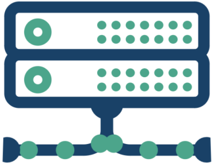
Particle Identification with Geometric Deep Learning

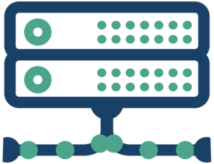
Anna Ershova
anna.ershova@llr.in2p3.fr

LLR, Ecole Polytechnique

28 November 2024

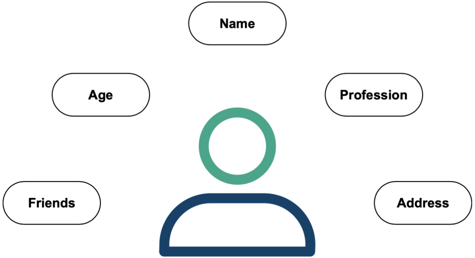


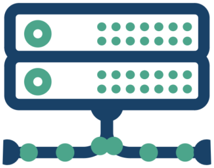
Data



Data

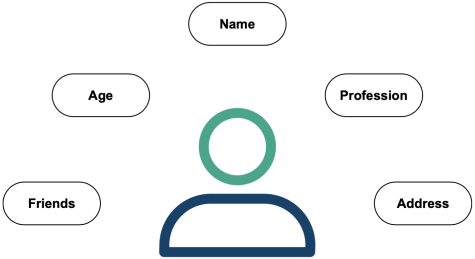
Features





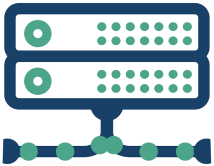
Data

Features



Label:
loan approval

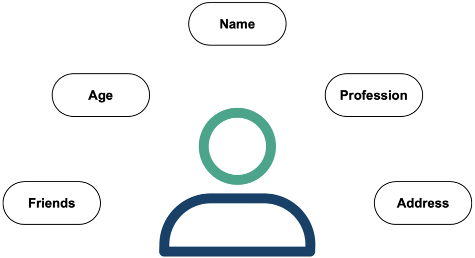
Supervised machine learning



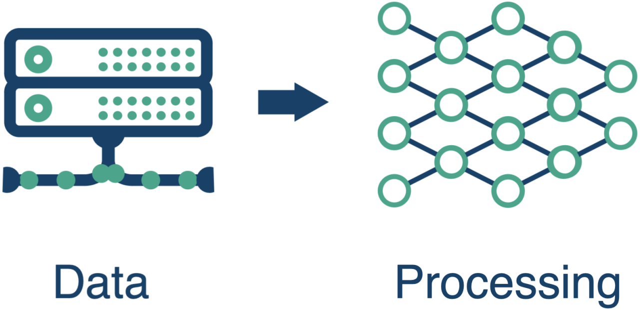
Data

Training examples:
$\{(x_1, y_1), ..., (x_N, y_N)\}$
x_i is a **feature vector** of the i-th example and y_i its **label**

Features

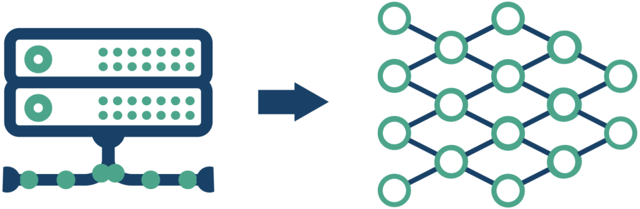


Label:
loan approval



Training examples:
$\{(x_1, y_1), ..., (x_N, y_N)\}$
x_i is a **feature vector** of the i-th example and y_i its **label**

Supervised machine learning

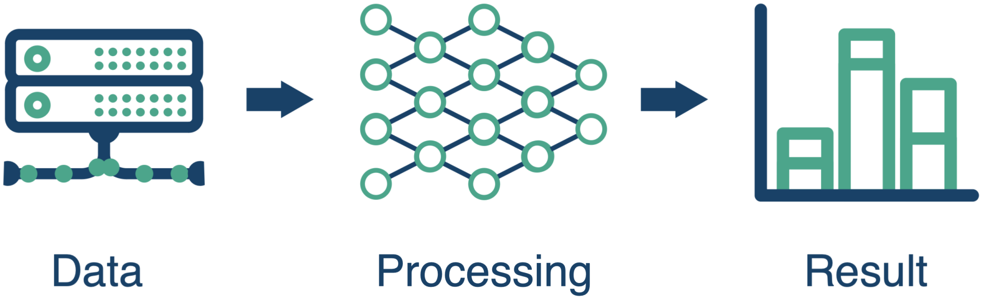


Data

Processing

Training examples:
$\{(x_1, y_1), ..., (x_N, y_N)\}$
x_i is a **feature vector** of the i-th example and y_i its **label**

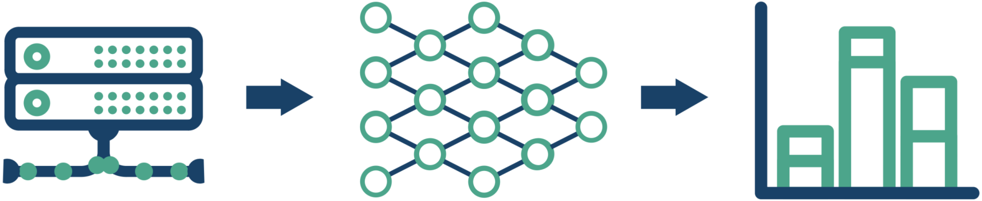
Search a function
$g : X \rightarrow Y$
prediction of Y based on X
Loss: $\frac{1}{N} \sum l(\hat{y}_i, y_i) \rightarrow min$



Training examples:
$\{(x_1, y_1), ..., (x_N, y_N)\}$
x_i is a **feature vector** of the i-th example and y_i its **label**

Search a function
$g : X \rightarrow Y$
prediction of Y based on X
Loss: $\frac{1}{N} \sum l(y_i, \hat{y}_i) \rightarrow min$

Supervised machine learning

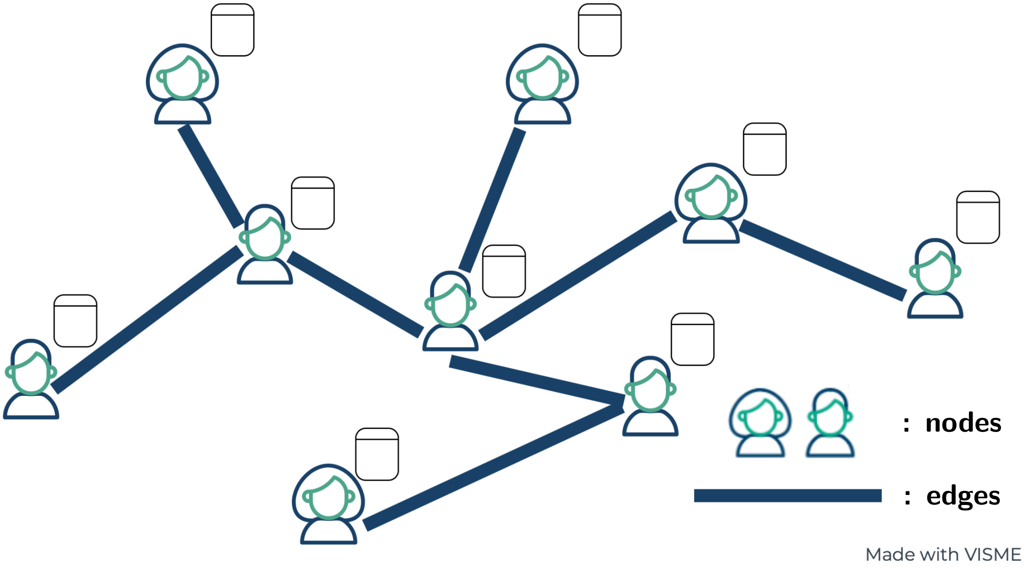


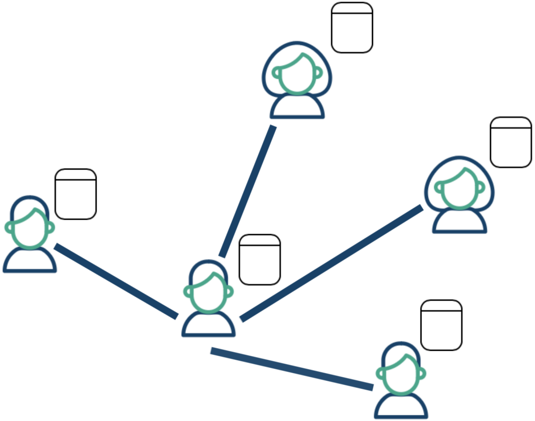
Data — Processing — Result

Training examples:
$\{(x_1, y_1), ..., (x_N, y_N)\}$
x_i is a **feature vector** of the i-th example and y_i its **label**

Search a function
$g : X \rightarrow Y$
prediction of Y based on X
Loss: $\frac{1}{N}\sum l(y_i, \hat{y}_i) \rightarrow min$

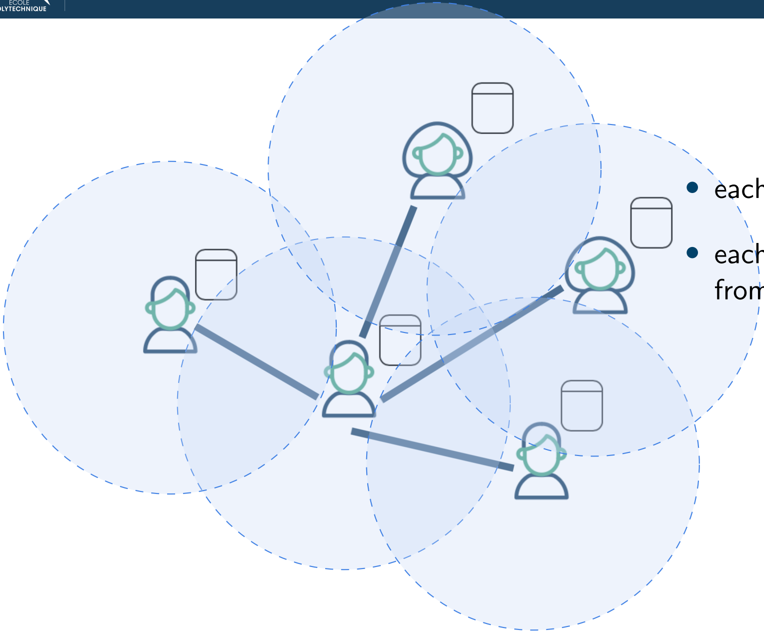
- **Prediction**: $\hat{y}_i = g(x_i)$ for new data
- Evaluate model according to the chosen **metrics**


: nodes
: edges
Made with VISME



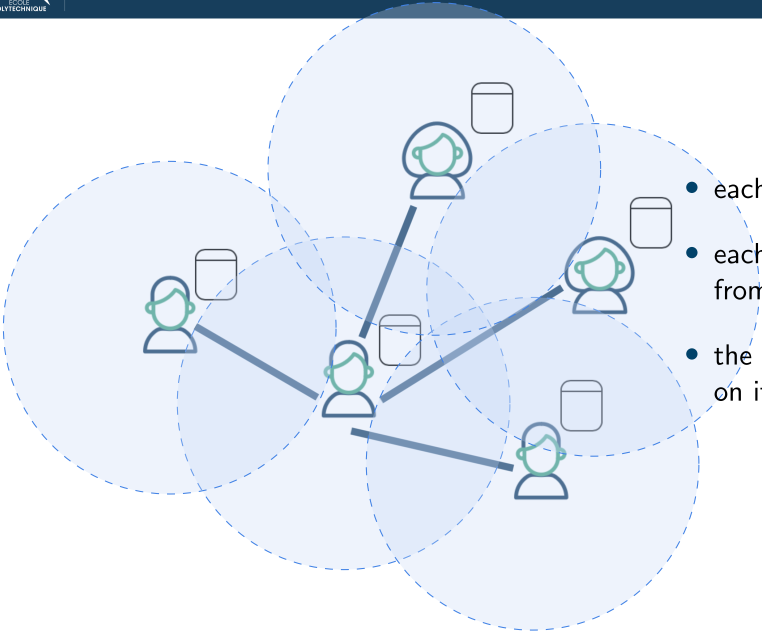
- each node has a feature vector

Graph message passing



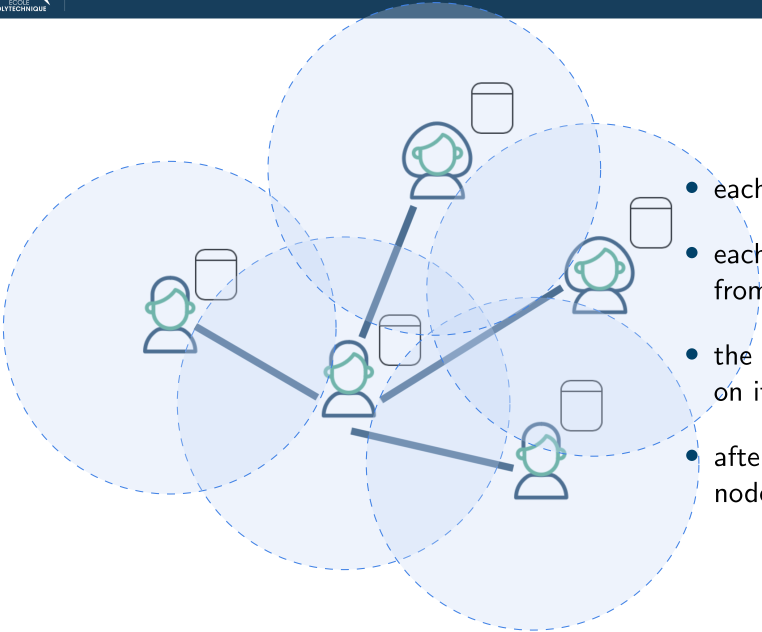
- each node has a feature vector
- each node aggregates information from its neighbors

Graph message passing



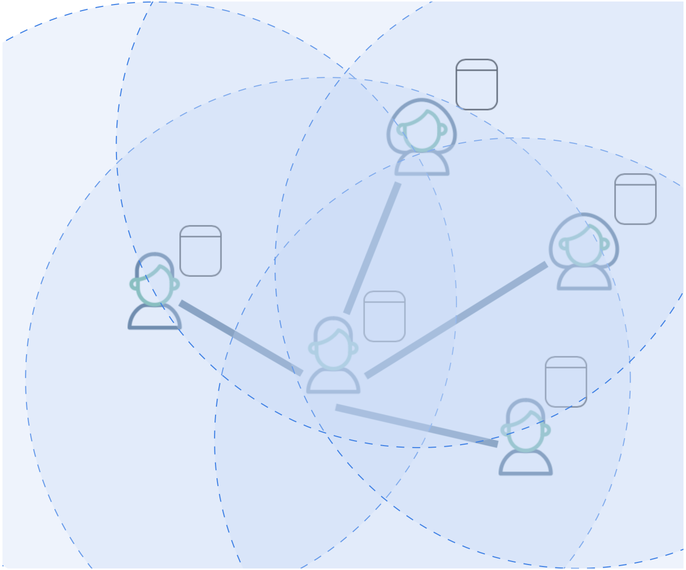
- each node has a feature vector
- each node aggregates information from its neighbors
- the node message is computed based on its feature vector

Graph message passing



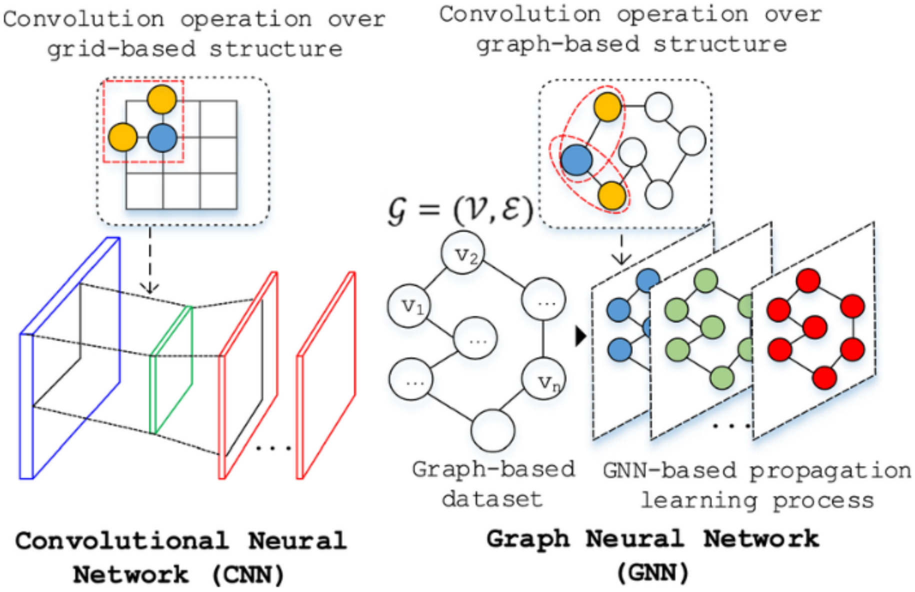
- each node has a feature vector
- each node aggregates information from its neighbors
- the node message is computed based on its feature vector
- after aggregating all messages, the node updates its feature vector

Graph message passing



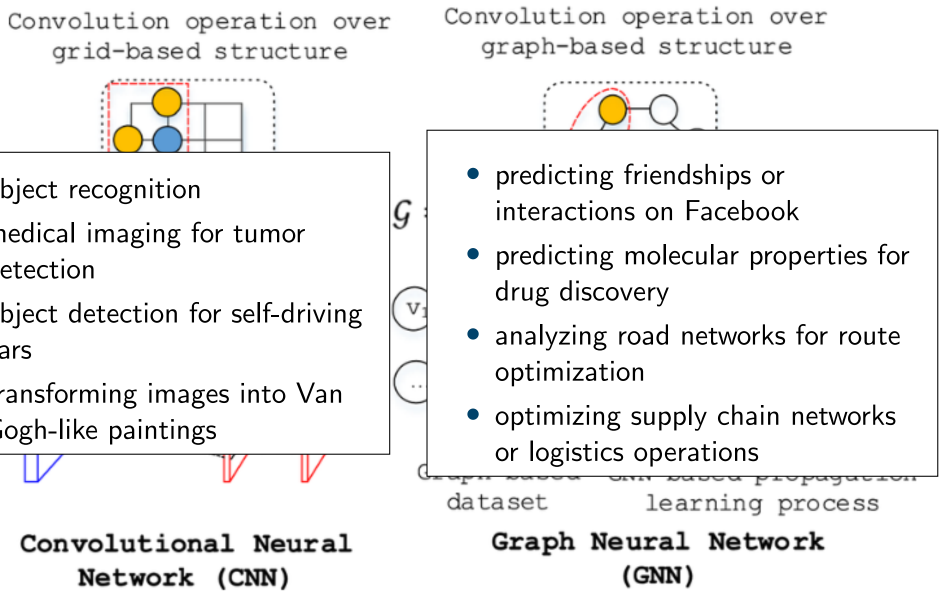
- each node has a feature vector
- each node aggregates information from its neighbors
- the node message is computed based on its feature vector
- after aggregating all messages, the node updates its feature vector
- After L layers, the node feature vectors incorporate information from nodes that are L hops away in the graph

GNN vs. CNN



P. Pham et. al., Artificial Intelligence Review

GNN vs. CNN

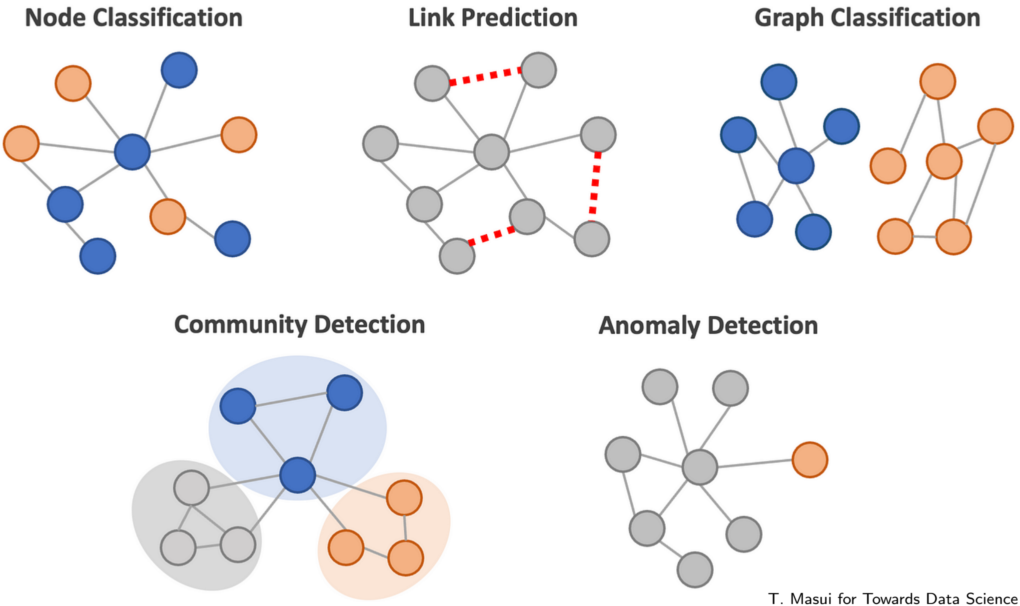


- object recognition
- medical imaging for tumor detection
- object detection for self-driving cars
- transforming images into Van Gogh-like paintings

- predicting friendships or interactions on Facebook
- predicting molecular properties for drug discovery
- analyzing road networks for route optimization
- optimizing supply chain networks or logistics operations

P. Pham et. al., Artificial Intelligence Review

Applications of GNNs



T. Masui for Towards Data Science

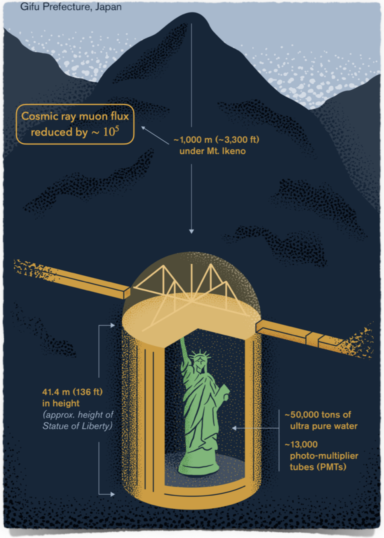
GRAph-based **N**eutron **T**agging
is coordinated in the context of **LLR/ILANCE** group effort
coordinator: Benjamin Quilain

aims of tagging neutrons from IBD interactions of DSNB electron antineutrinos in **Super-Kamiokande** and **Hyper-Kamiokande**: Antoine Beauchêne

- adaptation for atmospheric neutrinos in **Hyper-Kamiokande**: Christine Quach
- adaptation for atmospheric neutrinos in **Super-Kamiokande**: Christine Quach, Erwan le Blevec, and Mathieu Ferey
- low energy applications: neutron tagging in **WCTE**: Lorenzo Perisse
- adaptation for **WCTE**: Anna Ershova

Super-Kamiokande

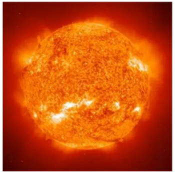
- Fiducial mass: 22.5 kton
- 11 129 PMTs in the Inner Detector
  - Diameter: $\sim 50\,\mathrm{cm}$
  - Time resolution: $\sim 3\,\mathrm{ns}$
  - Photocathode coverage: 40 %
- Since 2020 (SK phase VI): 0.02 % in mass of $Gd_2(SO_4)_3 \cdot 8H_2O$ ($\leftrightarrow$ Gd) was added to the tank
  $\Rightarrow$ Increase the neutron detection efficiency

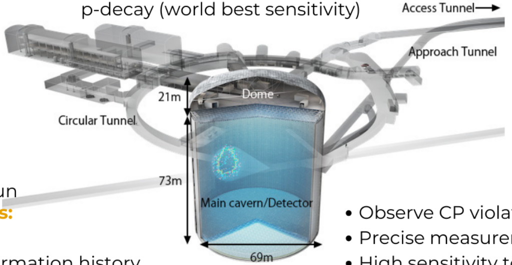


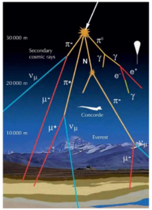
slide of Antoine Beauchêne

Proton decay

Probe Grand Unified Theories through p-decay (world best sensitivity)







- MSW effect in the Sun
- Non-standard interactions in the Sun

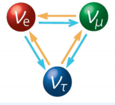
Supernovae neutrinos:

- Direct SNν: Constrains SN models
- Relic SNν: Constrains cosmic star formation history

- Observe CP violation for lepton at 5 σ
- Precise measurement of δCP
- High sensitivity to ν mass ordering

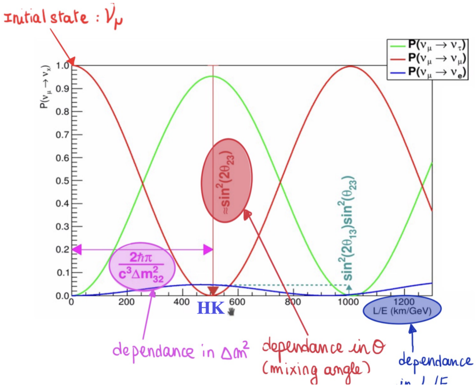
	SK	HK
Site	Mozumi	Tochibora
Overburden	2700 m.w.e.	1700 m.w.e.
Number of ID PMTs	11129	20000
Photo-coverage	40%	20% (x2 efficiency)
Mass/Fiducial mass	50 kton / 22.5 kton	258 kton / 186 kton
Beam power	500 kW to 1 MW	1.3 MW

$$P(\nu_\alpha \to \nu_\beta)(L,E) = \sin^2 2\Theta \sin^2\left(\frac{\Delta m^2 L}{4E}\right)$$



$$\approx \sin^2 2\Theta \sin^2\left(1.3\frac{\Delta m^2 L}{E}\right)$$

$$\Delta m^2 \equiv m_2^2 - m_1^2$$



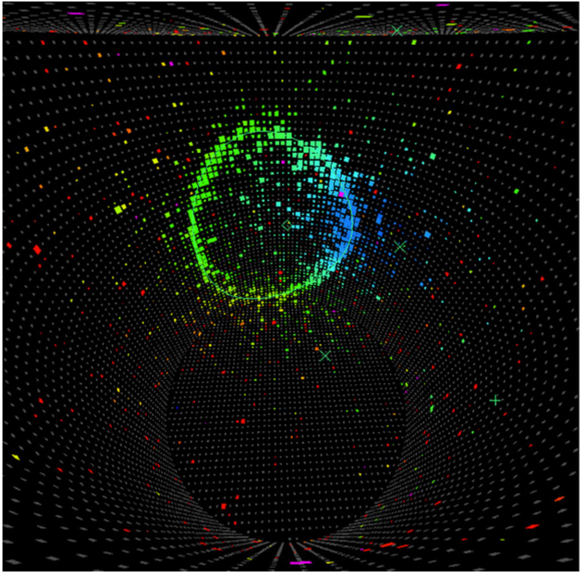
PMT output

- position X, Y, Z
- charge Q
- time t

Parameters to reconstruct

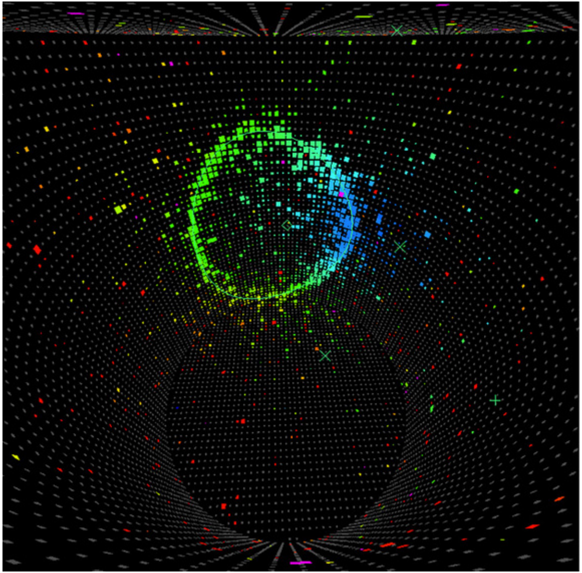
- flavor (PID)
- energy
- direction
- vertex

Input



- Number of triggered PMTs vary from event to event
- Information in the event:
  - position X, Y, Z
  - charge Q
  - time t
- Based on this information, we intend to identify the particle

Why not CNN?

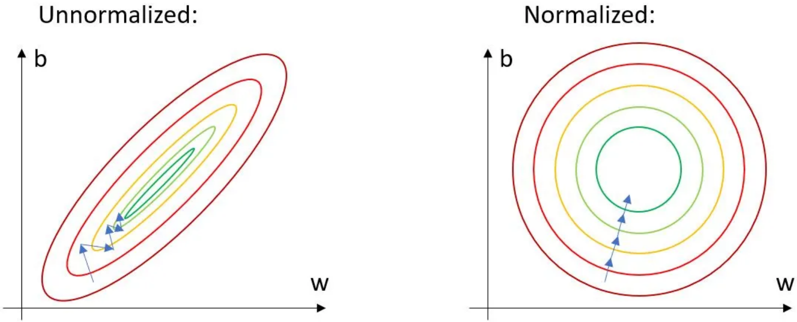


- **Irregular Geometry of the Detector**
  - PMTs form a non-uniform grid on a cylindrical surface: GNNs handle better irregular, non-Euclidean data structures
  - CNNs might have difficulties with handling 3D-data
- **Sparsity of the Signal**
  - only a small subset of PMTs is activated
  - GNNs process sparse signals by handling information across nodes (PMTs), CNNs require dense data grids
- **Relational Data**
  - GNNS capture relationships between PMTs through graph edges and message-passing

Preparing data

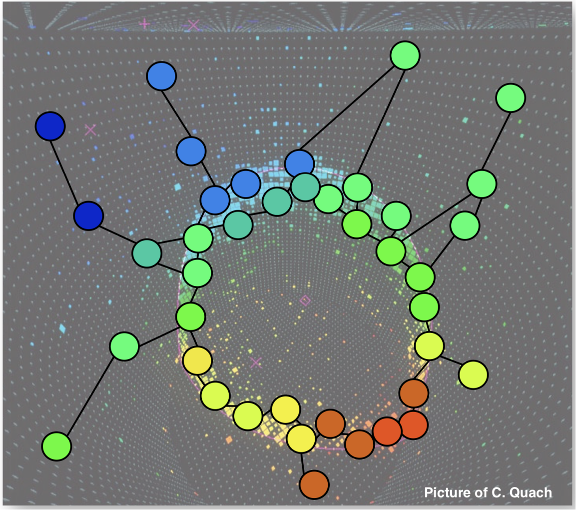
Normalization:

$$x_{norm} = \frac{x - min(x)}{max(x) - min(x)}$$



Data normalization ensures that all features **contribute equally** to the model, preventing **bias** toward larger values and **improving performance**.

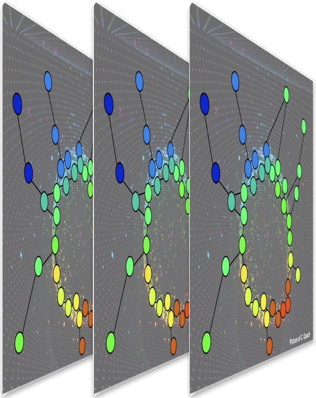
Building a graph



- One graph node is one PMT hit with features (X, Y, Z), Q, t
- Which nodes are close to each other?
  - based on position
  - based on charge and time
  - both
- How do we connect nodes?
  - too many $\rightarrow$ too much memory used
  - too little $\rightarrow$ we might loose information

These are the parameters to be optimized for our task

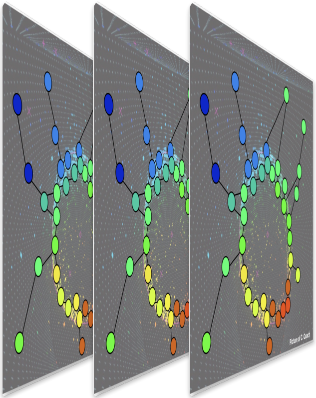
ResGatedGraphConv



arXiv

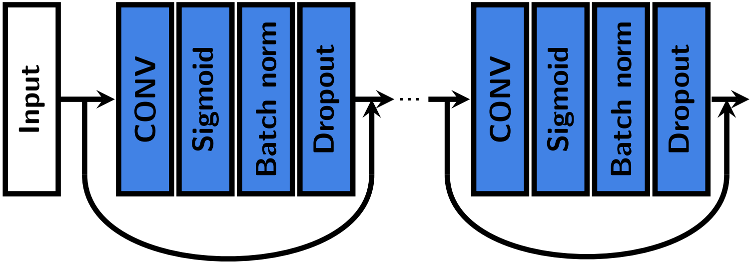
GRANT binary classification model: conv layers

ResGatedGraphConv



arXiv

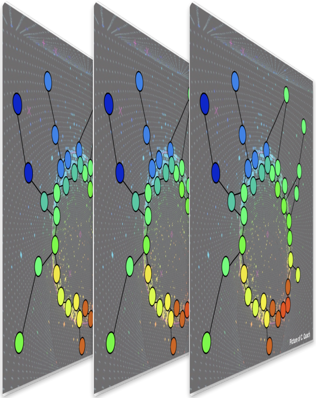
ResNet: arXiv



Skip connection technique:

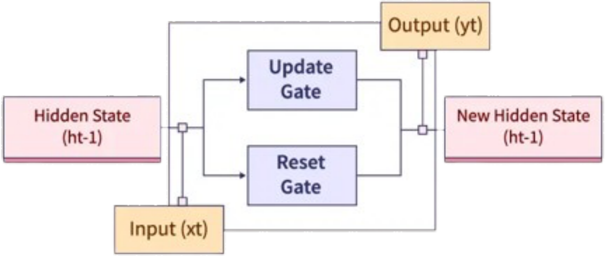
- addresses vanishing/exploding gradient issue $\rightarrow$ allows for deeper networks
- deeper networks can learn more complex features

ResGatedGraphConv



arXiv

Gated Recurrent Unit (GRU)

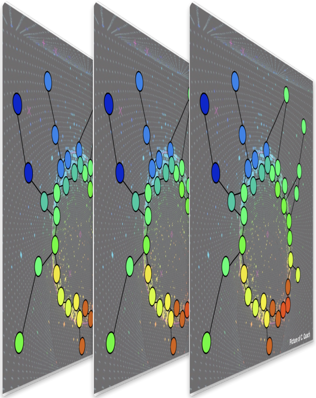


- **Reset gate:** how much of the previous state should be "forgotten" while computing the new state
- **Update gate:** controls the balance between the old information from the previous state and the new one from the new state
- **Applications:** sentiment analysis, machine translation, speech-to-text

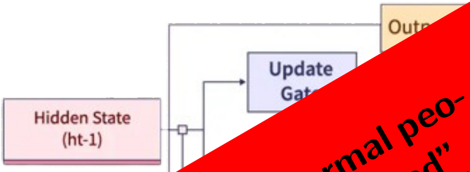
GRANT binary classification model: conv layers

Gated Recurrent Unit (GRU)

ResGatedGraphConv



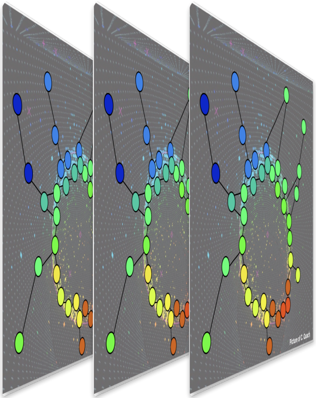
arXiv



This is what normal peo-ple mean by ” gated” but we are not them!

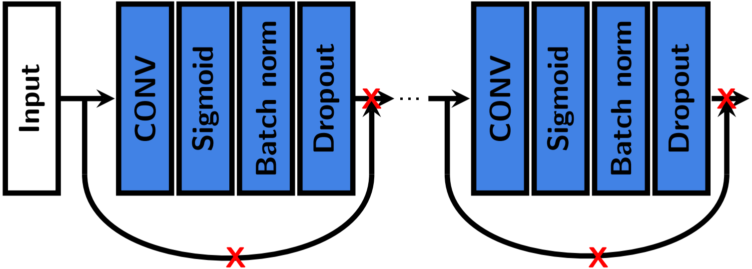
- **R** ... s state should be ... new state
- **U** ... balance between the old inf... previous state and the new one from the r...
- **Applic**...**ons:** sentiment analysis, machine translation, speech-to-text

ResGatedGraphConv



arXiv

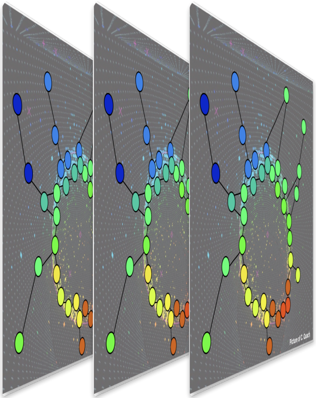
What does it mean for us:



The "gate" decides how much of the **residual** (input) information should be retained versus how much of the **newly computed** features should be added

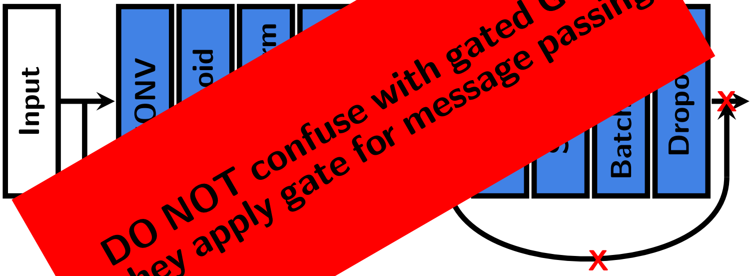
GRANT binary classification model: conv layers

ResGatedGraphConv



arXiv

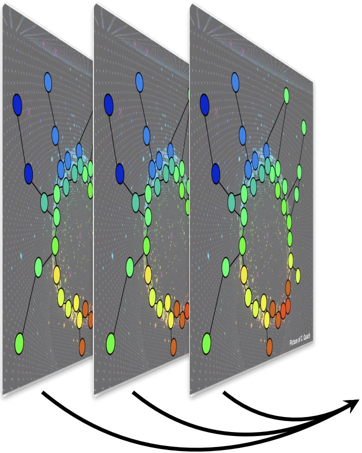
What does it mean f



DO NOT confuse with gated GNN: they apply gate for message passing

The "gate … how much of the **residual** (input) information should be re…ned versus how much of the **newly computed** features should be added

GRANT binary classification model: pooling



Mean and sum pooling

15	9	4	8
11	5	2	6
8	7	5	17
2	3	7	3

Sum pooling →

40	20
20	32

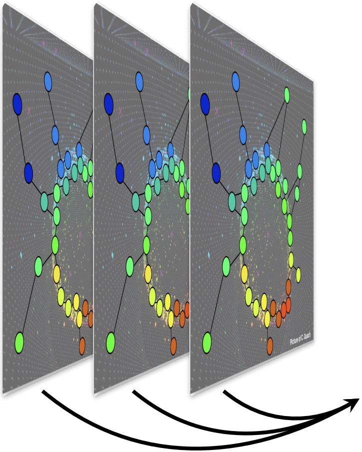
15	9	4	8
11	5	2	6
8	7	5	17
2	3	7	3

Mean pooling →

10	5
5	8

Classic pooling with 2x2 filter

GRANT binary classification model: pooling

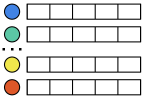


Mean and sum pooling

torch_geometric pooling for i-th graph:

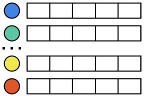
global add pool

$$r_i = \sum_{n=1}^{N_{\text{nodes}}}$$



global mean pool

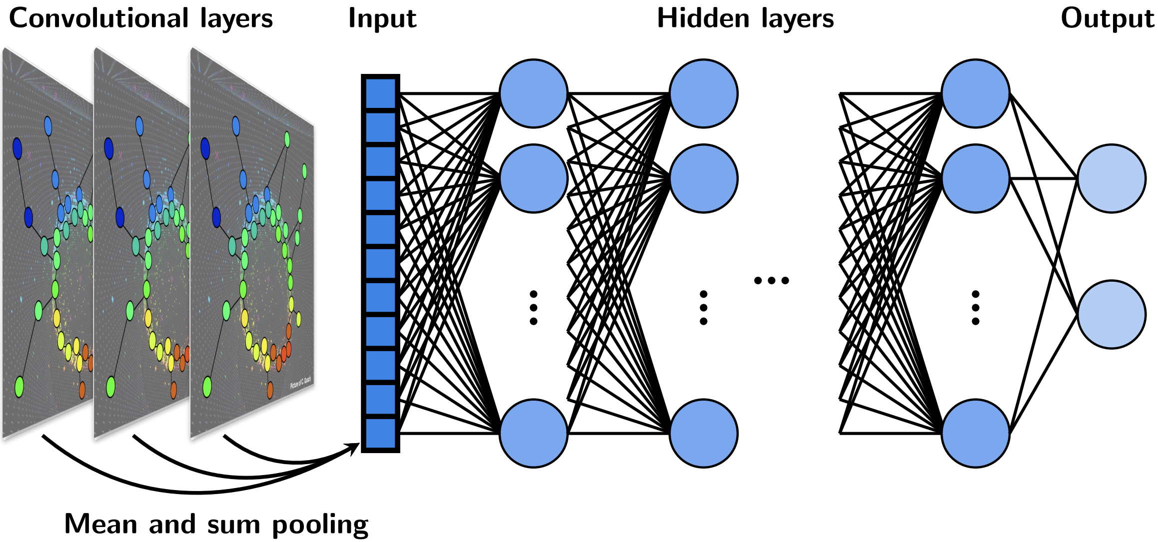
$$r_i = \frac{1}{N_i} \sum_{n=1}^{N_{\text{nodes}}}$$



If we have num_graphs graphs:

$$(\text{num\_nodes}, \text{num\_features}) \xrightarrow{\text{pool}} (\text{num\_graphs}, \text{num\_features})$$

GRANT binary classification model: visualization



Other hyperparameters

Number of convolutional and hidden layers, number of neurons, the way we build a graph...

What else could we do?

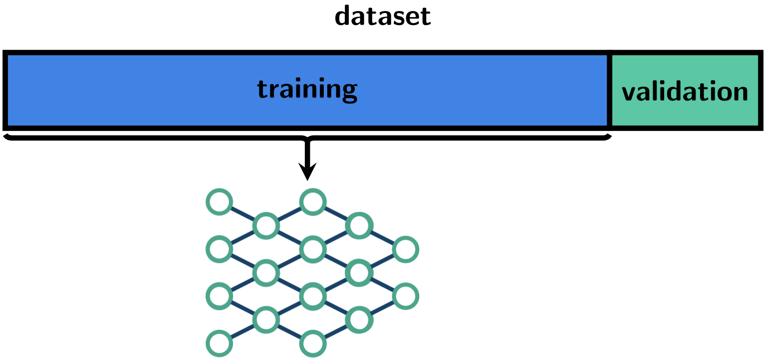
Other hyperparameters: epoch

dataset

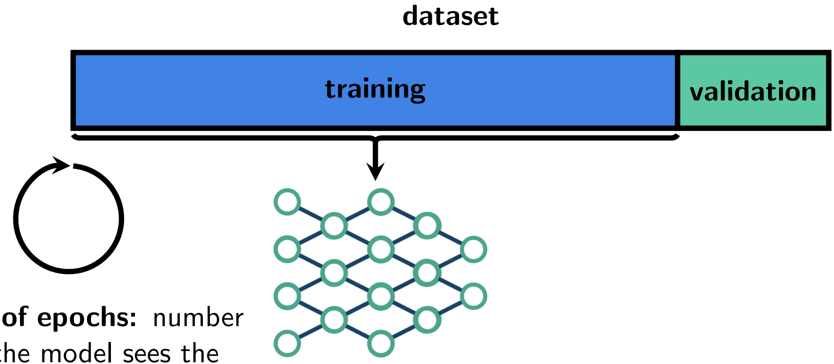
Other hyperparameters: epoch

dataset

Other hyperparameters: epoch

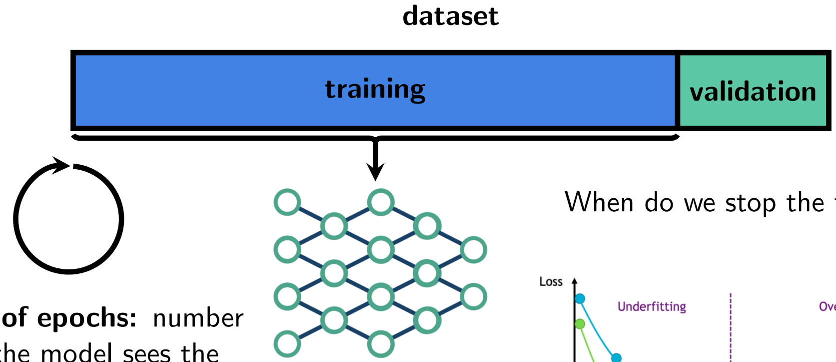


Other hyperparameters: epoch



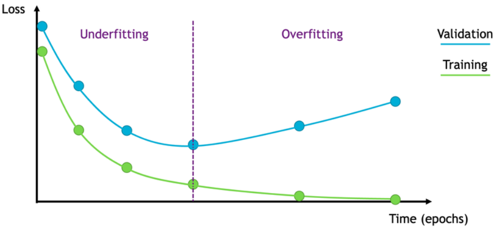
Number of epochs: number of times the model sees the entire traing set

Other hyperparameters: epoch



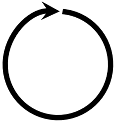
Number of epochs: number of times the model sees the entire traing set

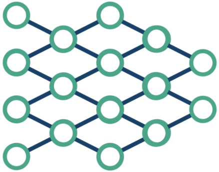
When do we stop the training process?



Other hyperparameters: epoch

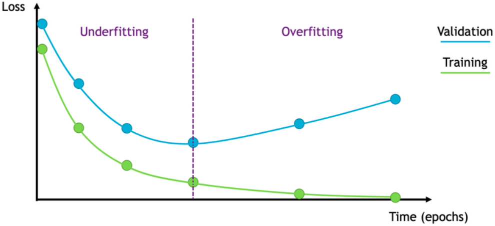
dataset





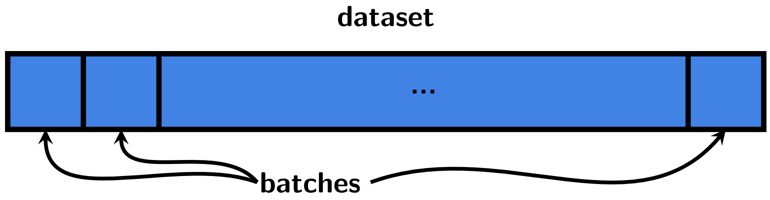
Number of epochs: number of times the model sees the entire traing set

When do we stop the training process?

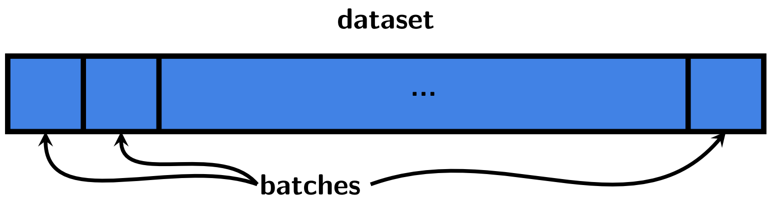


We can play with **number** of epochs and the **stopping condition**

Other hyperparameters: batch size

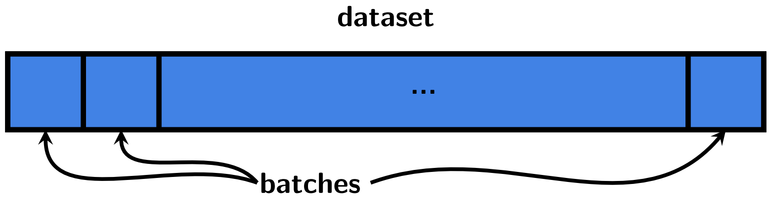


Other hyperparameters: batch size



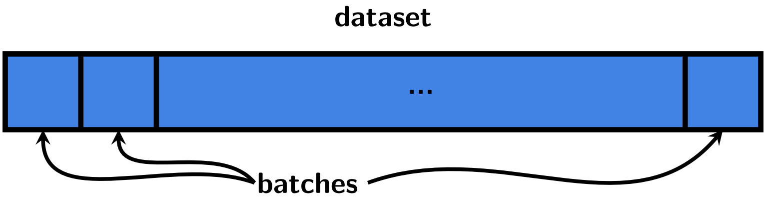
Batch: number of samples that will be simultaniously propagated through the network; based on these samples the weights of the model are updated. Update is based on the **gradient descent**

Other hyperparameters: batch size



Batch: number of samples that will be simultaneously propagated through the network; based on these samples the weights of the model are updated. Update is based on the **gradient descent**

- \+ requires less memory
- +- trains slower but *can* converge faster
- \- gradient descent may be less accurate
- ! for larger batches the significant degradation of the model is observed (ability to generalize)



Batch: number of samples that will be simultaneously propagated through the network; based on these samples the weights of the model are updated. Update is based on the **gradient descent**

- \+ requires less memory
- +- trains slower but *can* converge faster
- \- gradient descent may be less accurate
- ! for larger batches the significant degradation of the model is observed (ability to generalize)

$L(x, w)$ — loss function, x — training set, w — weights to be optimized

Goal: to minimize $L(x, w)$. Gradient descent update:

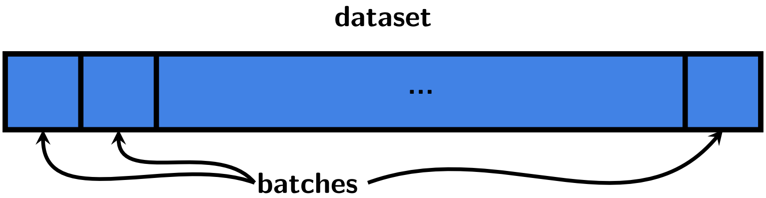
$w_{t+1} = w_t - \lambda \nabla_w L(x, w)$

w_t: represents the current weights at iteration t

$\nabla_w L(x, w)$: is the gradient of the loss function with respect to the weights

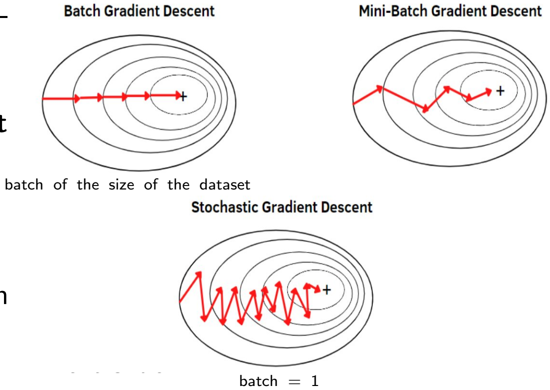
λ: learning rate, is discussed on the next slide

Other hyperparameters: batch size



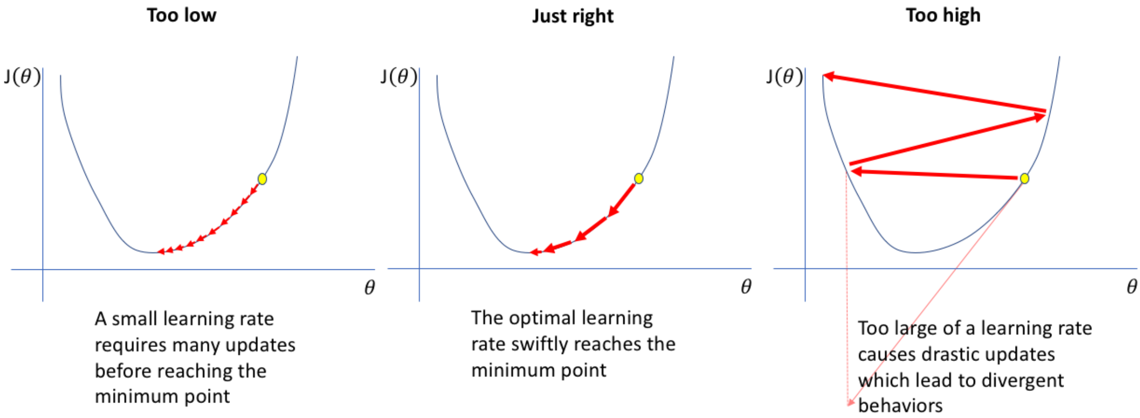
Batch: number of samples that will be simultaneously propagated through the network; based on these samples the weights of the model are updated. Update is based on the **gradient descent**

- \+ requires less memory
- +- trains slower but *can* converge faster
- \- gradient descent may be less accurate
- ! for larger batches the significant degradation of the model is observed (ability to generalize)

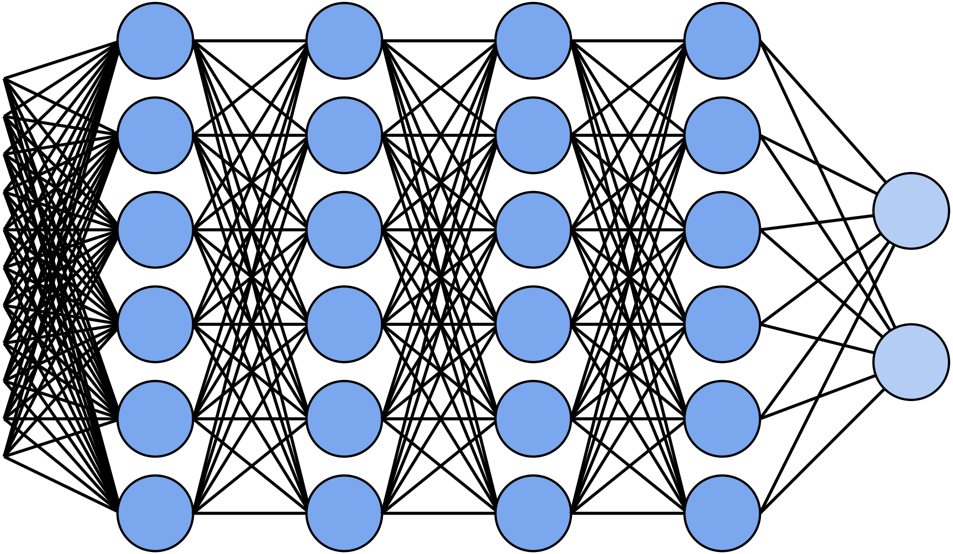


Other hyperparameters: learning rate

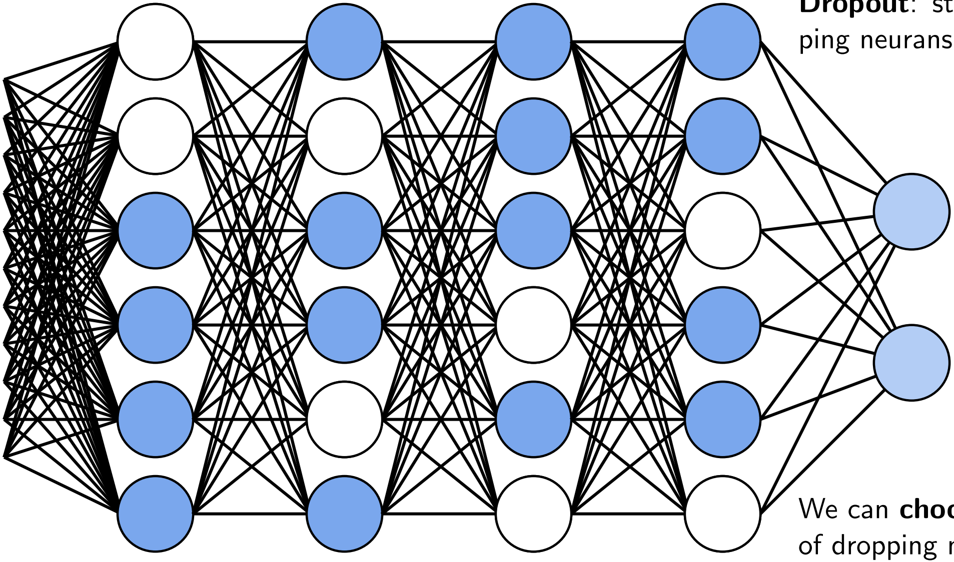
Learning rate determines how much the neural network weights change within the context of optimization while minimizing the loss function. It determines how much the **new information** will influence the model. It varies within $[0, 1]$.



Other hyperparameters: dropout



Other hyperparameters: dropout



Dropout: stochastically dropping neurans in the network

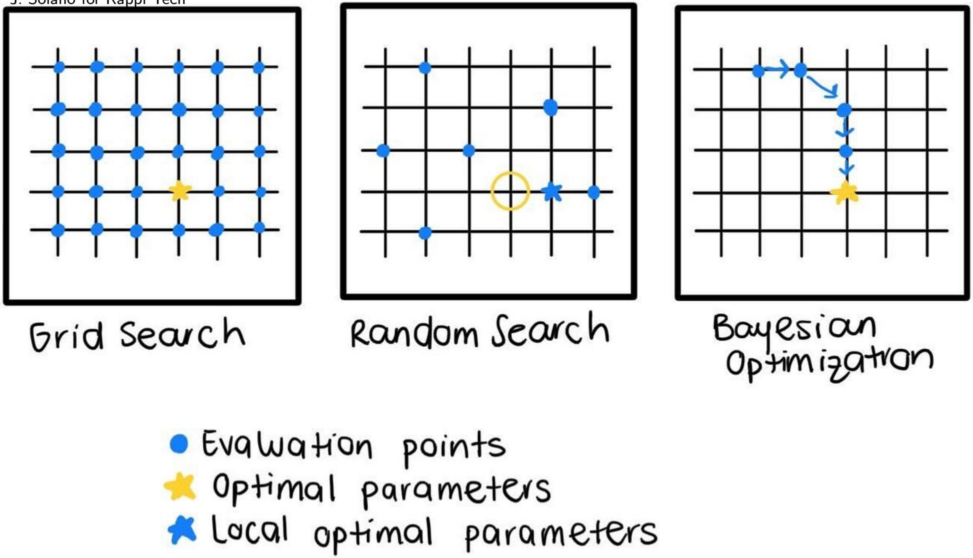
We can **choose** the probability of dropping neurons

Our parameters

• use all features in training	yes/no
• use all features for graph creation	yes/no
• k nearest neighbors	[1, ..., 70]
• number of convolutional layers	[1, 2, 3, 4, 5, 6, 7]
• number of hidden layers	[1, 2, 3, 4, 5, 6, 7]
• number of neurons	[2, 4, 8, 16, 32, 64, 128, 256, 512]
• batch size	[8, 16, 32, 64, 128, 256, 512, 1024]
• learning rate	[0.00001, 0.0001, 0.001, 0.01, 0.1]
• dropout	[0.1, 0.2, 0.3, 0.4]

We need to search for the best set of parameters!

J. Solano for Rappi Tech



Hyperparameters search: Bayesian optimization

Problem:

- optimise $f(x)$ (find min or max)
- $f(x)$ can be whatever
- $f'(x)$ is unknown
- evaluating $f(x)$ is expensive

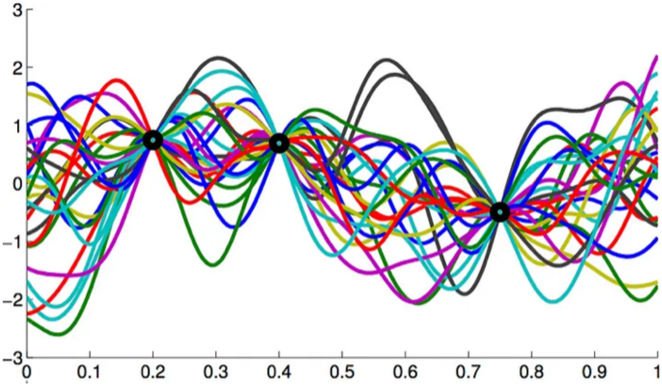
Hyperparameters search: Bayesian optimization

Problem:

- optimise $f(x)$ (find min or max)
- $f(x)$ can be whatever
- $f'(x)$ is unknown
- evaluating $f(x)$ is expensive

Solution:

- evaluate $f(x)$
- train gaussian process regressor



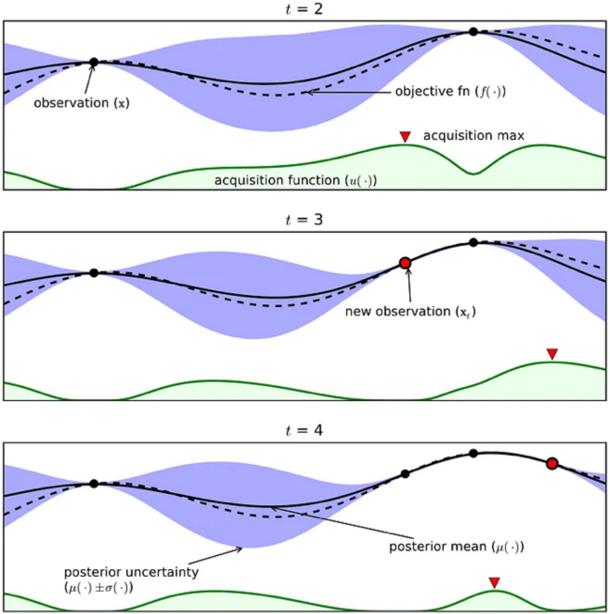
Hyperparameters search: Bayesian optimization

Problem:

- optimise $f(x)$ (find min or max)
- $f(x)$ can be whatever
- $f'(x)$ is unknown
- evaluating $f(x)$ is expensive

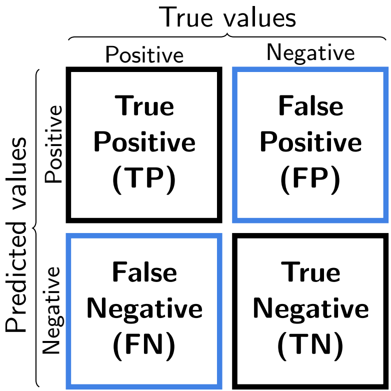
Solution:

- evaluate $f(x)$
- train gaussian process regressor
- calculate acquisition function
- define which evaluation to do next



Performance evaluation of binary classification

Confusion matrix



Binary classification: separate "signal" from "noise"

There are many metrics we can use to evaluate our model:

- **accuracy=** $\frac{\text{correct predictions}}{\text{total predctions}} = \frac{TP+TN}{TP+TN+FP+FN}$
- **precision=** $\frac{\text{correctly predicted positives}}{\text{total predicted positives}} = \frac{TP}{TP+FP}$
- **recall=** $\frac{\text{correctly predicted actual positives}}{\text{all actual}} = \frac{TP}{TP+FN}$

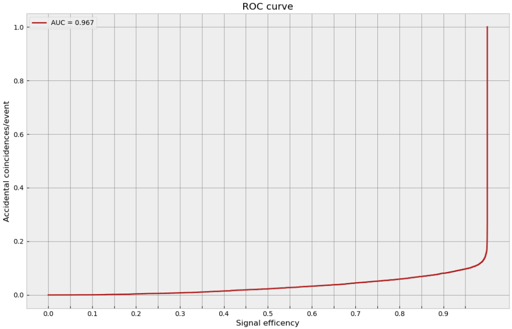
In **our case** we will use **accuracy**

Confusion matrix

		True values: Positive	True values: Negative
Predicted values	Positive	**True Positive (TP)**	**False Positive (FP)**
Predicted values	Negative	**False Negative (FN)**	**True Negative (TN)**

Receiver Operating Characteristic curve (ROC curve)

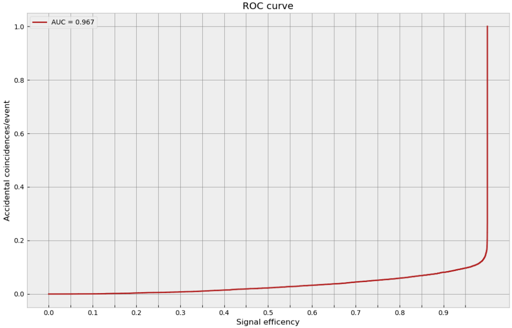
- **True Positive Rate**: $TPR = \frac{TP}{TP+FN}$ (signal efficiency)
- **False Positive Rate**: $FPR = \frac{FP}{FP+TN}$ (accidental background)



Performance evaluation of binary classification

Confusion matrix

		True values: Positive	True values: Negative
Predicted values	Positive	**True Positive (TP)**	**False Positive (FP)**
Predicted values	Negative	**False Negative (FN)**	**True Negative (TN)**



- the **bigger** area under curve (AUC) the **better**
- diagonal line from (0, 0) to (1, 1) corresponds to random choice
- allows to calculate model performance at various levels of **background acceptance**

Setup

Simulation:

	WCTE	Hyper-Kamiokande
Energy	200-1000 MeV	100-1000 MeV
Direction	Isotropic	Isotropic
Position	Center of WCTE	Isotropic inside the detector

Particle classification:

- electron vs muon
- muon$^-$ vs pion$^-$
- electron vs gamma

Ideal pipeline:

Small dataset
(large parameters phase space)

Large dataset
(precise parameter search)

Test set

Simulation:

	W
Energy	200-1(
Direction	Isotro
Position	Cente WCTE

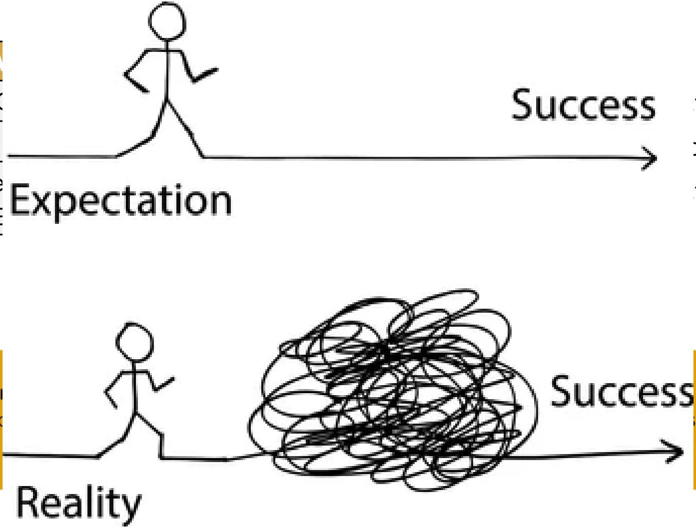
classification:

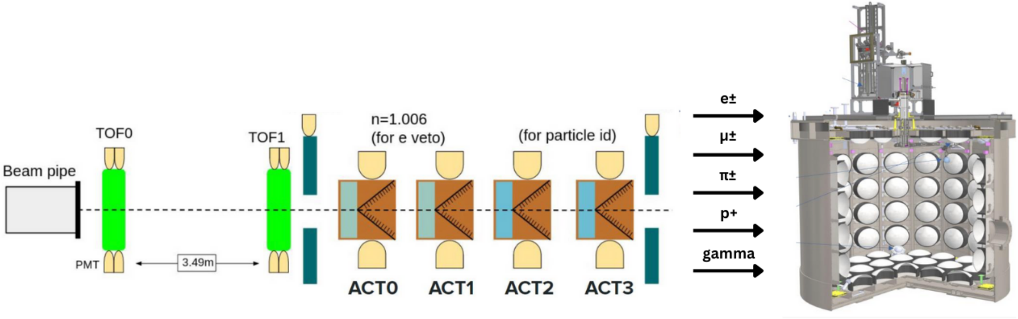
- tron vs muon
- on$^-$ vs pion$^-$
- tron vs gamma

Ideal pipeline:

Si (large p

et



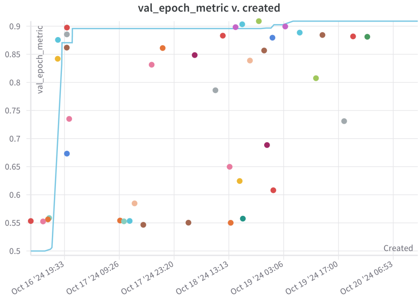


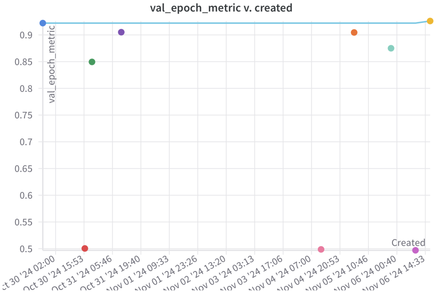
Goals: proof of technology and physics for Hyper-Kamiokande: measure important physical processes for water Cherenkov detectors: charged pion hadronic scattering, secondary neutron production, and Cherenkov light production from secondary particles. **Unique dataset for testing ML algorithms on the well-controlled data.**

- **Now:** WCTE beam data-taking
- **2025:** gadolinium loading


muon
VS
antiup quark
down quark

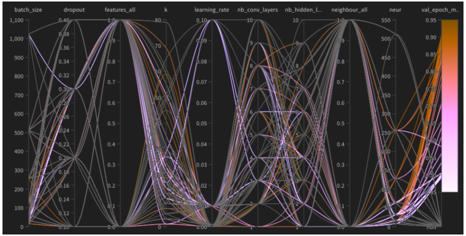
Parameters search

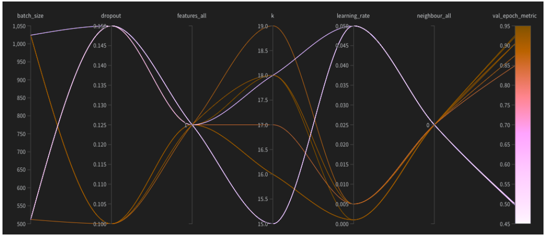




Bayesian optimisation-based parameter search

Bayesian optimisation-based parameter search

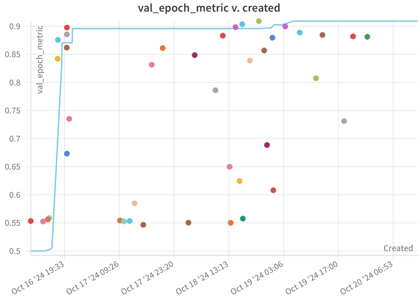




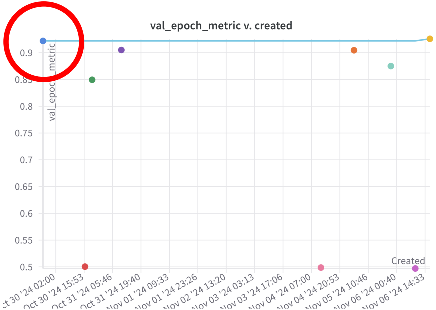
Small dataset, large phase-space

Big dataset, small phase-space

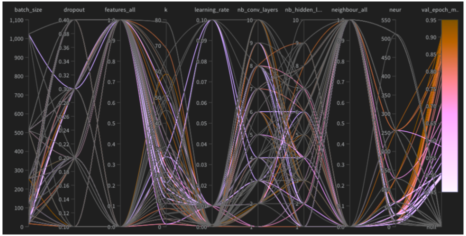
Parameters search



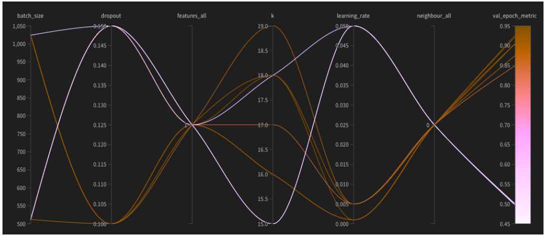
Bayesian optimisation-based parameter search



Bayesian optimisation-based parameter search



Small dataset, large phase-space

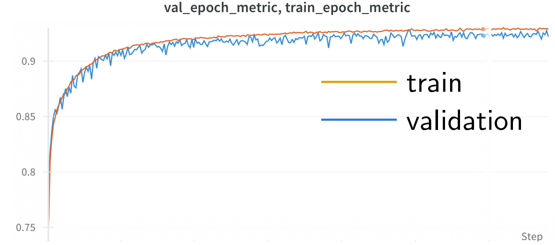


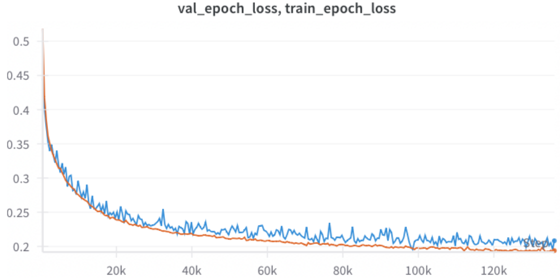
Big dataset, small phase-space

Best model

Hyperparameter	Value
use all features	true
use all features for graphs	false
k nearest neighbors	18
convolutional layers	2
hidden layers	3
neurons	32
batch size	512
learning rate	0.001
dropout	0.15

Number of parameters: 16242
Training time: 20.5 hours

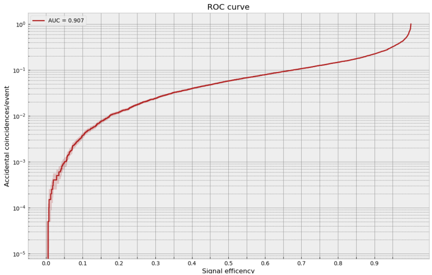


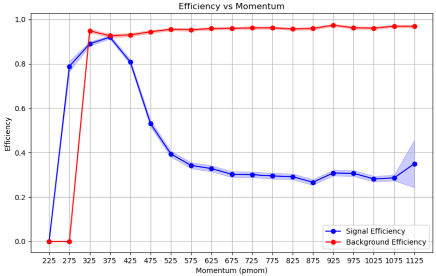


Performance on the test set

		Predicted values: Negative	Predicted values: Positive
True values	Negative	**18993**	**881**
True values	Positive	**11398**	**8599**

Overall accuracy: 69%

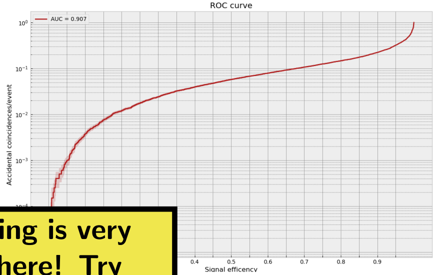




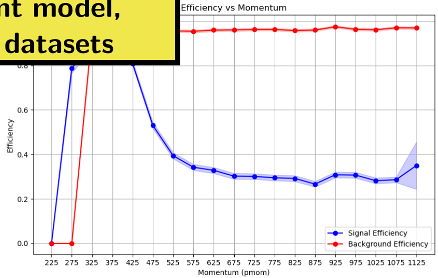
Performance on the test set

True values \ Predicted values	Negative	Positive
Negative	18993	881
Positive	11398	8599

Overall accuracy: 69%



Something is very wrong here! Try different model, check datasets

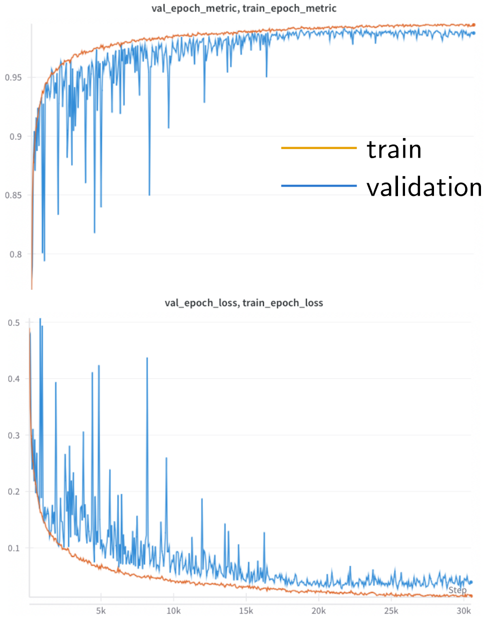



electron
VS
muon

Best model

Hyperparameter	Value
use all features	true
use all features for graphs	true
k nearest neighbors	23
convolutional layers	3
hidden layers	5
neurons	32
batch size	512
learning rate	0.001
dropout	0.2

Number of parameters: 34532
Training time: 6 hours, small dataset

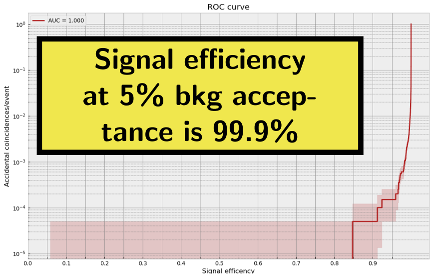


Performance on the test set

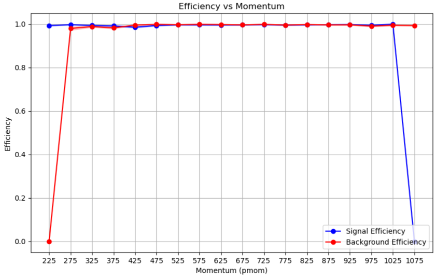
Predicted values

True values	Predicted: Negative	Predicted: Positive
Negative	**19901**	**96**
Positive	**93**	**19873**

Overall accuracy: 99.5%



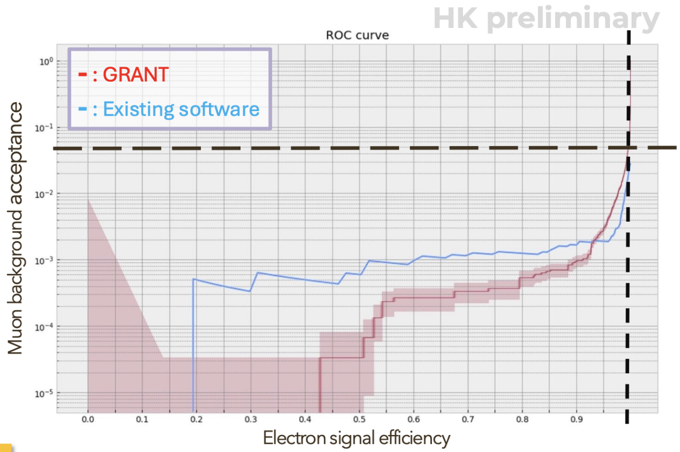
Signal efficiency at 5% bkg acceptance is 99.9%



3 Performance comparation: GRANT vs existing software

a) e/mu b) e/pi0 c) e/gamma d) Energy e) Vertex

Roc curve



GRANT :
99% efficiency at
5% bg acceptance

Existing software :
99% efficiency at
5% bg acceptance

0.1 s per event (GRANT)
1min30 (Existing software)

86

Christine's results: HK

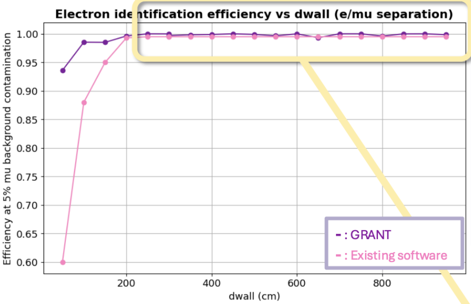
3 Performance comparation: GRANT vs existing software

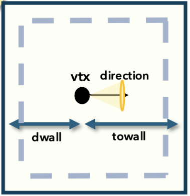
a) e/mu b) e/pi0 c) e/gamma d) Energy e) Vertex

dwall

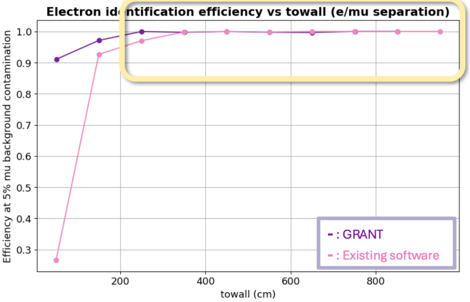
towall

HK preliminary





HK preliminary



After 2 m, efficiency above 99.4% !

88

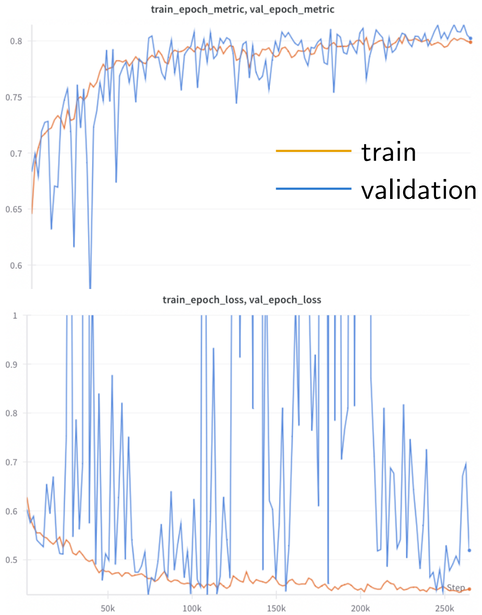
VS

Best model

Hyperparameter	Value
use all features	false
use all features for graphs	true
k nearest neighbors	31
convolutional layers	8
hidden layers	2
neurons	8
batch size	16
learning rate	0.001
dropout	0.1

Number of parameters: 12834

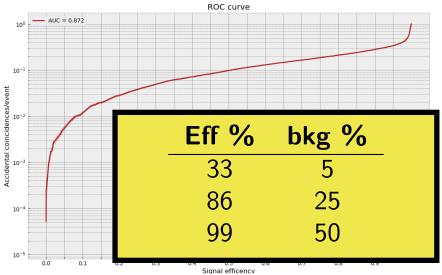
Training time: 3 hours, small dataset



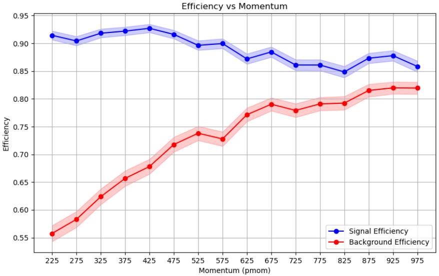
Performance on the test set

		Predicted values: Negative	Predicted values: Positive
True values	Negative	**13818**	**5132**
True values	Positive	**2203**	**17763**

Overall accuracy: 81%



Eff %	bkg %
33	5
86	25
99	50

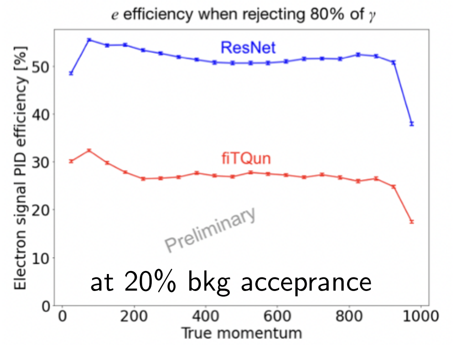


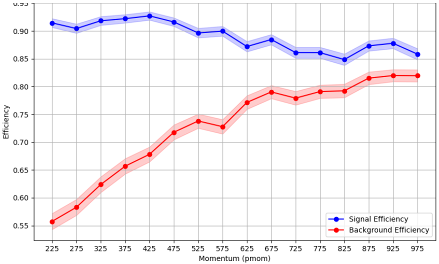
INSTITUT POLYTECHNIQUE DE PARIS

Performance on the test set

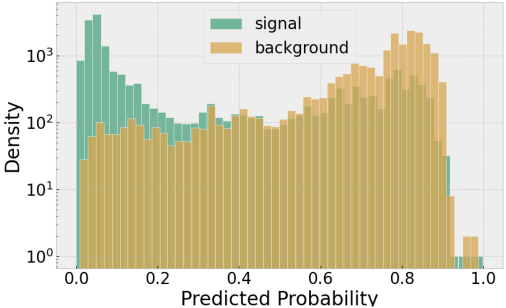
		Predicted values	
		Negative	Positive
True values	Negative	13818	5132
	Positive	2203	17763

Overall accuracy: 81%

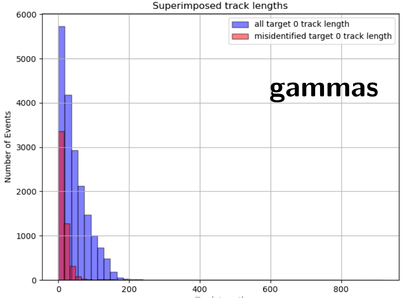


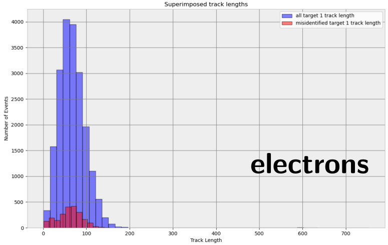


Error analysis

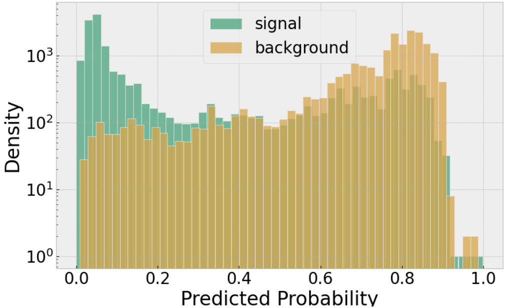


Separation strength of the two particle types, there is a **disproportion** towards misidentifying gammas

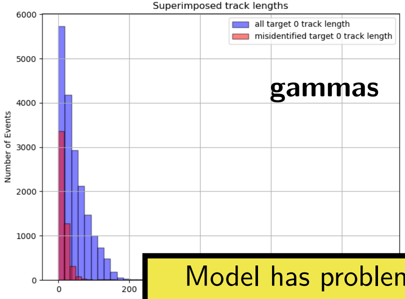




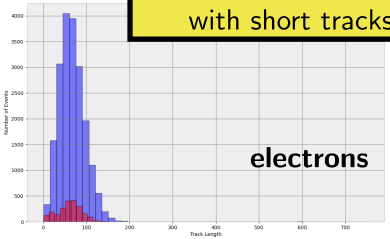
Error analysis



Separation strength of the two particle types, there is a **disproportion** towards misidentifying gammas



Model has problems with identifying particles with short tracks

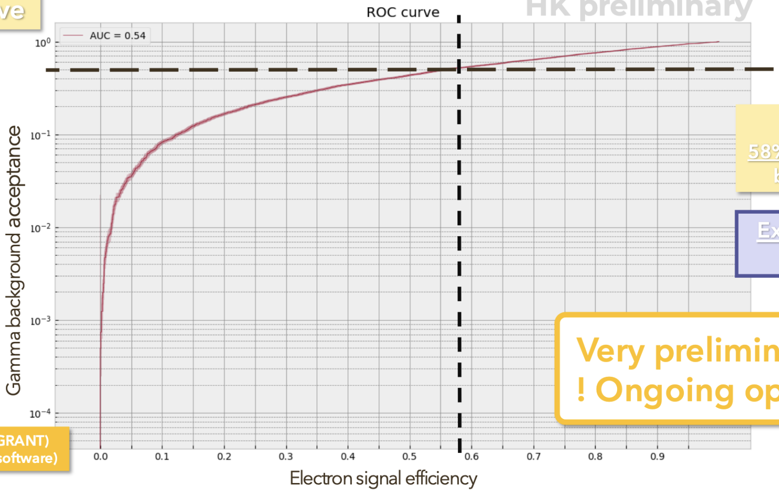


Christine's results: HK

3 Performance comparation: GRANT vs existing software

a) e/mu   b) e/pi0   c) e/gamma   d) Energy   e) Vertex

Roc curve



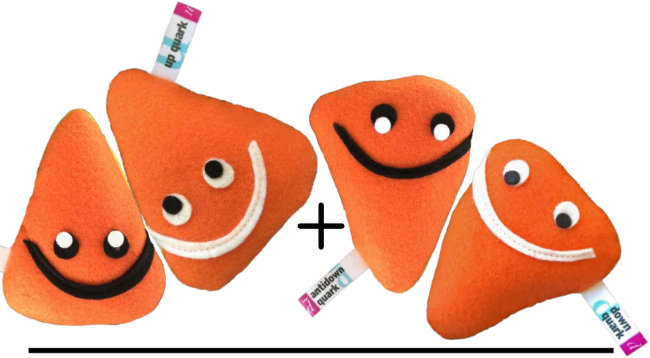
GRANT : 58% efficiency at 50% bg acceptance...

Existing software : Not done

Very preliminary results ! Ongoing optimization

0.1 s per event (GRANT)
1min30 (Existing software)

vs



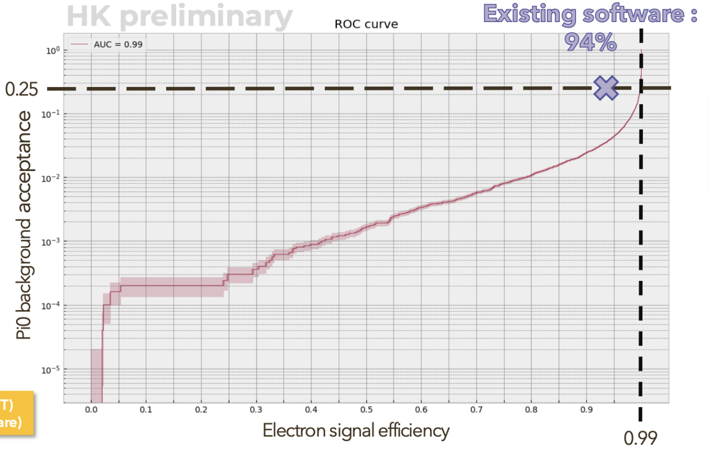
$\sqrt{2}$

Christine's results: HK

3 Performance comparation: GRANT vs existing software

a) e/mu **b) e/pi0** c) Energy d) Vertex e) Direction

Roc curve



GNN : 99% efficiency at 25% bg acceptance

Existing software : 94% efficiency at 25% bg acceptance

0.1 s per event (GRANT)
1min30 (Existing software)

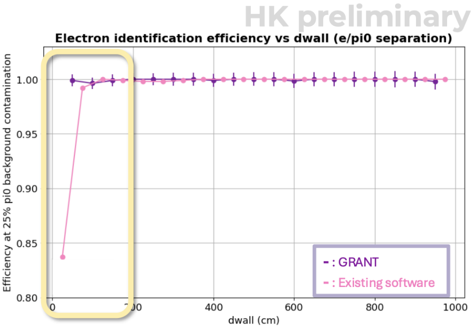
93

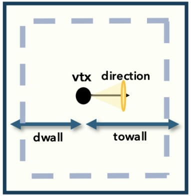
3 Performance comparation: GRANT vs existing software

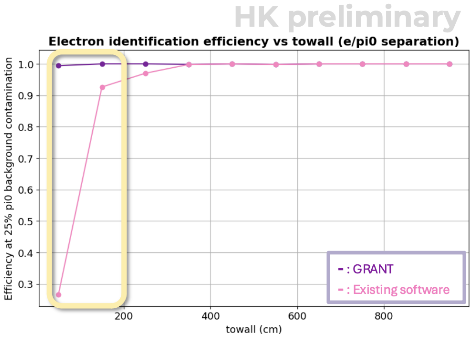
a) e/mu b) e/pi0 c) Energy d) Vertex e) Direction

dwall

towall







For events close to the wall : GNN > Existing software => potentially increase FV

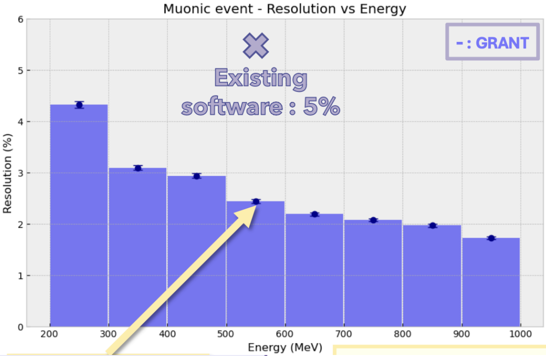
95

Energy reconstruction

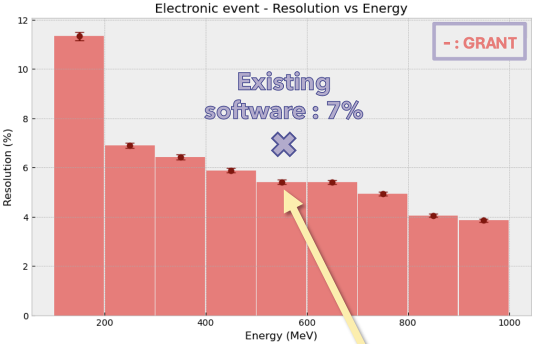
3 Performance comparation: GRANT vs existing software

a) e/mu b) e/pi0 c) e/gamma d) Energy e) Vertex

Muon



Electron



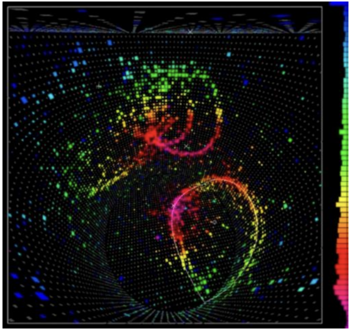
Energy reconstruction for e & mu (1D)	**Electron : 5.5%** resolution at 500 MeV, energy bias at **~1.5%** **Muon : 2.5%** resolution at 500 MeV, energy bias at **~0.5%**	**Electron : 7%** resolution at 500 MeV, energy bias at **~1%** **Muon : 6%** resolution at 500 MeV, energy bias at **~1%**

Now you know how graph neural nets work!

PID	WCTE	HK
muon vs pion	69% accuray, more work needed	-
electron vs muon	99.9% efficiency at 5% bkg	99.9% efficiency at 5% bkg
electron vs gamma	86% efficiency at 25% bkg	58% efficiency at 50% bkg, in progress
electron vs pion	-	99% efficiency at 25% bkg

Prospects:

- Continuing the effort for multidimensional reconstruction
  - Optimizing for 3D vertex reconstruction,
  - Simultaneous vertex and direction reconstruction
- Enhancing e/gamma and muon/pion separation
- μ^+/μ^- and e/π^0 separation for WCTE (to be developed)
- Ring counting (to be developed)
- Application to SK data



BACK UP