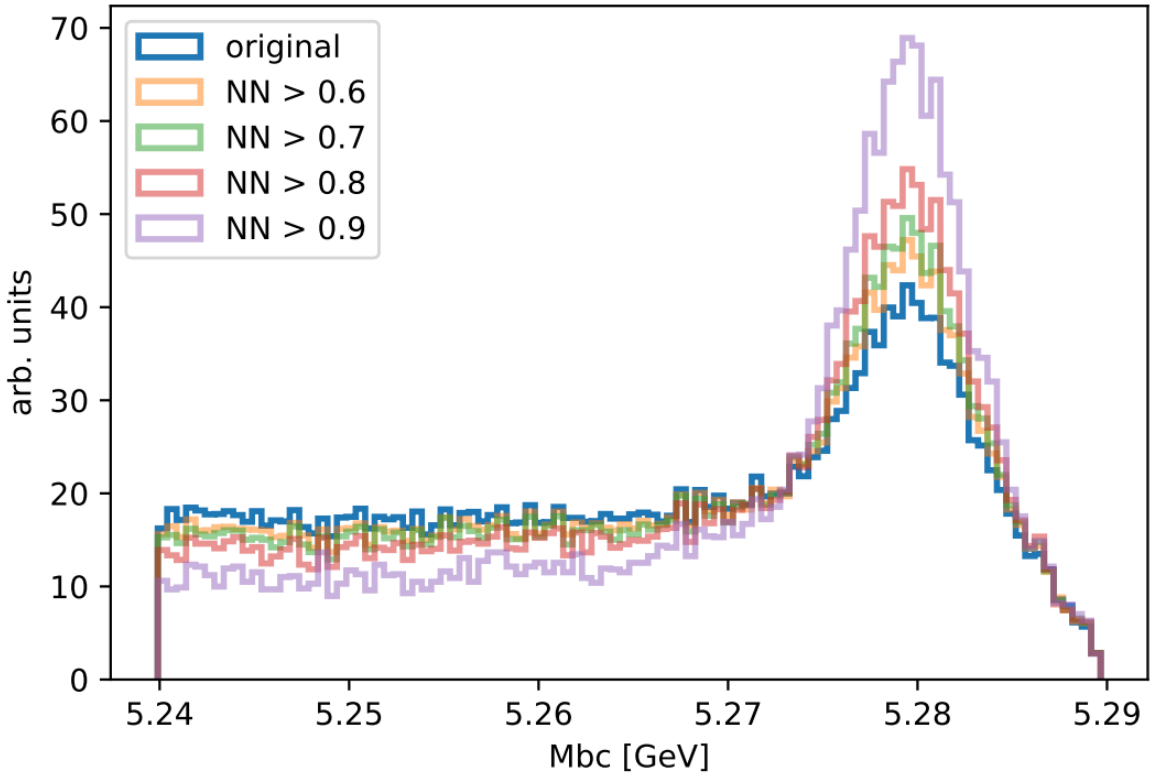
Final Architecture: GAT+GAP

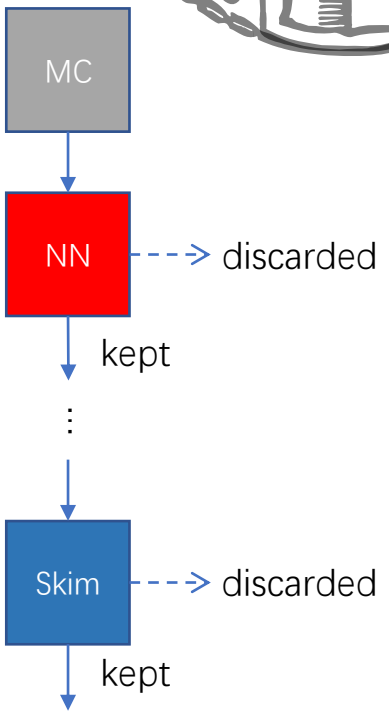


Final Architecture: GAT+GAP

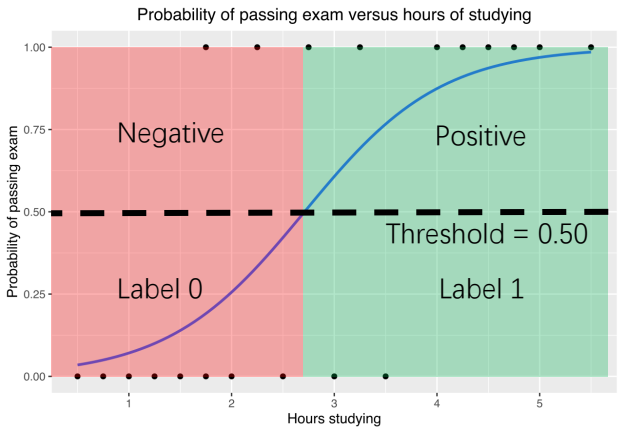


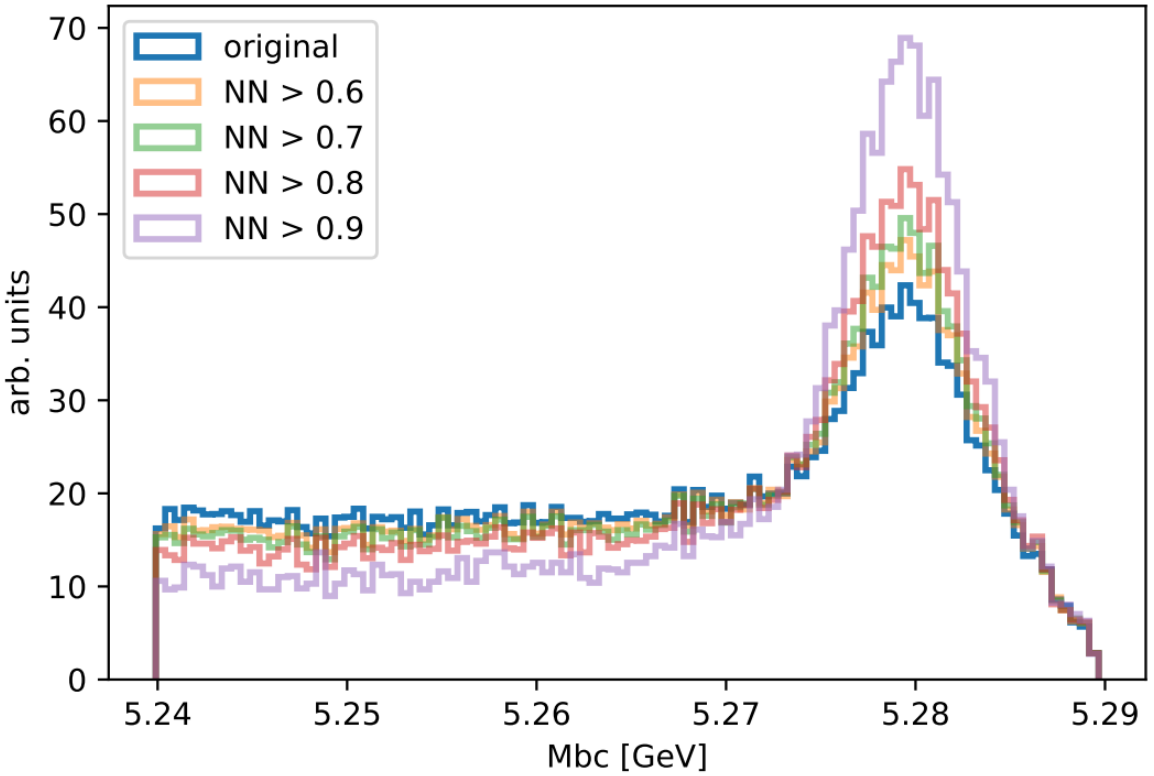
Bias

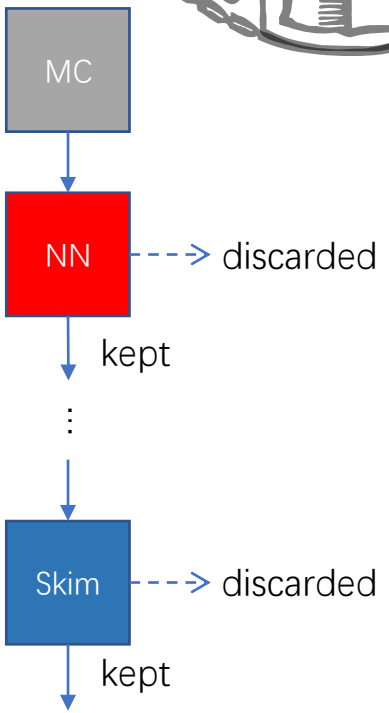

original
NN > 0.6
NN > 0.7
NN > 0.8
NN > 0.9
arb. units
0
10
20
30
40
50
60
70
5.24
5.25
5.26
5.27
5.28
5.29
Mbc [GeV]


MC
NN
discarded
kept
Skim
discarded
kept

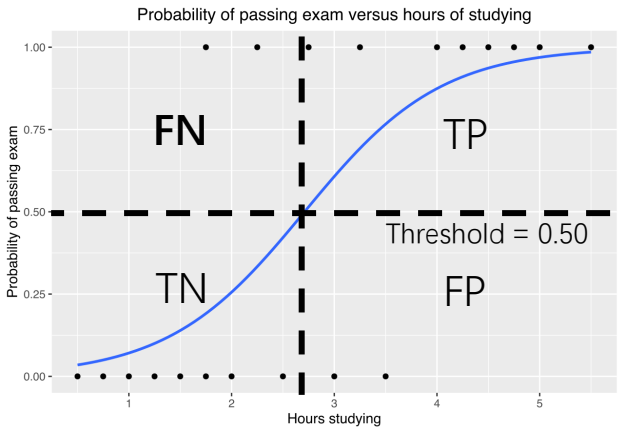
Bias

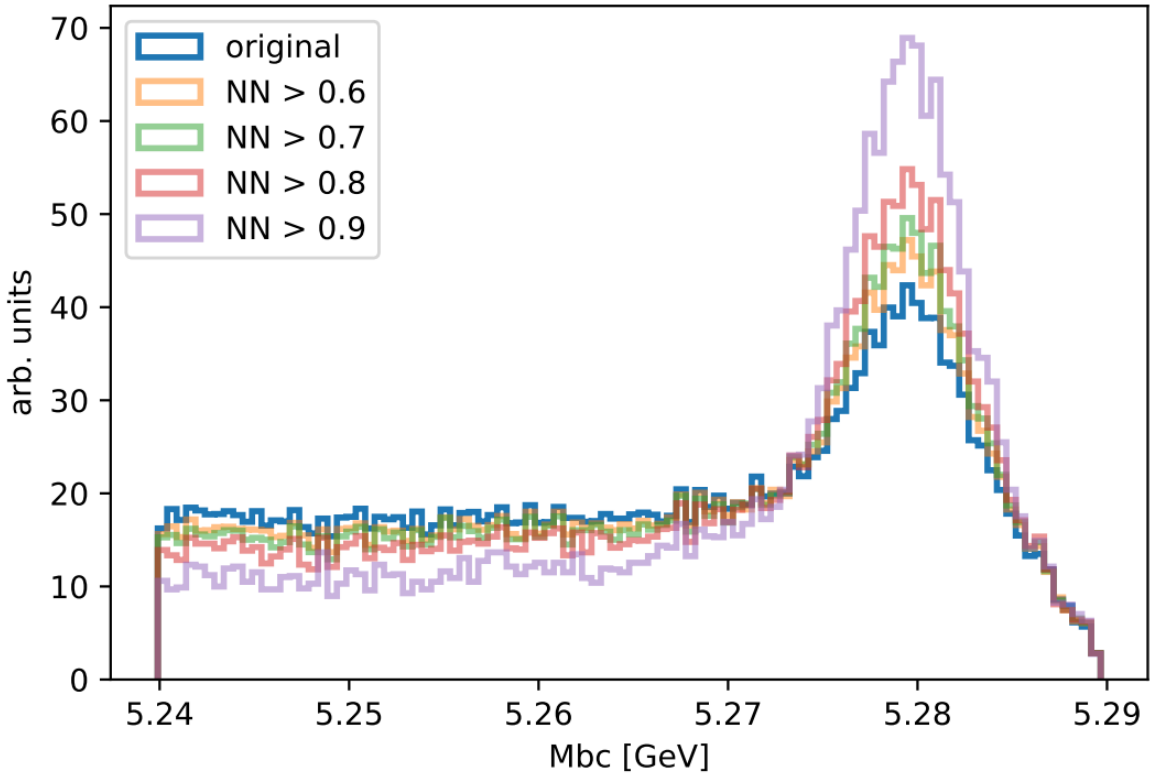

Probability of passing exam versus hours of studying
Negative
Positive
Threshold = 0.50
Label 0
Label 1
Probability of passing exam
Hours studying

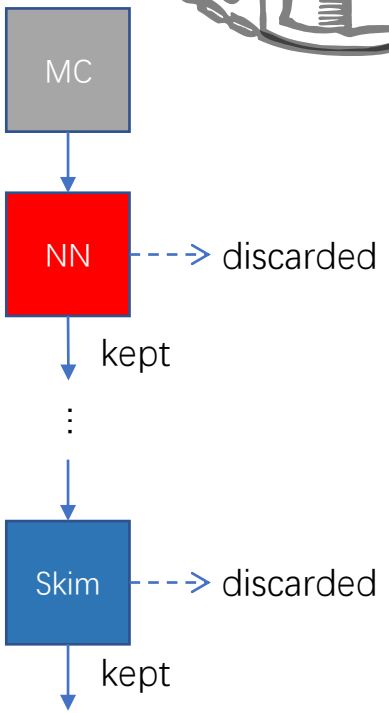

original
NN > 0.6
NN > 0.7
NN > 0.8
NN > 0.9
arb. units
Mbc [GeV]


MC
NN
discarded
kept
Skim
discarded
kept

Bias due to False-Negatives with Naive Filtering







Skim \ NN	Positive	Negative
Pass	True-Positive (TP)	**False-Negative (FN)**
Fail	False-Positive (FP)	True-Negative (TN)

Weighting Methods to Reduce Bias

	Sampling Method	Reweighting Method
Use of NN output		
Weight		
Loss to train NN		

Weighting Methods to Reduce Bias

	Sampling Method	Reweighting Method
Use of NN output	As probability to keep event randomly	As score for selection according to fix threshold
Weight		
Loss to train NN		

Weighting Methods to Reduce Bias

	Sampling Method	Reweighting Method
Use of NN output	As probability to keep event randomly	As score for selection according to fix threshold
Weight	Inverse of NN output	Decided with the help of another classifier
Loss to train NN		

Weighting Methods to Reduce Bias

	Sampling Method	Reweighting Method
Use of NN output	As probability to keep event randomly	As score for selection according to fix threshold
Weight	Inverse of NN output	Decided with the help of another classifier
Loss to train NN	Speedup	Binary cross entropy

Metric: Speedup

--Improvement of computation time to produce the same

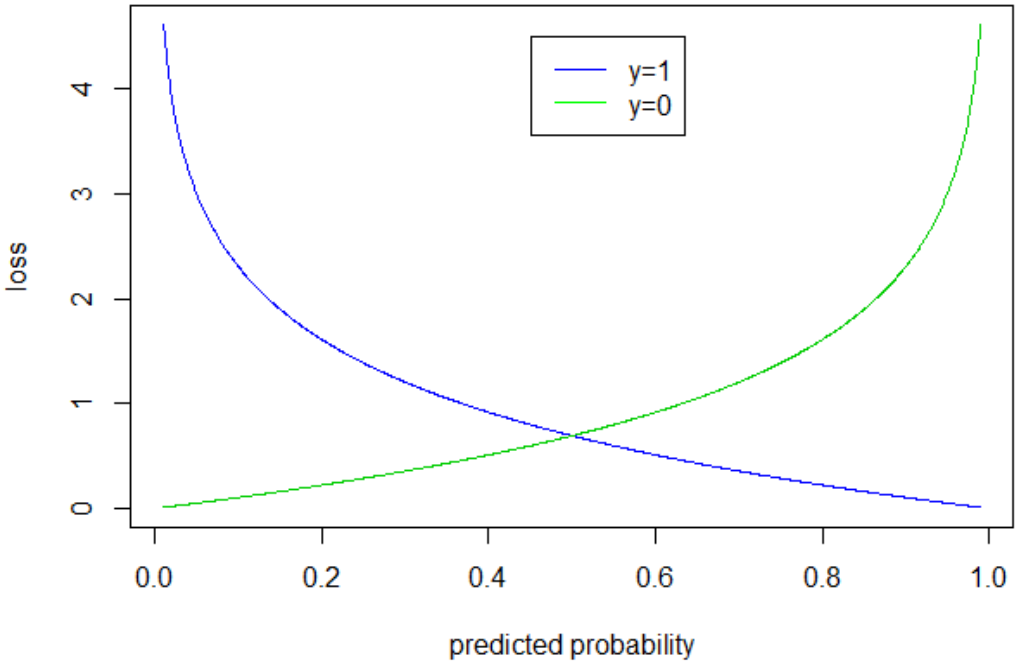
effective number of events with the help of NN filter:

$$\text{Speedup: } s = \frac{t_{no\_filter}}{t_{filter}}$$

$$\text{Effective Sample Size: } N_{eff} = \frac{(\sum \omega_i)^2}{\sum \omega_i^2}$$

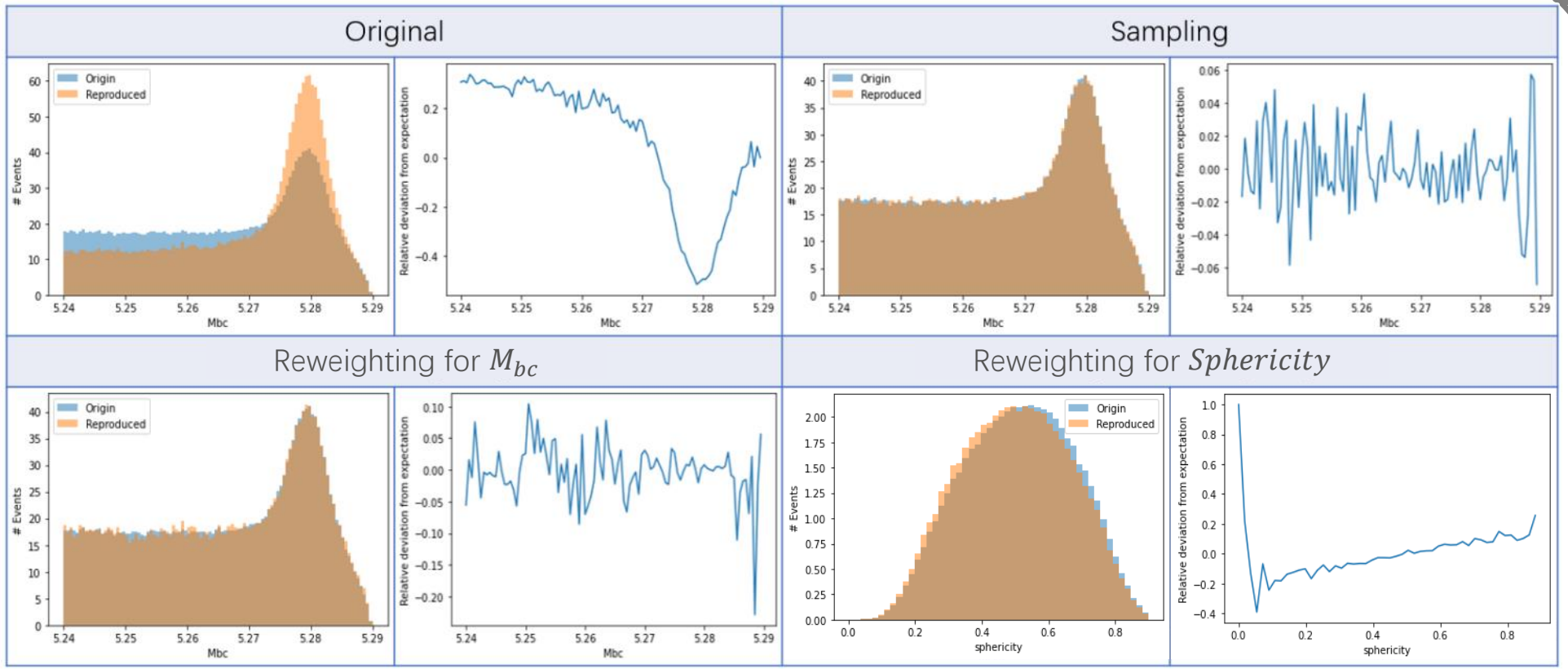
Metric: Binary Cross Entropy

-- Amount of information provided by the prediction

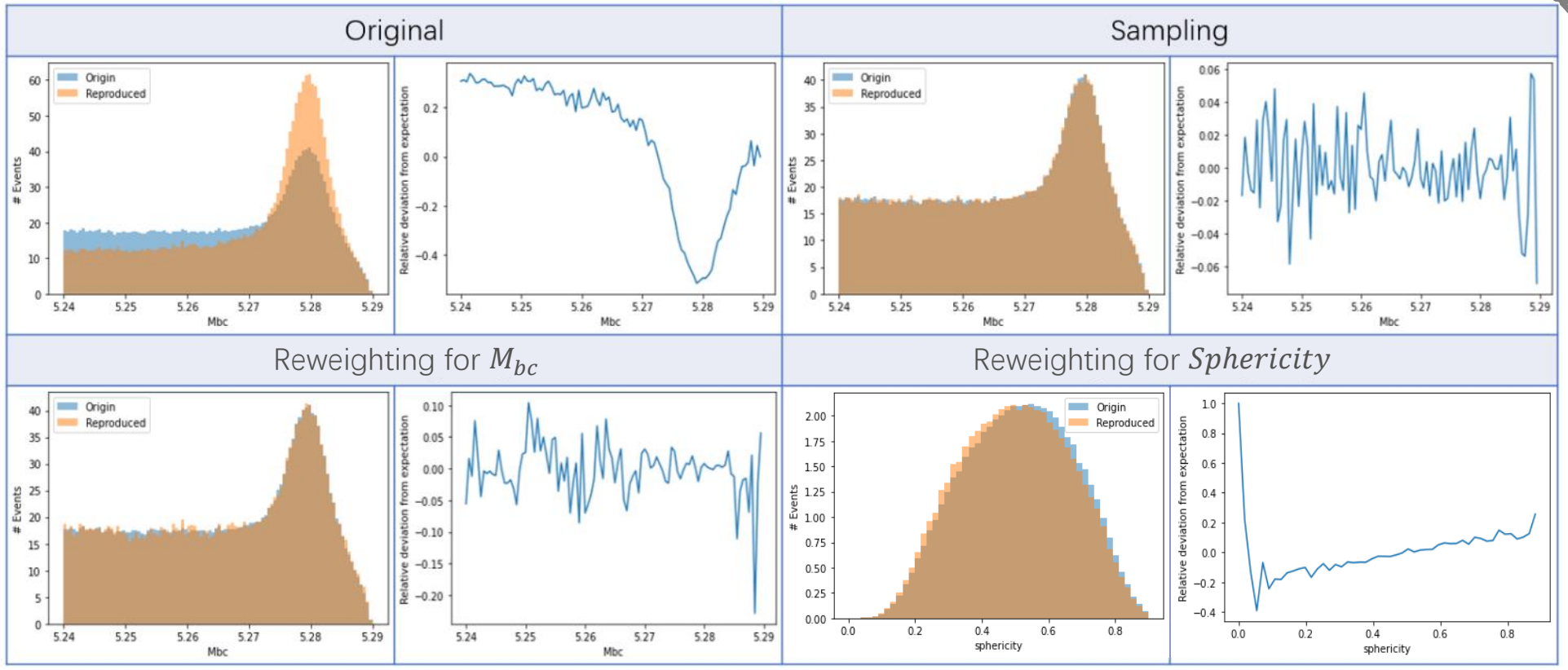


$$H(x) = -\sum_{i=1}^{n} y_i(x)\log(p_i(x))$$
$$= -y(x)\log\big(p(x)\big) - \big(1-y(x)\big)\log(1-p(x))$$
$$= \begin{cases} -\log(p(x)), & \text{for y(x)=1} \\ -\log(1-p(x)), & \text{for y(x)=0} \end{cases}$$

Performances



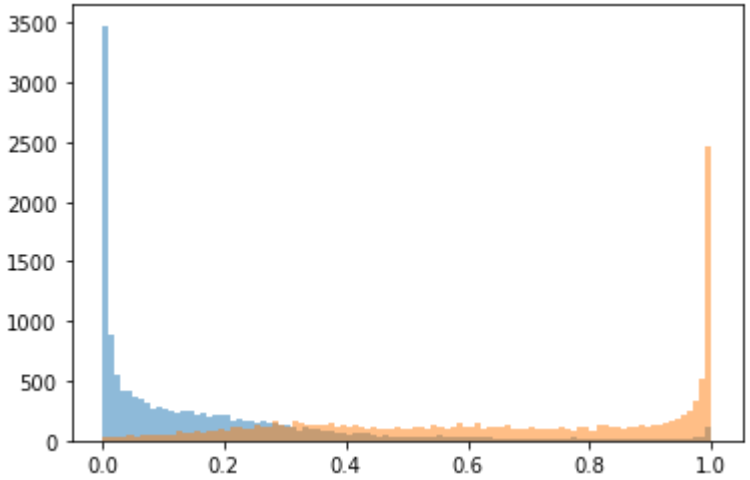
Performances



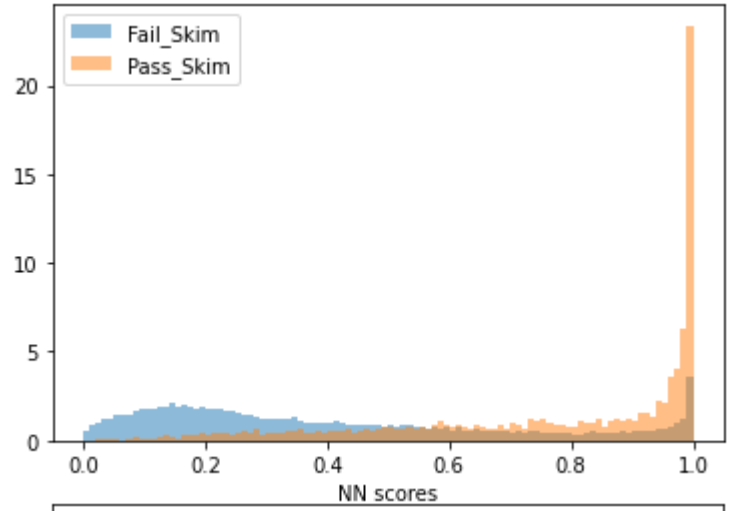
	Sampling	Reweighting
Maximum speedup	2.0	6.5
Bias	No bias	Potential residual bias

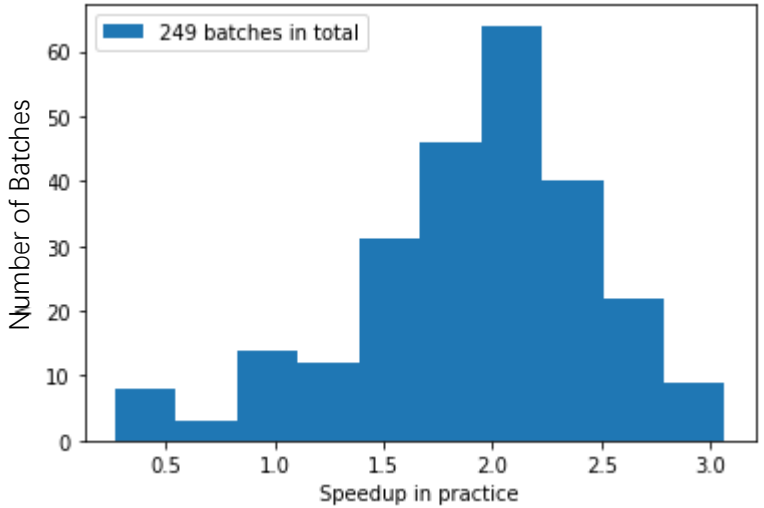
Performances on another skim with sampling

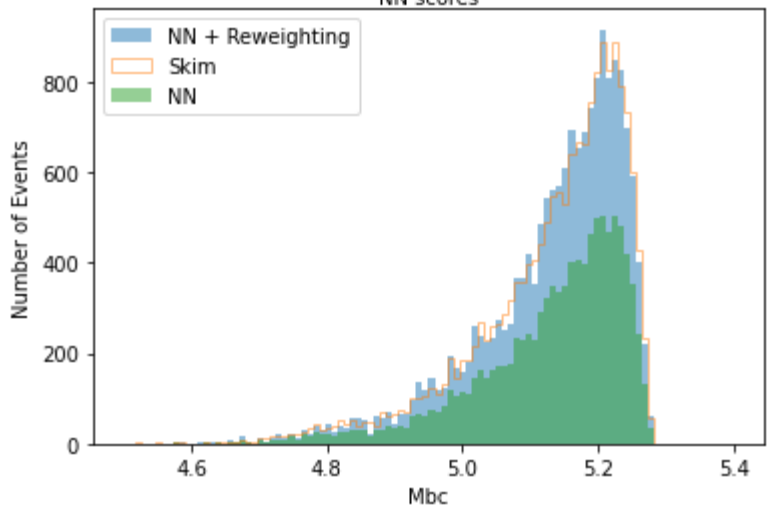
Predictions during training



Predictions in test workflow







Results:

- Graph Attention Networks can learn the information from particle decay trees and help to simulate skims with low retention rate at early stage
- Bias is avoided with sampling method while a speedup of factor 2 can still be maintained.
- Reweighting method can reach much higher speedup up to 6.5 but will still have some bias in the variables that are not used in the training of the extra classifier.
- SmartBKG can be invested to other skims

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Thank You for your Attention

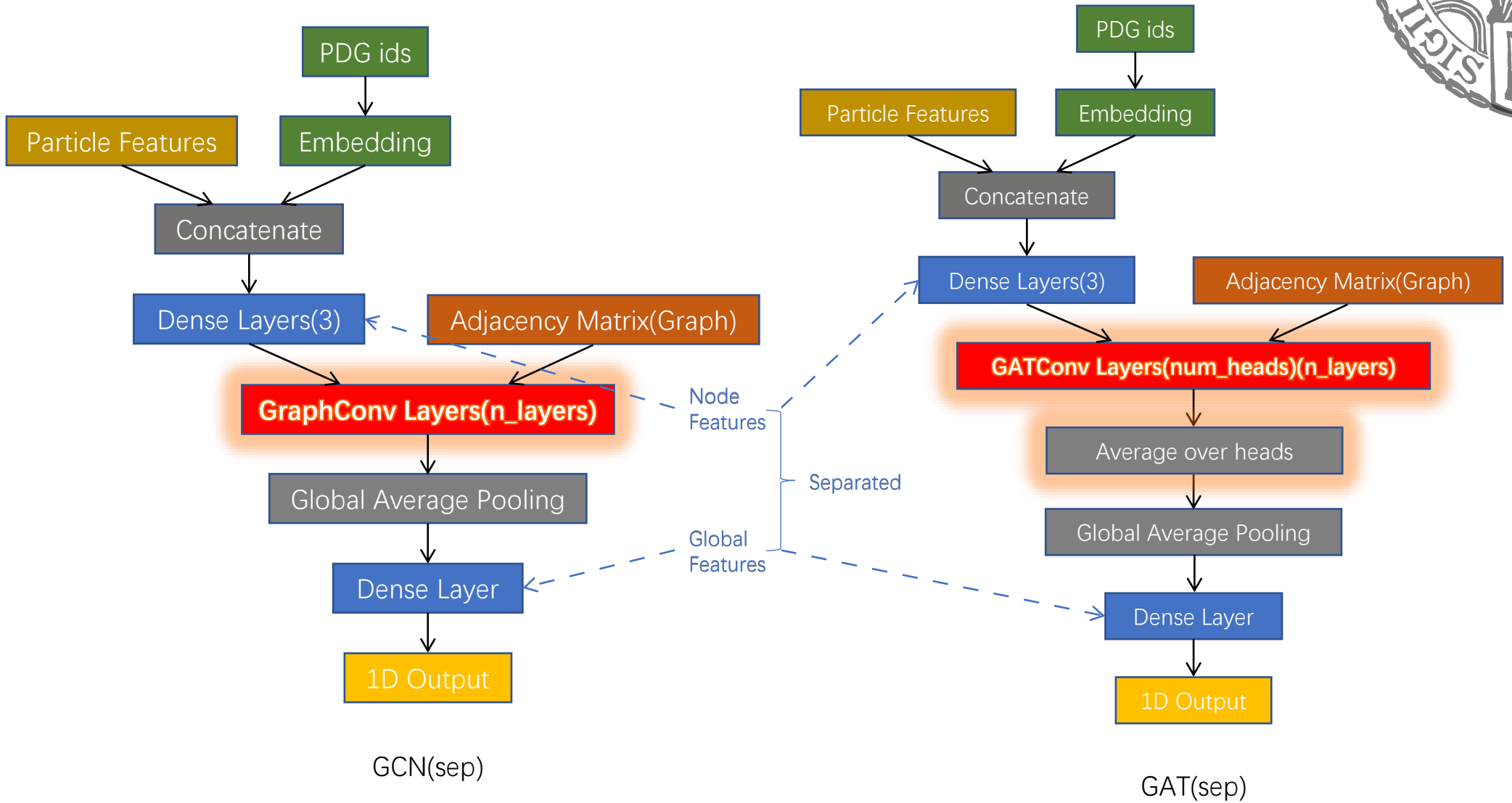
Boyang Yu, Nikolai Hartmann, Luca Schinnerl, Thomas Kuhr

Joint Seminar of Particle Physics Groups, July 13th, 2022

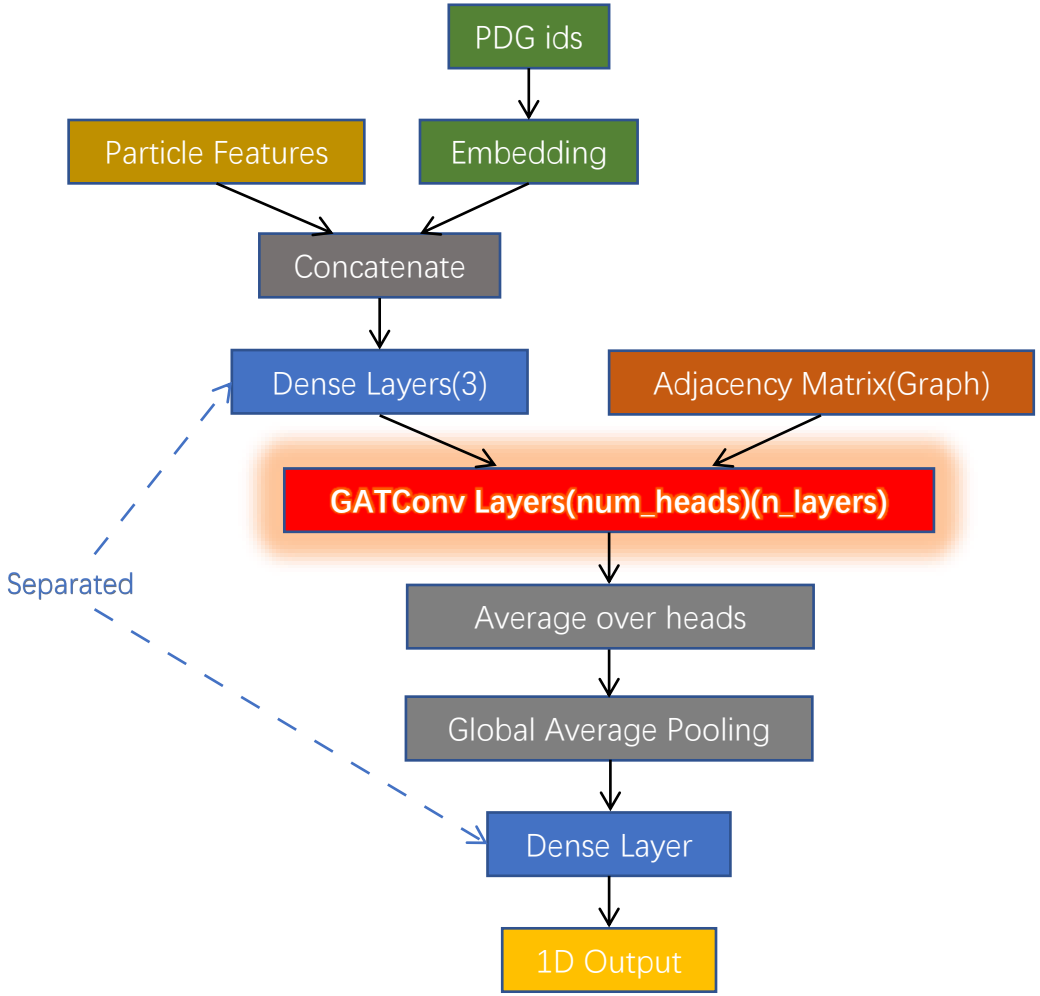
LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Backup

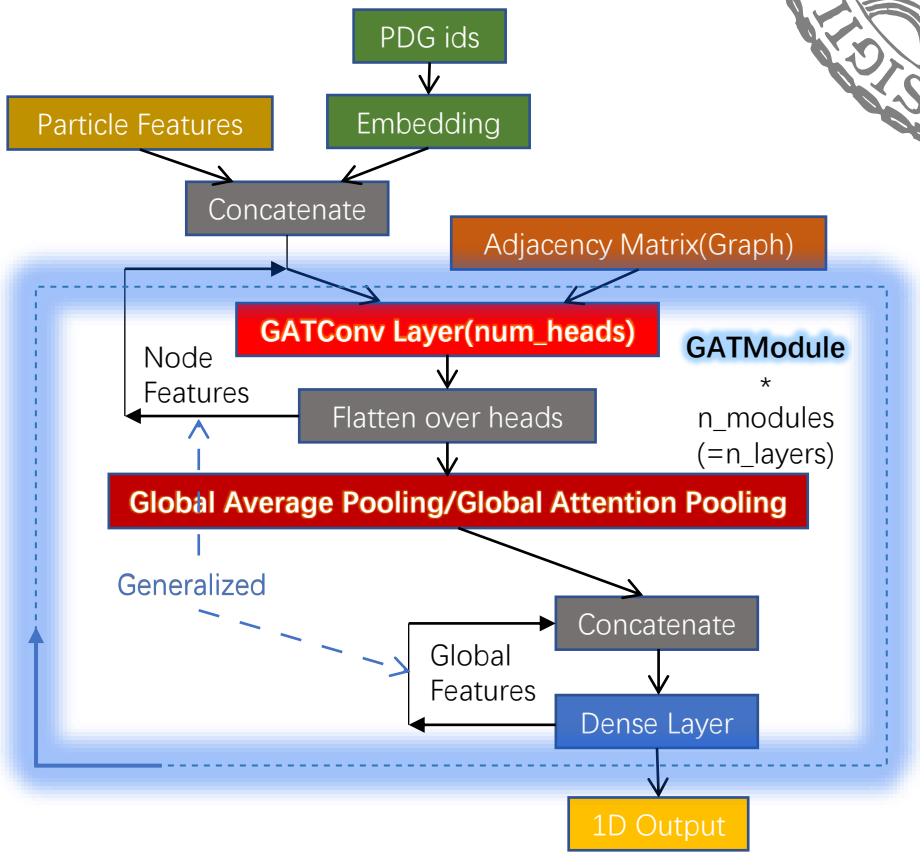
Backup: Network Structures



Backup: Network Structures

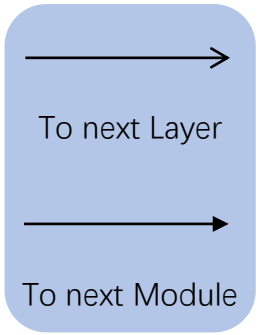


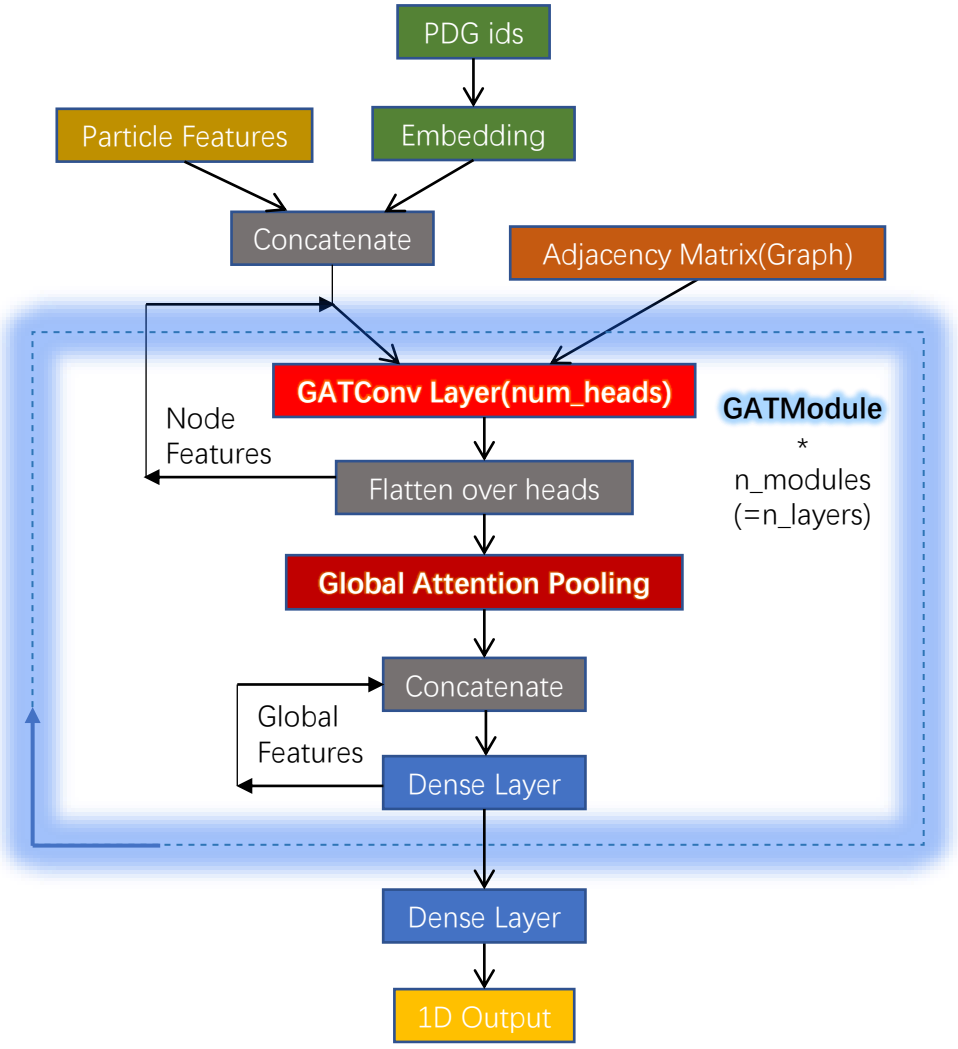
GAT(sep)



GAT(gen)
or
GAT+GAP(gen)

Final Architecture: GAT+GAP

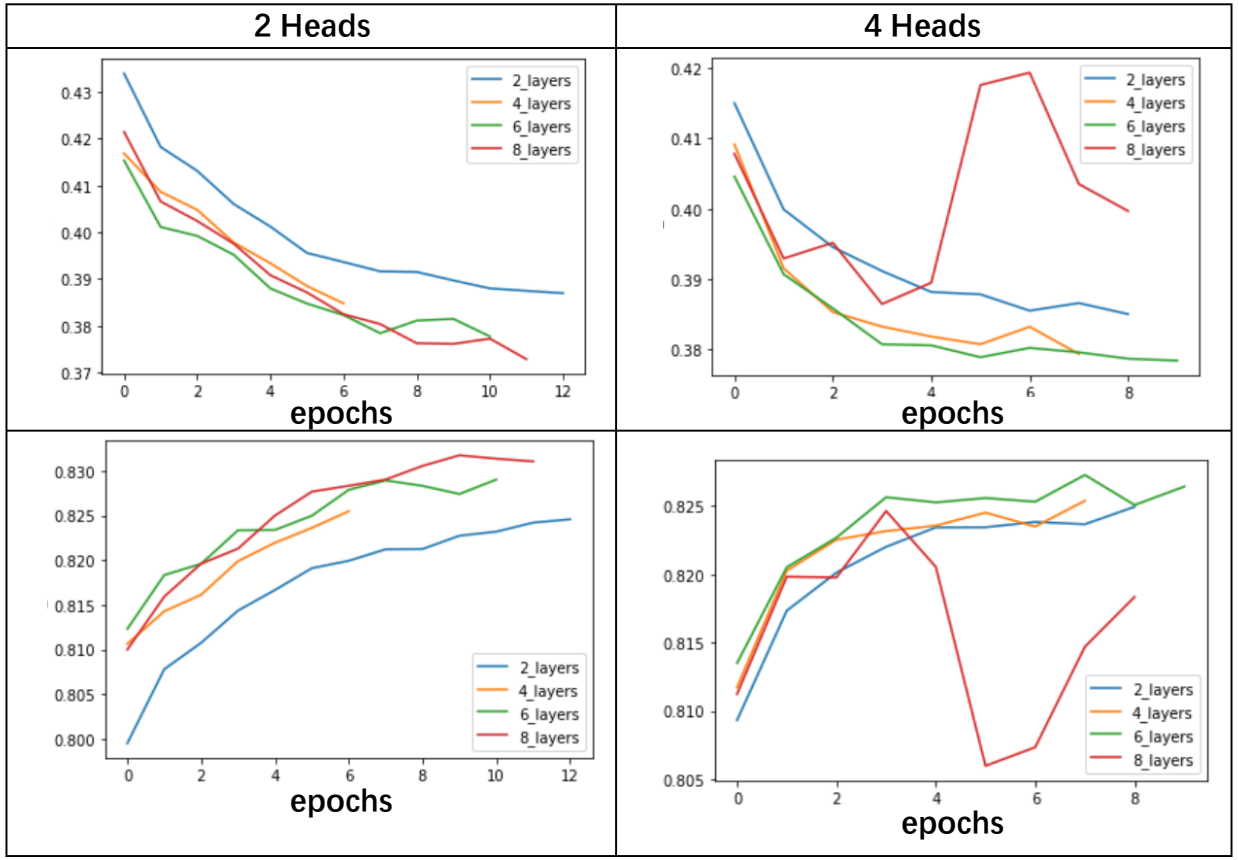


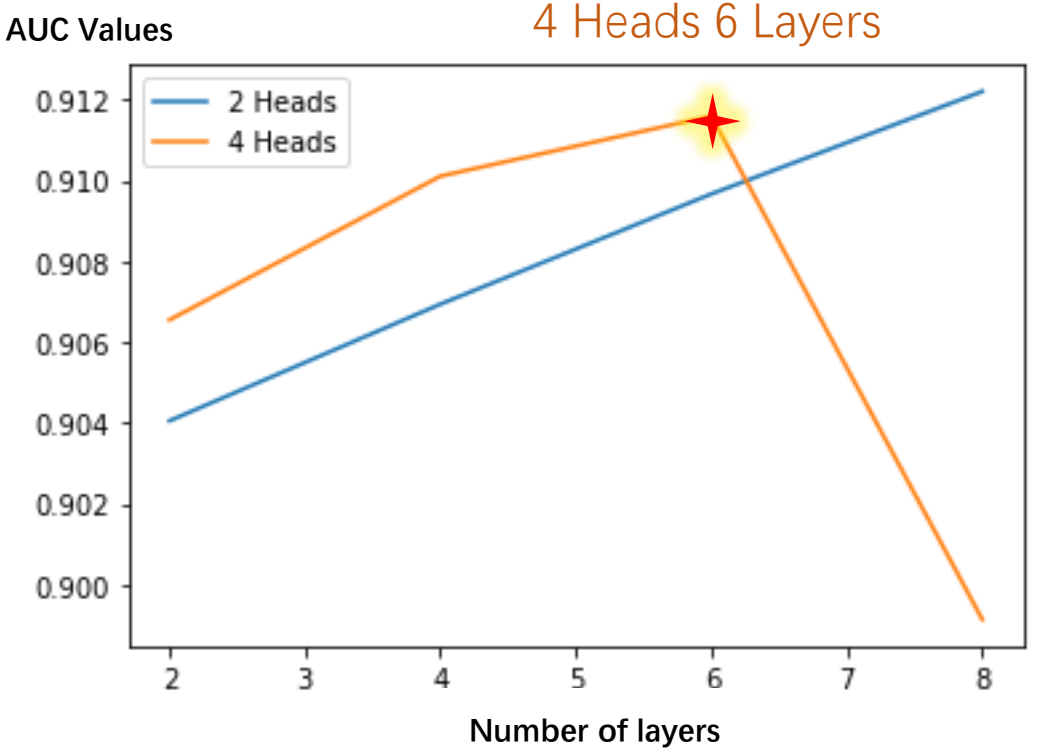


Improved Selective Background Monte Carlo Simulation at Belle II with Graph Attention Networks and Weighted Events

Backup: Network Structures

Quantitative Studies





Improved Selective Background Monte Carlo Simulation at Belle II with Graph Attention Networks and Weighted Events

Backup: Evaluations

Comparison

Parameters:

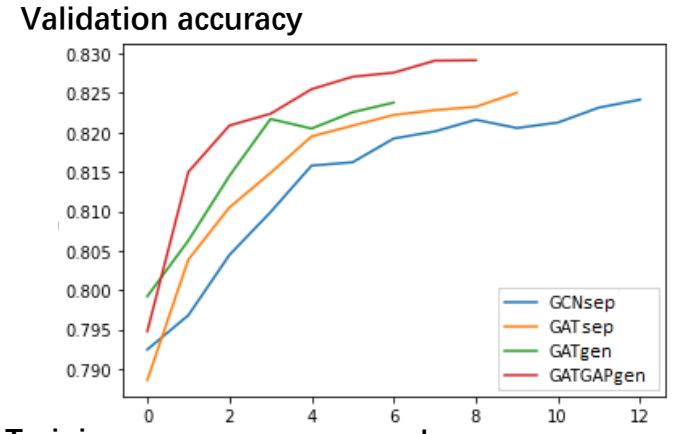
- n_heads = 4
- n_layers = 6
- n_units = 128
- batch_size = 128
- n_train = 0.9M
- n_val = 0.1M
- n_test = 0.5M

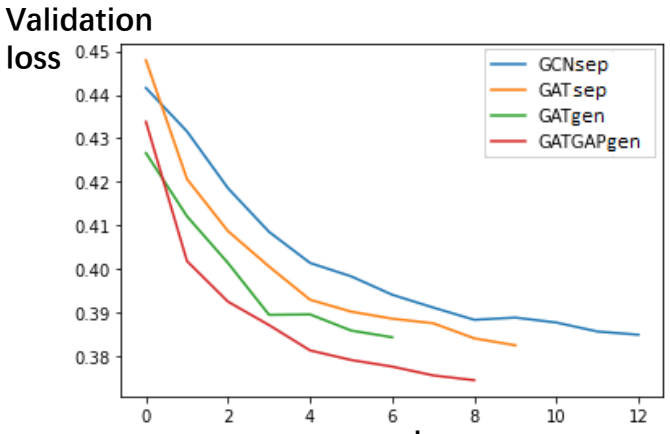
Loss:

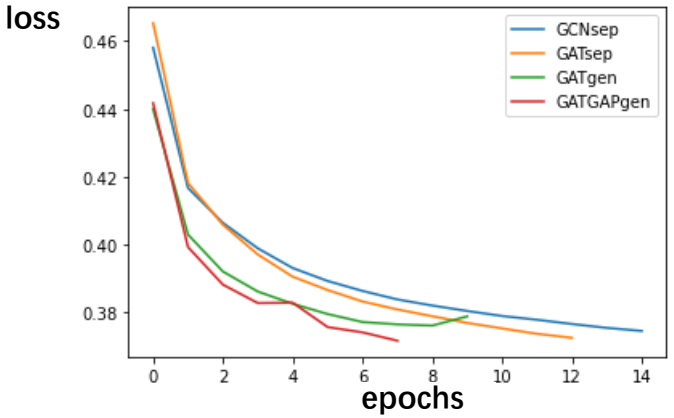
- Entropy

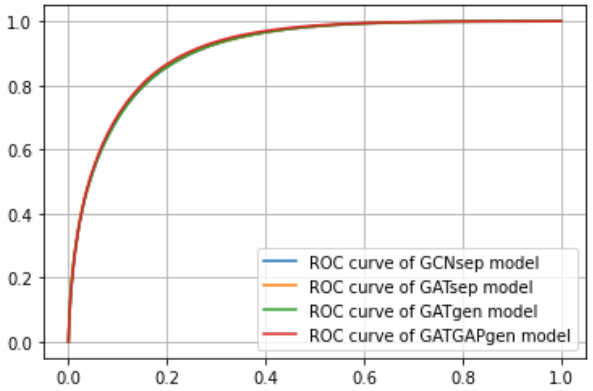
EarlyStopping:

- patience = 3
- delta = 1e-5









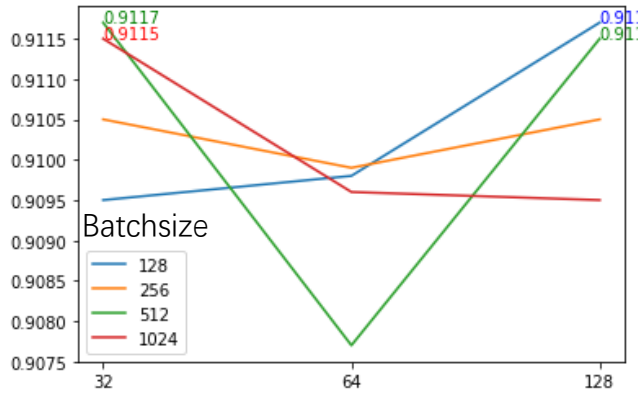
	GCN(sep)	GAT(sep)	GAT(gen)	GAT+GAP(gen)
TrainingTime	3619.46s	4047.47s	3471.48s	5049.81s
AUCValues	0.90831	0.90937	0.90891	0.91216

Improved Selective Background Monte Carlo Simulation at Belle II with Graph Attention Networks and Weighted Events

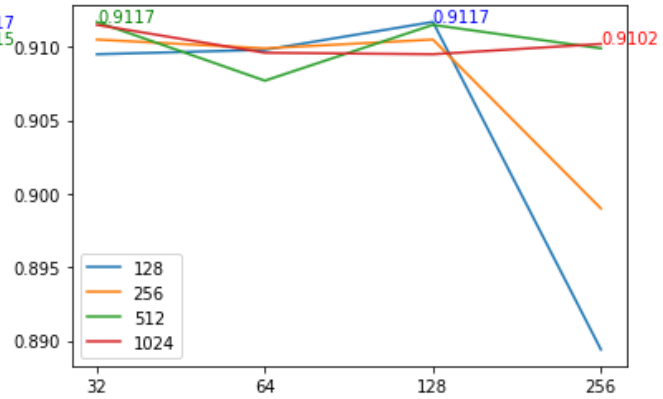
Backup: Evaluations

Grid Search

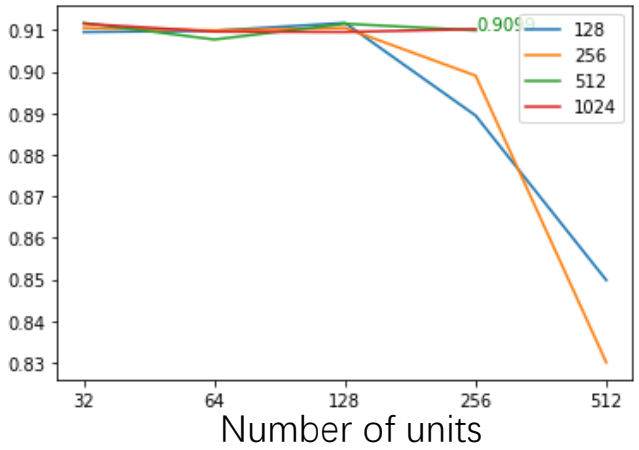
AUC



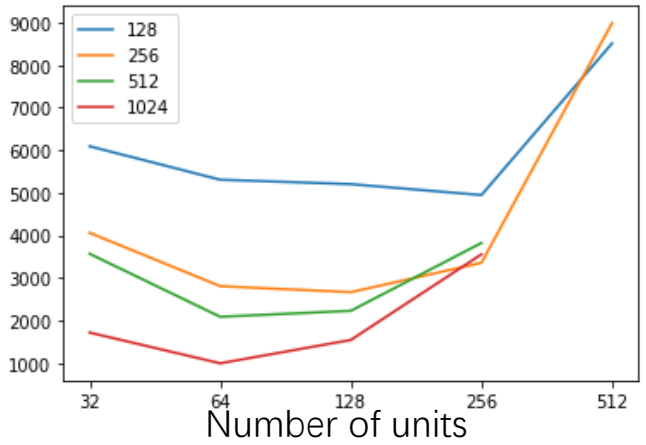
AUC



AUC



Training Time



Best Combinations

Batch-size	Number of units	AUC	Training Time
128	128	0.9117	5205
256	32	0.9105	4061
256	128	0.9105	2666
512	32	0.9117	3568
512	128	0.9115	2228
1024	32	0.9115	1716
1024	256	0.9102	3556

Network Sizes

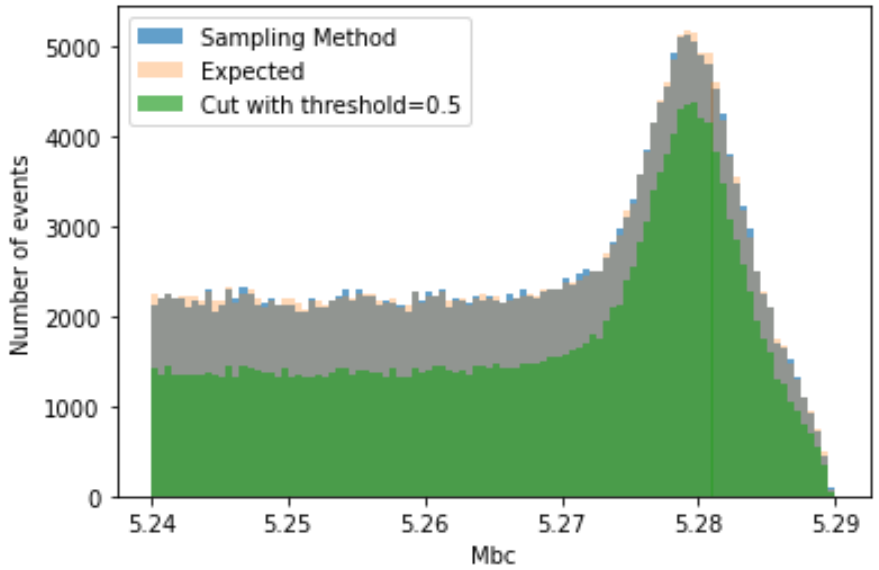
# Units	# Parameters
32	120,527
64	459,951
128	1,808,495
256	7,184,367
512	28,651,247

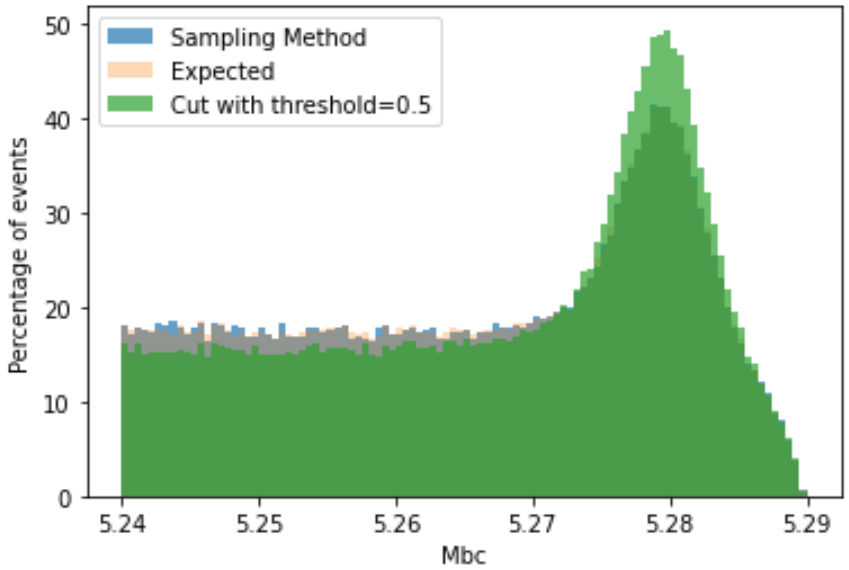
Improved Selective Background Monte Carlo Simulation at Belle II with Graph Attention Networks and Weighted Events

Sampling

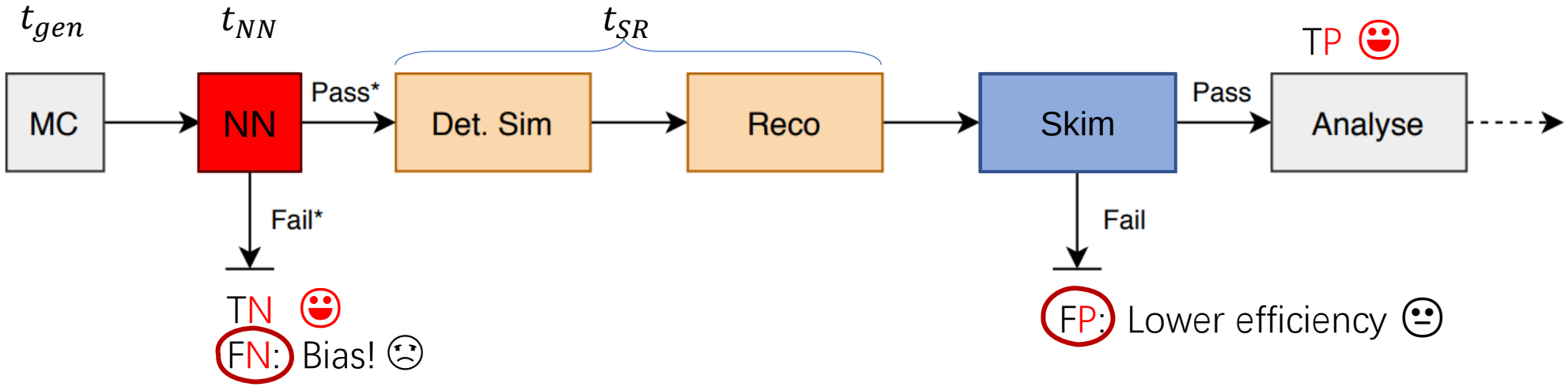
- Previous method: Cut according to neural network outputs
- Problem: Inevitable bias

- Our method: Sampling with probability given by neural network outputs
- Problem: Statistical uncertainty





Implementation of NN filters

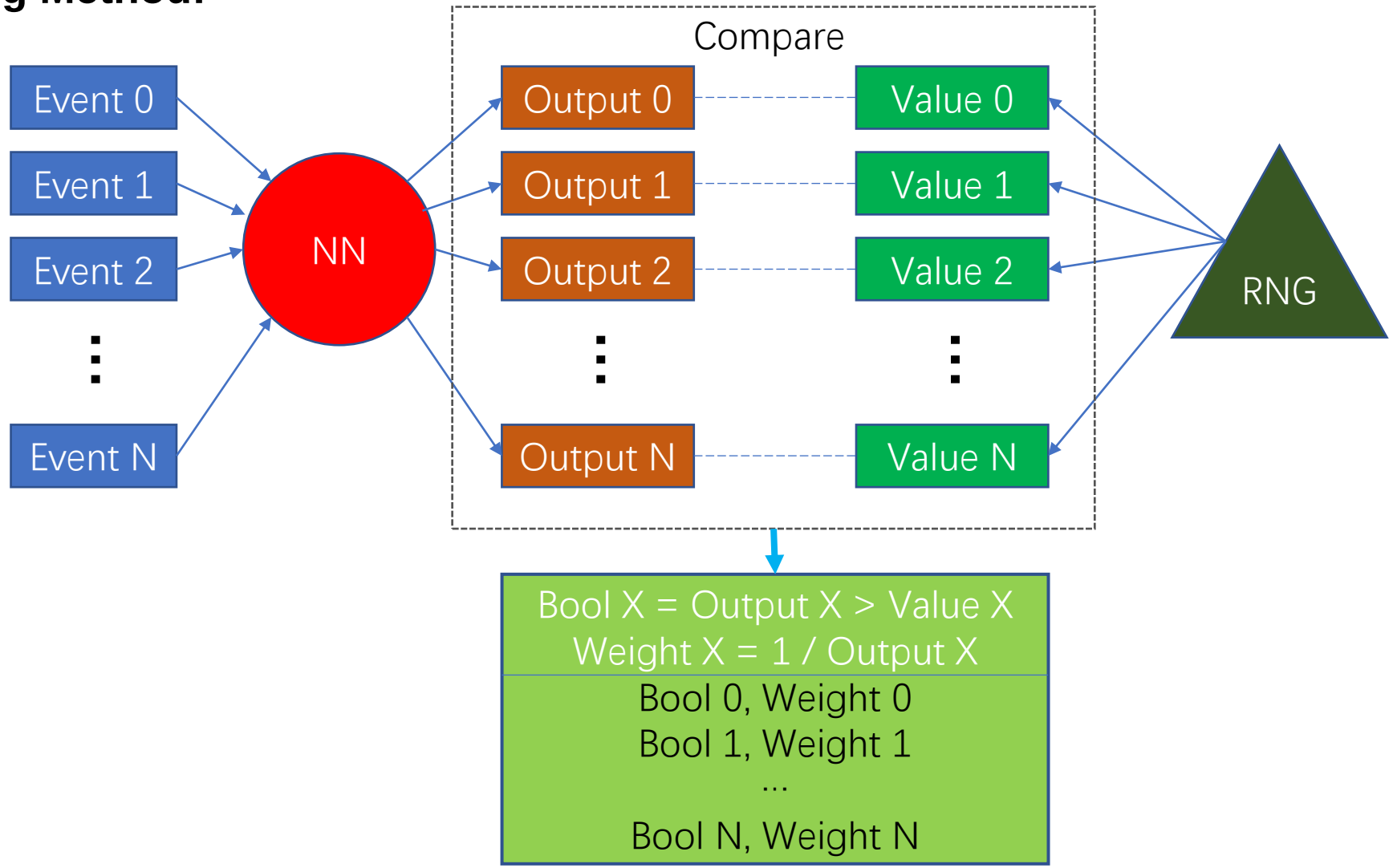


Skim \ NN	Positive	Negative
Pass	True-Positive (TP)	False-Negative (FN)
Fail	False-Positive (FP)	True-Negative (TN)

	t_{gen}	t_{NN}	t_{SR}	
TP	√	√	√	😀
FP	√	√	√	😐
FN	√	√		☹
TN	√	√		😀

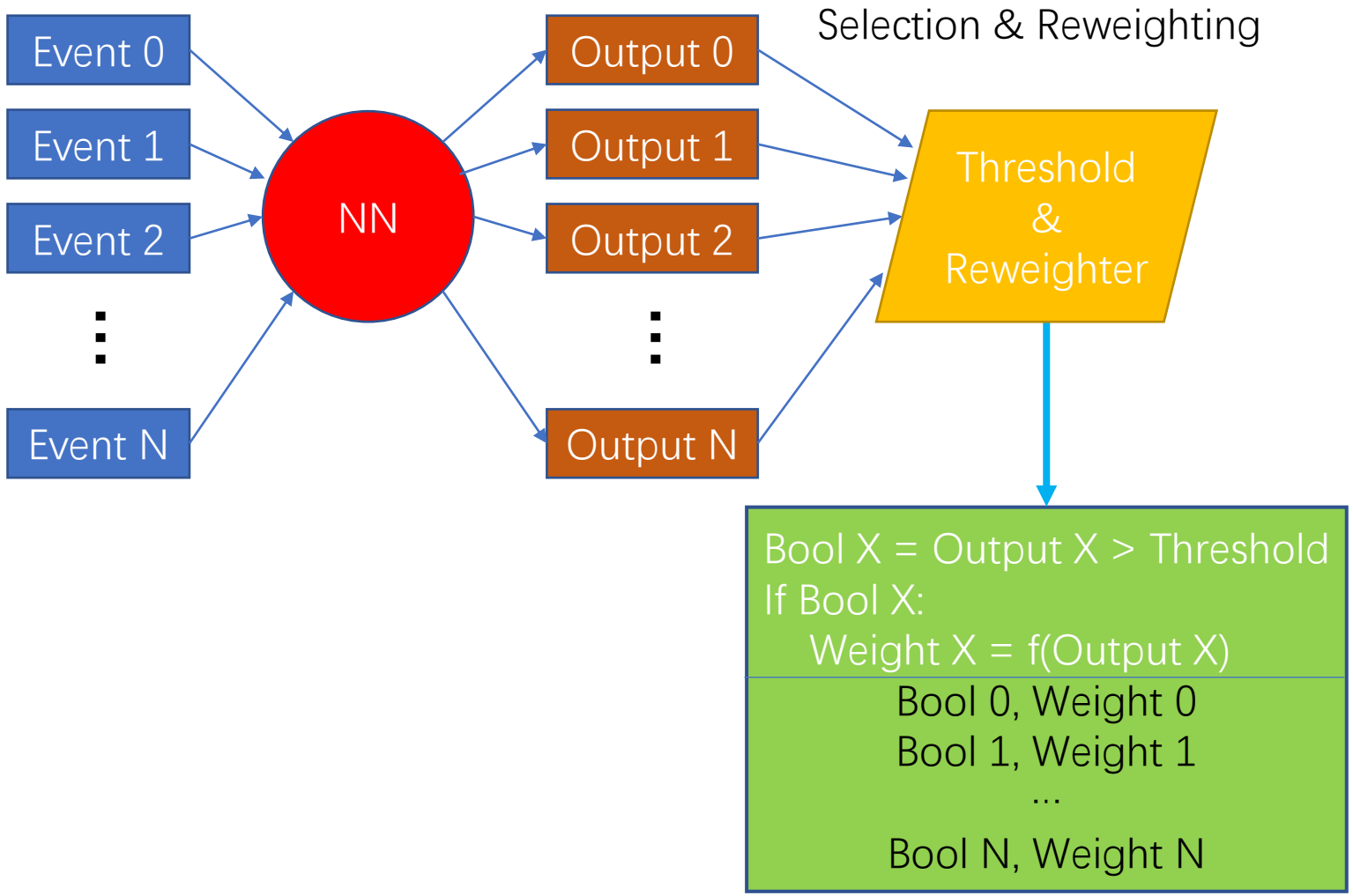
Event Weighting Methods

Sampling Method:



Event Weighting Methods

Reweighting Method:



Studied reweighters:

- GBDT Reweighting
- Histogram Reweighting

Reweighting Method:

- Train a Gradient Boosting Decision Tree (GBDT) classifier with some event level variables to distinguish between True-Positve events and False-Negative events
- GBDT Reweighting: use the outputs of the classifier directly:

$$w = \frac{1}{p_{clf}} = \frac{1}{p_{TP}/p_{TP+FN}} = \frac{p_{pass\_skim}}{p_{TP}}$$

- Histogram Reweighting: compare the score histogram of all the events that can pass the skim (True-Positive + False-Negative) with the score histogram of True-Positives to give each bin of score a scaling factor:

$$w = w_{bin_i}|_{p_{clf} \in bin_i} = \frac{H_{pass\_skim,i}}{H_{TP,i}}|_{p_{clf} \in bin_i}$$

Skim \ NN	Positive	Negative
Pass	True-Positive (TP)	False-Negative (FN)
Fail	False-Positive (FP)	True-Negative (TN)

Relative statistical uncertainty and effective sample size

Variable	Formula	Remark
NN outputs / Probabilities to pass	$\{p_i\}$	'i' refers to each event in the whole sample (batch)
Weights	$\{\omega_i\} = \left\{\frac{1}{p_i}\right\}$	Infinities (at $p_i = 0$) are excluded and set to 0 Avoid the bias by construction
Relative statistical uncertainty	$S = \frac{\sqrt{\sum \omega_i^2 p_i}}{\sum \omega_i p_i}$	$\sum \omega_i^2 p_i = \sum \omega_i$ $\sum \omega_i p_i = N$ Here consider only passed events (label = 1)
Effective sample size	$N_{eff} = \frac{1}{S^2}$	Number of events needed to reach **the same statistical uncertainty without sampling**

Speedup rate

Variable	Formula	Remark
Skim retention rate	$r = 0.05$	Probability to pass the skim process
Times of different phases in ms	$t_{gen} = 0.08$ $t_{NN} = 0.63$ $t_{SR} = 97.04$	Taken from previous studies
Effective number of events after sampling	$n_+ = \sum p_i$ $n_- = \sum (1 - p_i)$	$\{p_i\}$ will be devided into two subsets where the events will/won't pass the skim process
Time consuming with NN filter	$t_+ = [n_{TP} r + n_{FP}(1-r)](t_{gen} + t_{NN} + t_{SR})$ $t_- = [n_{FN} r + n_{TN}(1-r)](t_{gen} + t_{NN})$	Positive/Negative: Result of sampling True/False: Result of sampling == skim process
Time consuming without NN	$t_0 = N_{eff}(t_{gen} + t_{NN})$	To reach the same statistical uncertainty
(Inverse) Speedup rate	$R = \frac{t_+ + t_-}{t_0}$	The lower the better

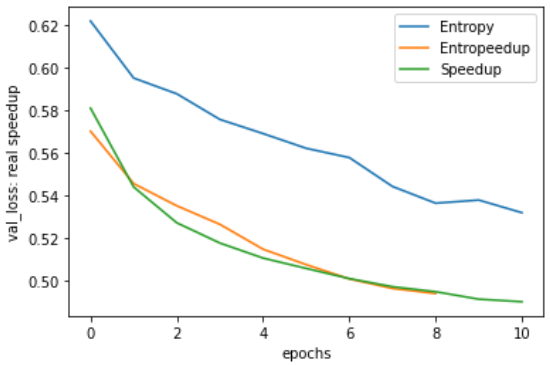
Backup: Sampling

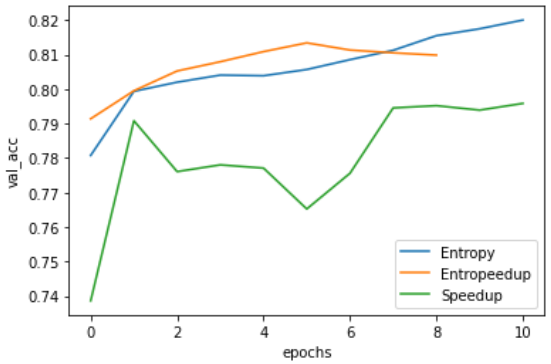
Performances of different training losses

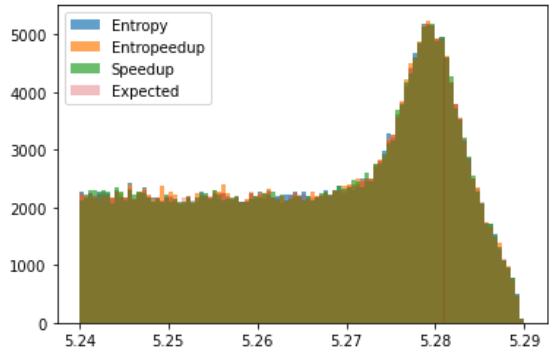
Speedup always as validation loss!

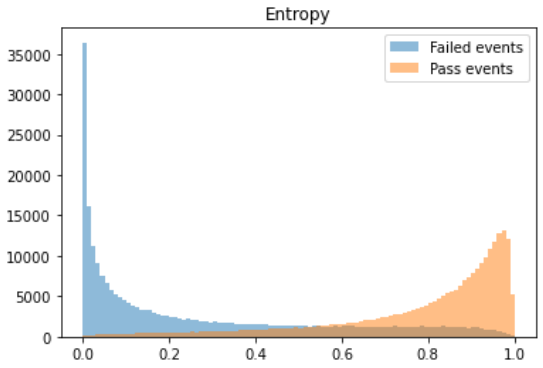
Entropy: Binary cross entropy between NN outputs $\{p_i\}$ and skim results $\{y_i \in \{0,1\}\}$
Speedup: (Inverse) Speedup rate R
Entropeedup: Training-epoch-dependent combination of Entropy and Speedup

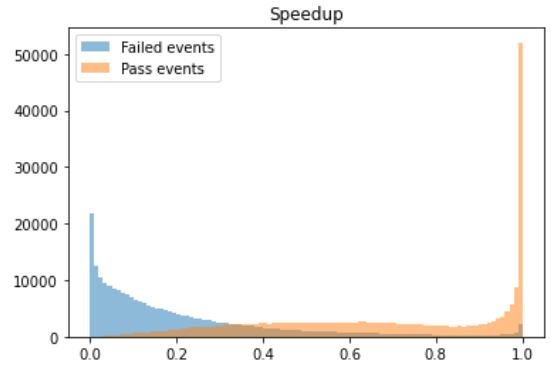
$$\text{Entropeedup} = e^{-epoch/10} Entropy + Speedup$$

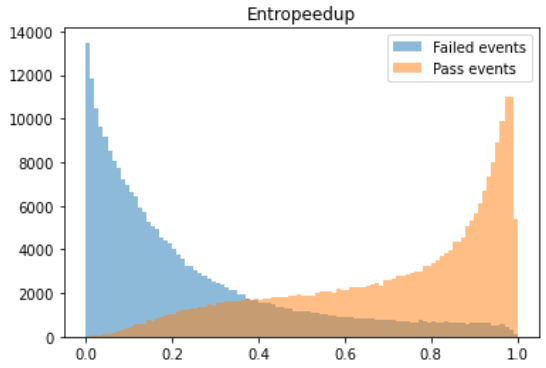












Improved Selective Background Monte Carlo Simulation at Belle II with Graph Attention Networks and Weighted Events

Event Weighting Methods

Metric: Speedup

--Improvement of computation time to produce the same effective number of events with the help of NN filter:

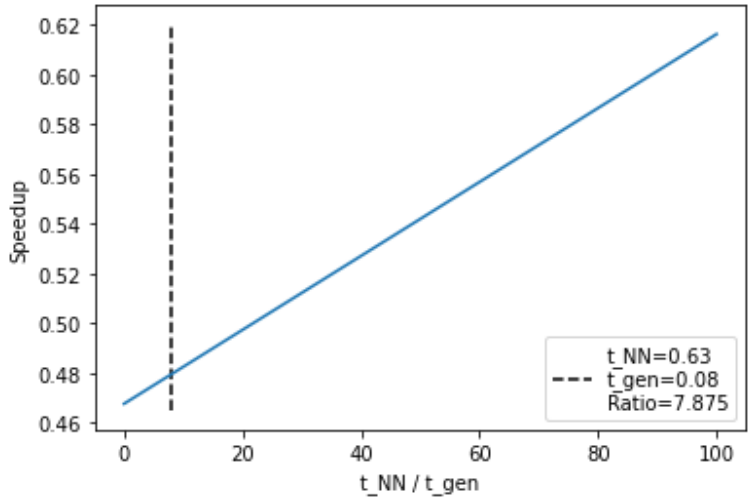
$$\text{Speedup} = \frac{t_{no\_filter}}{t_{filter}} \sim \text{Effective Sample Size} = \left(\frac{(\sum \omega_i p_i)^2}{\sum \omega_i^2 p_i}\right)$$

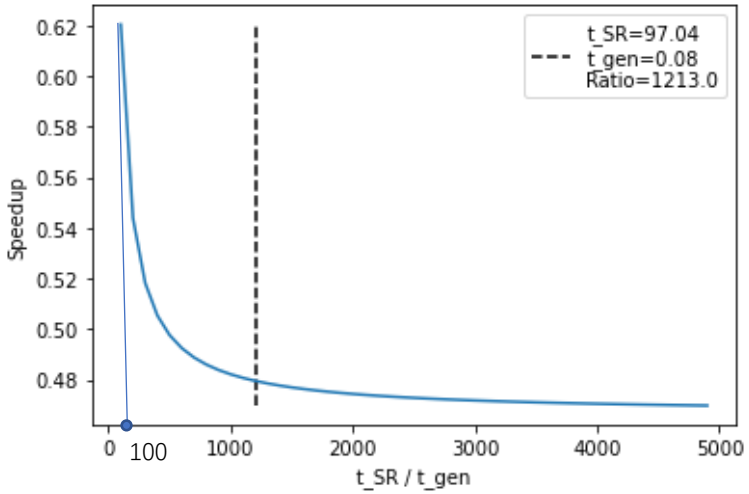
Robustness:

Weak dependency of **Speedup** on $\boldsymbol{t_{NN}}$ and $\boldsymbol{t_{SR}}$

↓

Safe to generalize

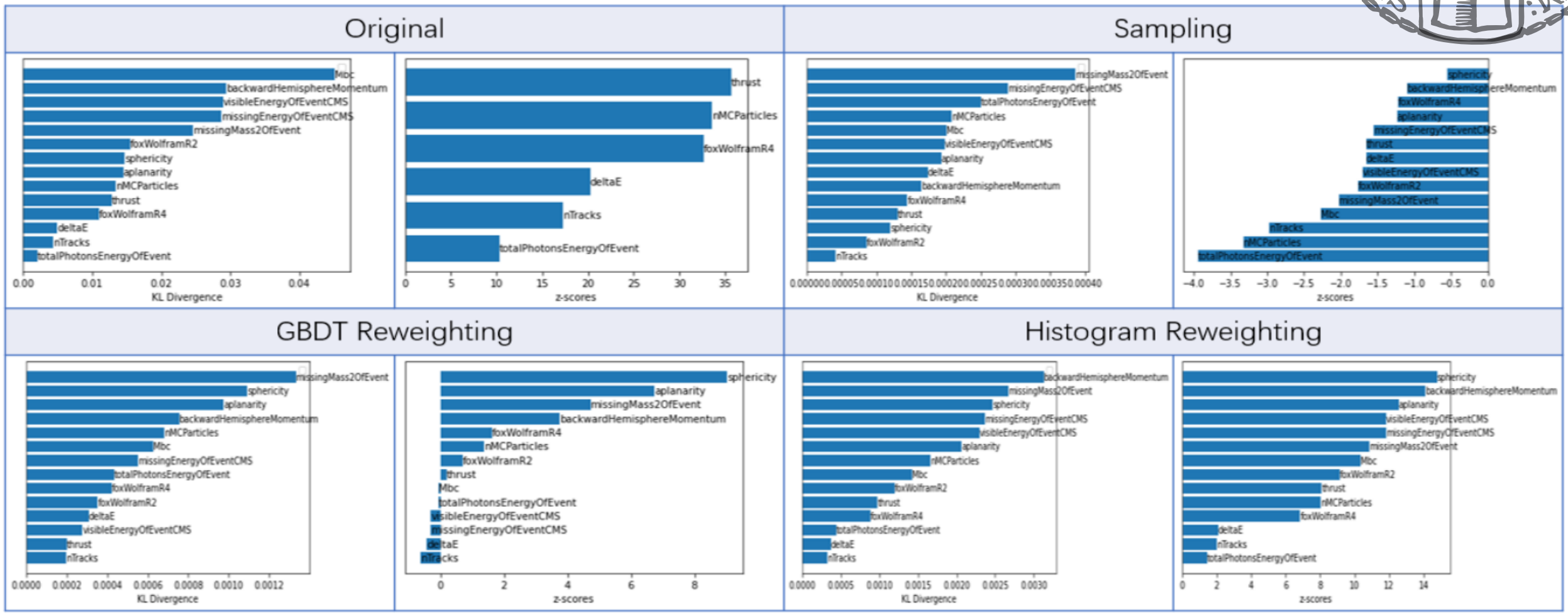


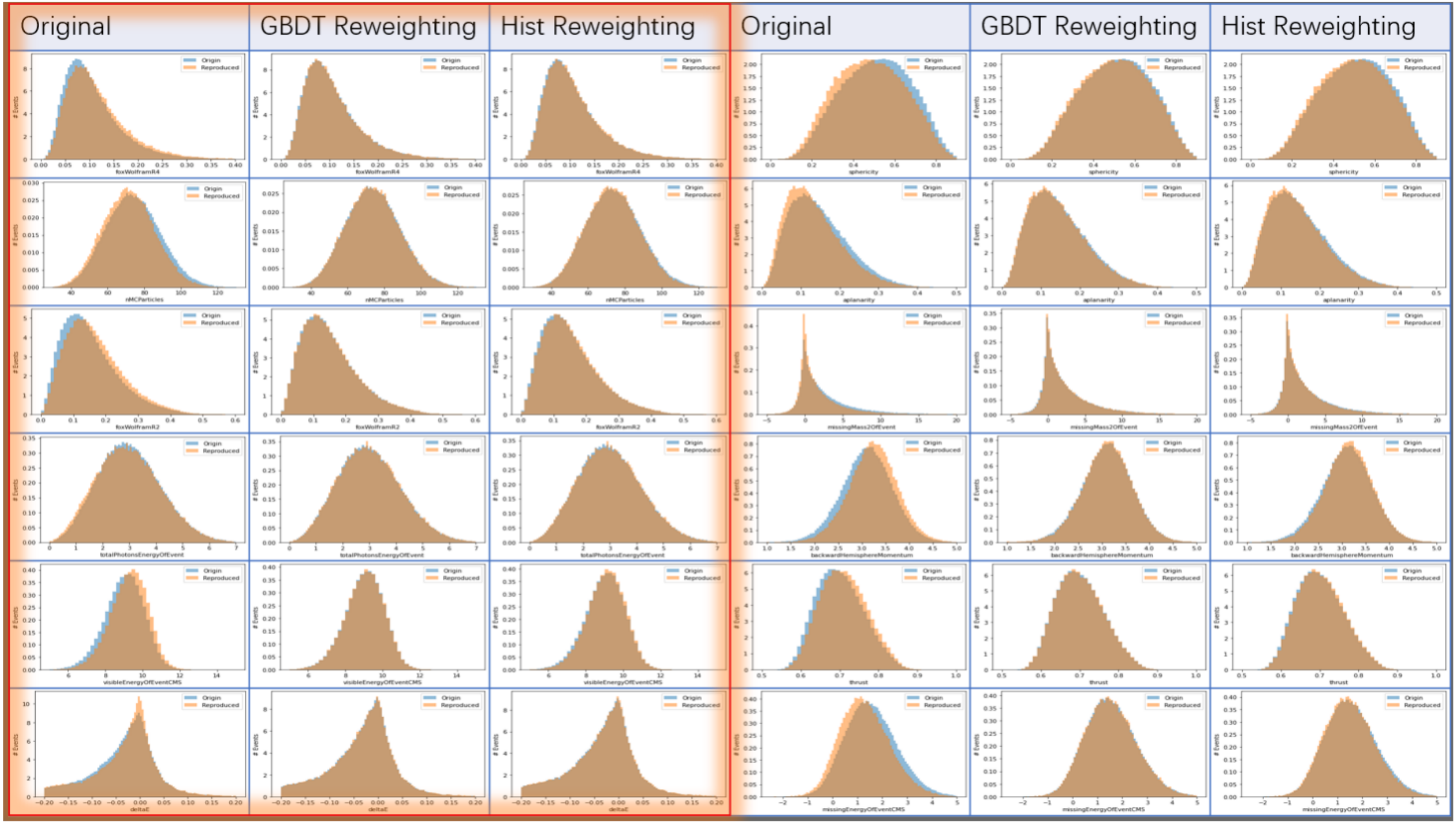


Improved Selective Background Monte Carlo Simulation at Belle II with Graph Attention Networks and Weighted Events

Backup: Event Weighting Methods

KS-Test




Original
GBDT Reweighting
Hist Reweighting
Original
GBDT Reweighting
Hist Reweighting
Origin
Reproduced
Events
foxWolframR4
nMCParticles
foxWolframR2
totalPhotonsEnergyOfEvent
visibleEnergyOfEventCMS
deltaE
sphericity
aplanarity
missingMass2OfEvent
backwardHemisphereMomentum
thrust
missingEnergyOfEventCMS