Graph neural network for stop pair and $Ht\bar{t}$ productions at LHC

Jin Min Yang

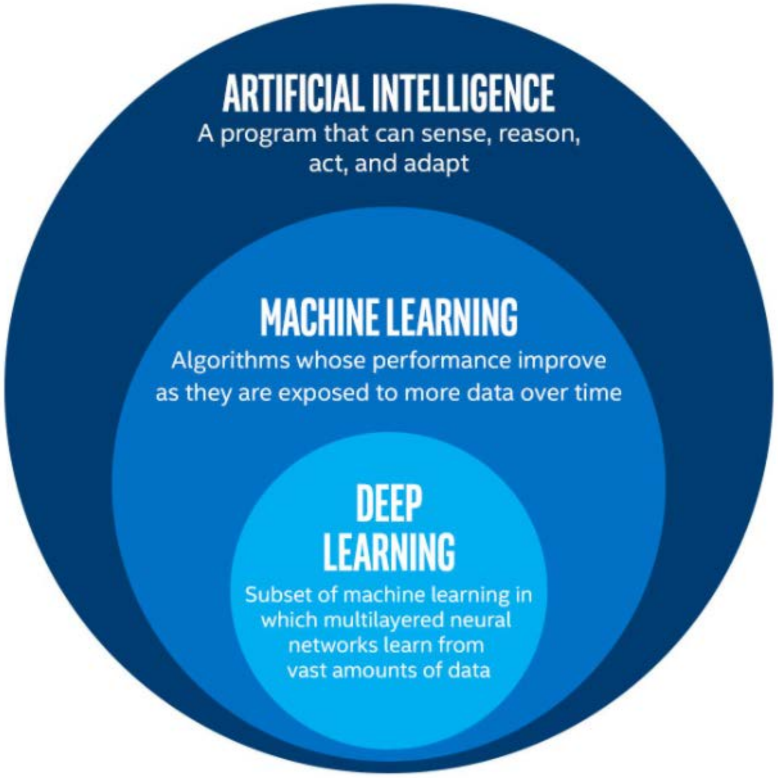
ITP, Beijing/Tohoku U

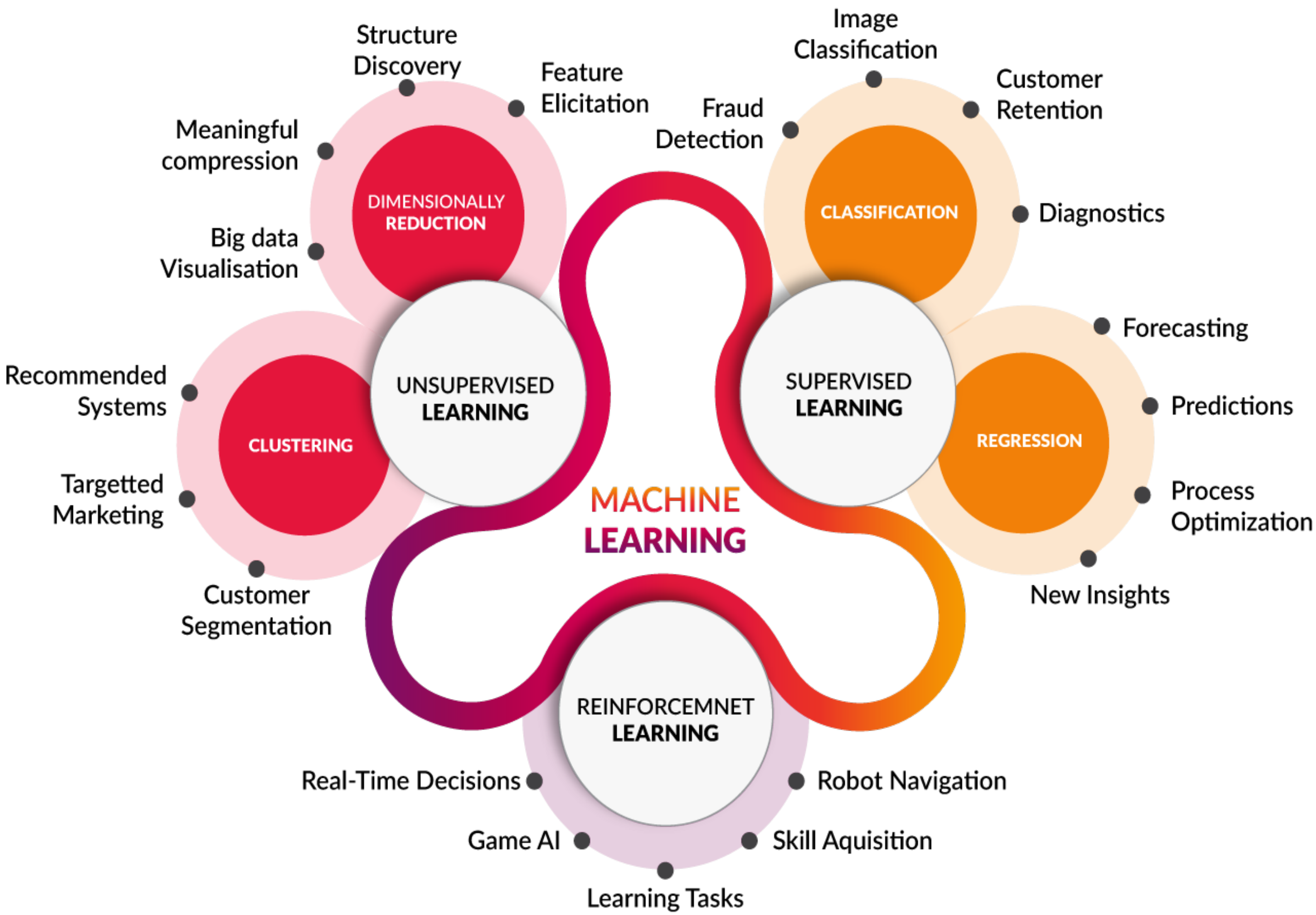
1807.09088 M Abdu, J Ren, L Wu, JMY

1901.05627 J Ren, L Wu, JMY

Contents

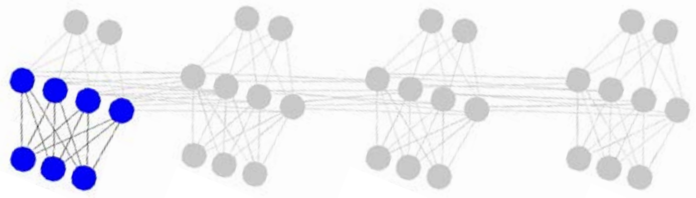
1 Introduction



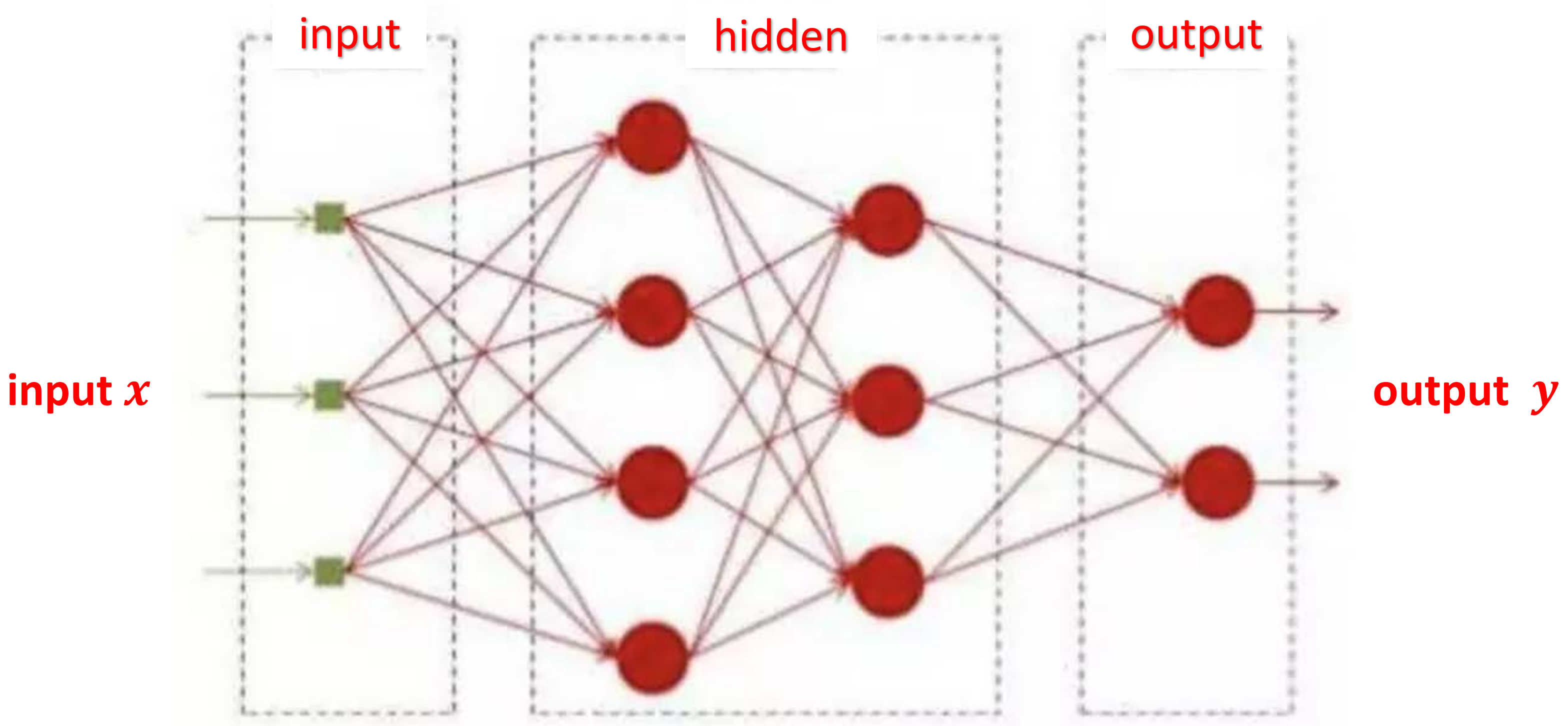

MACHINE LEARNING
UNSUPERVISED LEARNING
DIMENSIONALLY REDUCTION
Structure Discovery
Feature Elicitation
Meaningful compression
Big data Visualisation
CLUSTERING
Recommended Systems
Targetted Marketing
Customer Segmentation
SUPERVISED LEARNING
CLASSIFICATION
Image Classification
Customer Retention
Fraud Detection
Diagnostics
REGRESSION
Forecasting
Predictions
Process Optimization
New Insights
REINFORCEMNET LEARNING
Real-Time Decisions
Robot Navigation
Game AI
Skill Aquisition
Learning Tasks

Neural Network

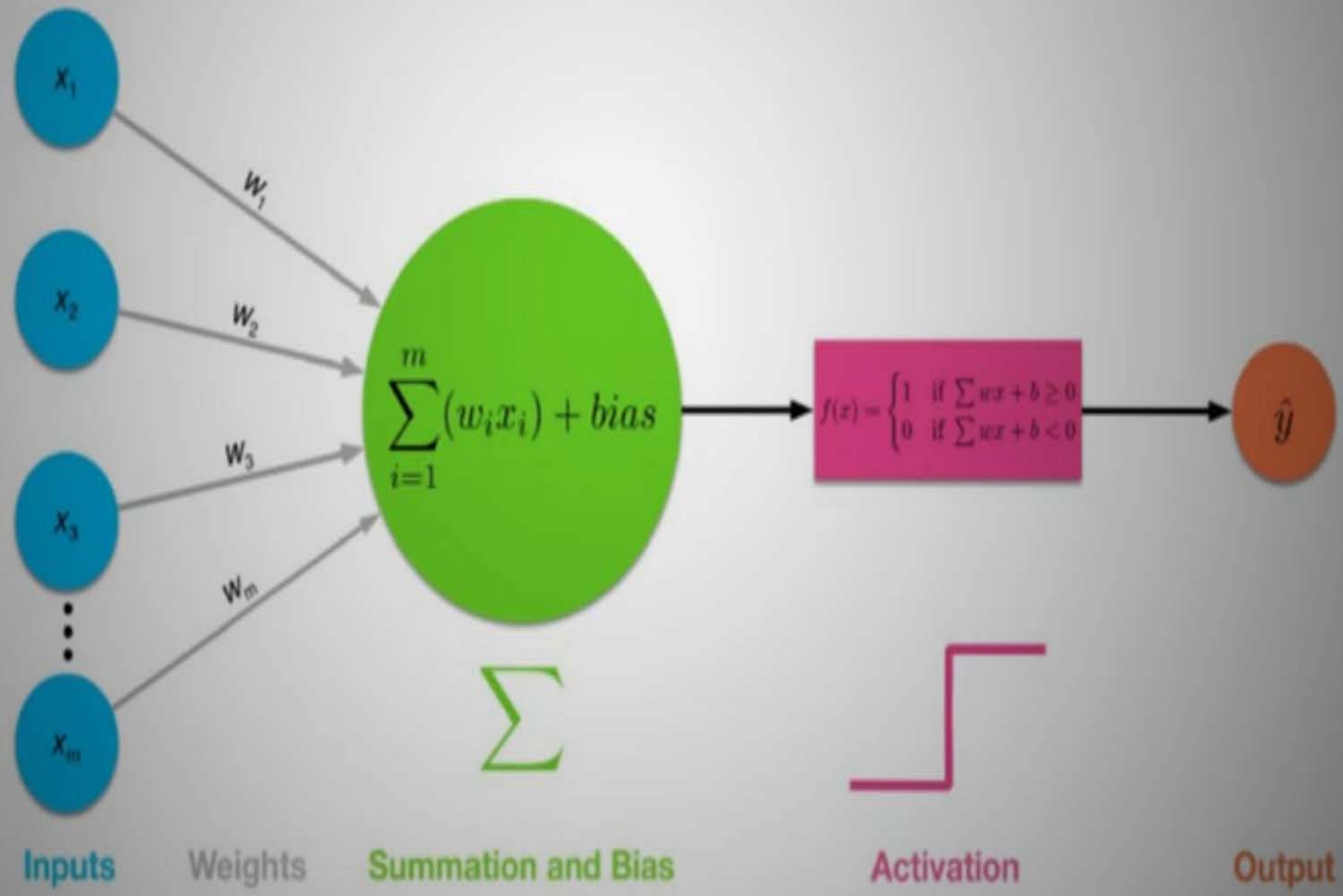
- Warren McCulloch and Walter Pitts (**1943**) created the **first neural network** based on mathematics and algorithms called threshold logic.
- The **perceptron algorithm** was invented in **1957** at the Cornell Aeronautical Laboratory by Frank Rosenblatt.
- For **multilayer perceptron** (feed-forward neural network), where at least one hidden layer exists, more sophisticated algorithms such as backpropagation (Rumelhart, Hinton and Williams, **1986**) must be used.



Neural Network

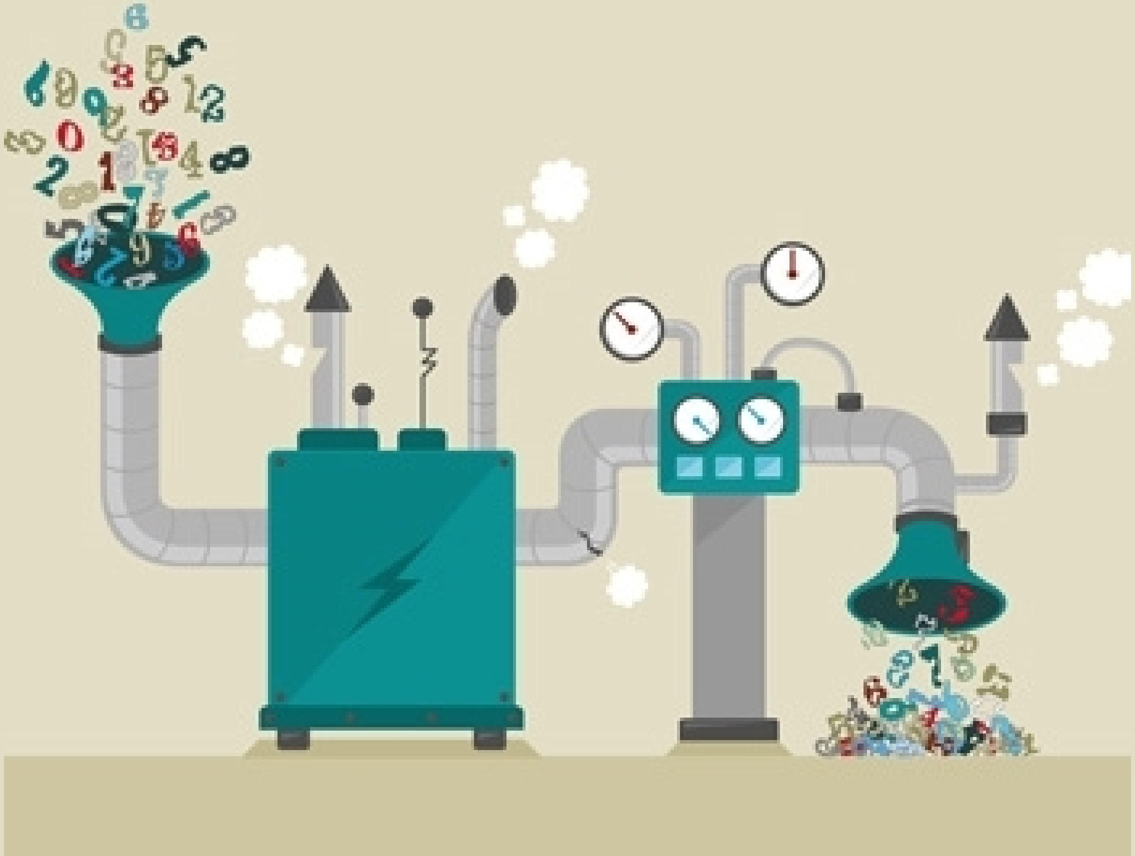


ARTIFICIAL NEURON - THE HEART OF A NEURAL NETWORK



Difficult to interpret

(crucial for physics but not for industry)



Machine Learning in HEP

GOAL

- **“Solve” HEP problems using DATA**

EXAMPLE

- Physics model selection
  - Scan (e.g. 1011.4306, 1106.4613, 1703.01309, 1708.06615)
- Collider
  - Parton distribution function (e.g. 1605.04345)
  - Object reconstruction (e.g. NIPS-DLPS)
  - Pileup mitigation (e.g. 1512.04672, 1707.08600)
  - Jet tagging (e.g. 1407.5675, 1501.05968, 1612.01551, 1702.00748)
  - Event selection (e.g. 1402.4735, 1708.07034, 1807.09088)
  - Decayed object reconstruction
  - Anomaly event detection (e.g. 1807.10261)

1.2 Brief Overview of Machine Learning Algorithms in HEP

This section provides a brief introduction to the most important machine learning algorithms in HEP, introducing key vocabulary (in *italic*).

Machine learning methods are designed to exploit large datasets in order to reduce complexity and find new features in data. The current most frequently used machine learning algorithms in HEP are Boosted Decision Trees (BDTs) and Neural Networks (NN).

Typically, variables relevant to the physics problem are selected and a machine learning *model* is *trained* for *classification* or *regression* using signal and background events (or *instances*). Training the model is the most human- and CPU-time consuming step, while the application, the so called *inference* stage, is relatively inexpensive. BDTs and NNs are typically used to classify particles and events. They are also used for regression, where a continuous function is learned, for example to obtain the best estimate of a particle's energy based on the measurements from multiple detectors.

Neural Networks have been used in HEP for some time; however, improvements in training algorithms and computing power have in the last decade led to the so-called Deep Learning revolution, which has had a significant impact on HEP. Deep Learning is particularly promising when there is a large amount of data and features, as well as symmetries and complex non-linear dependencies between inputs and outputs.

There are different types of deep neural networks used in HEP: fully-connected (FCN), convolutional (CNN) and recurrent (RNN). Additionally, neural networks are used in the context of Generative Models, when a Neural Network is trained to mimic multidimensional distributions to generate any number of new instances. Variational AutoEncoders (VAE) and more recent Generative Adversarial Networks (GAN) are two examples of such generative models used in HEP.

Our Work

- **Machine learning in SUSY parameter space exploration**

  1708.06615 J. Ren, L. Wu, JMY, J. Zhao

- **Graph neural network for stops at LHC**

  1807.09088 M Abdu, J Ren, L Wu, JMY

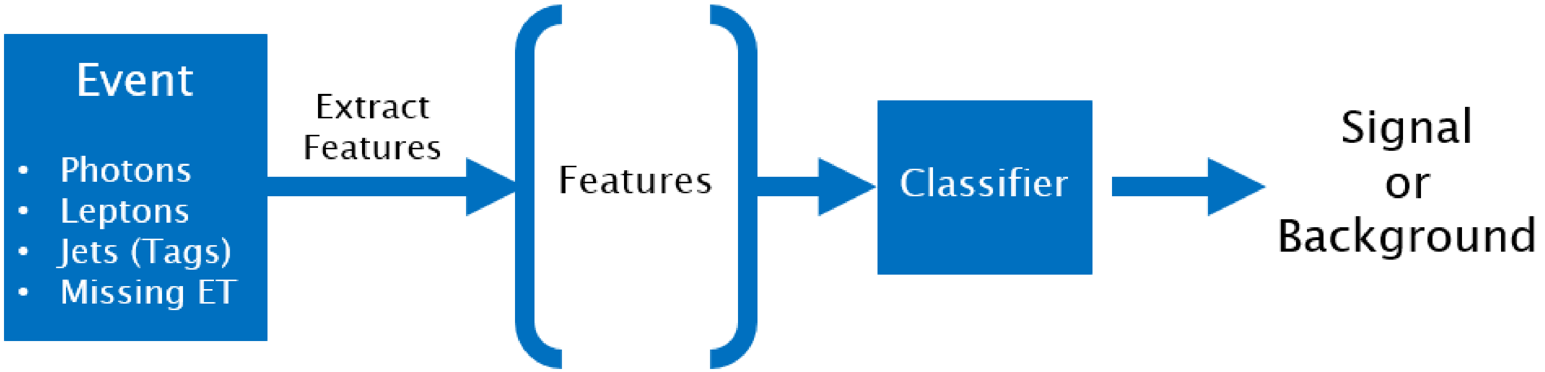
- **Graph neural network for $Ht\bar{t}$ search at LHC**

  1901.05627 J Ren, L Wu, JMY

2 Graph Neural Network for stops at LHC

1807.09088 M Abdu, J Ren, L Wu, JMY

An **event** is a **signal** or **background** ?



HIGH–LEVEL FEATURES

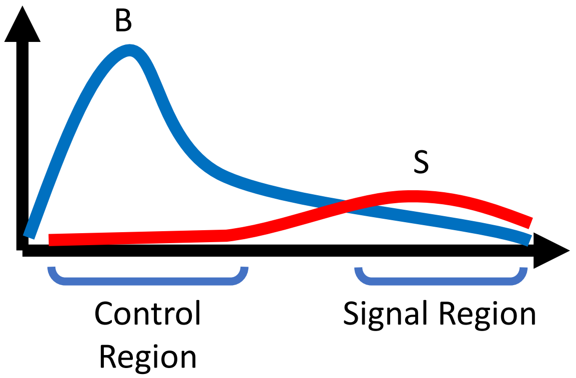
- Number of jets
- p_T of the leading lepton
- $\Delta\phi$ between the leading jet and missing ET
- Reconstructed top mass
- ...

LOW–LEVEL FEATURES

- four–momenta of reconstructed objects
- ...

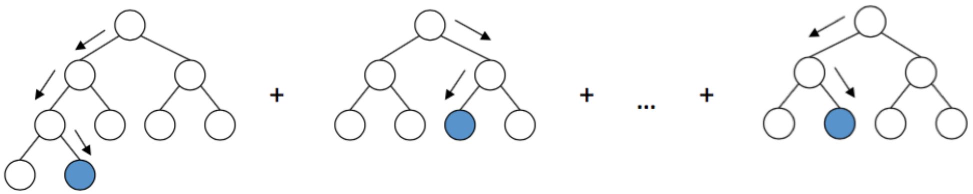
Methods for Event Selection

- **Cut-flow**



Methods for Event Selection

- **Cut-flow**
- **Machine Learning**
  - **Boosted Decision Tree (BDT)**

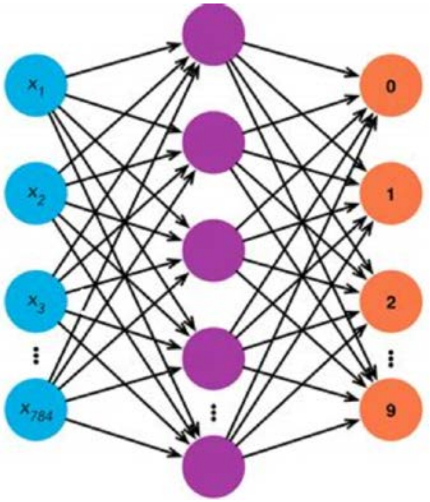


a lot of trees → a forest

When an event comes, it passes each tree and is valued 1(signal) or 0(background). Finally, these values are averaged.

Methods for Event Selection

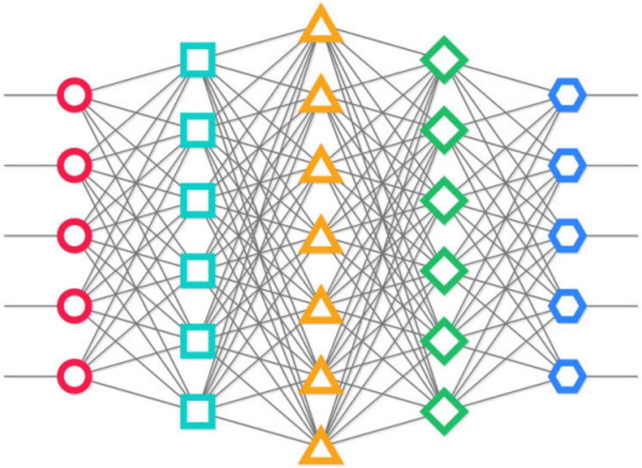
- **Cut-flow**
- **Machine Learning**
  - **Boosted Decision Tree (BDT)**
  - **Neural Networks**
    - Shallow Neural Network (NN)



Methods for Event Selection

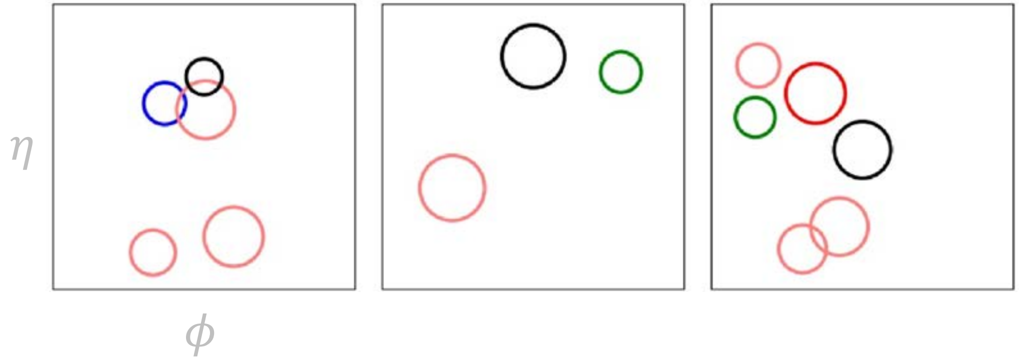
- **Cut-flow**
- **Machine Learning**
  - **Boosted Decision Tree (BDT)**
  - **Neural Networks**
    - Shallow Neural Network (NN)
    - Deep Learning
      - Deep Neural Network (DNN)

1410.3469, 1402.4735, 1803.01550



Methods for Event Selection

- **Cut-flow**
- **Machine Learning**
  - **Boosted Decision Tree (BDT)**
  - **Neural Networks**
    - Shallow Neural Network (NN)
    - Deep Learning
      - Deep Neural Network (DNN) 1410.3469, 1402.4735, 1803.01550
      - Convolutional Neural Network (CNN) 1708.07034

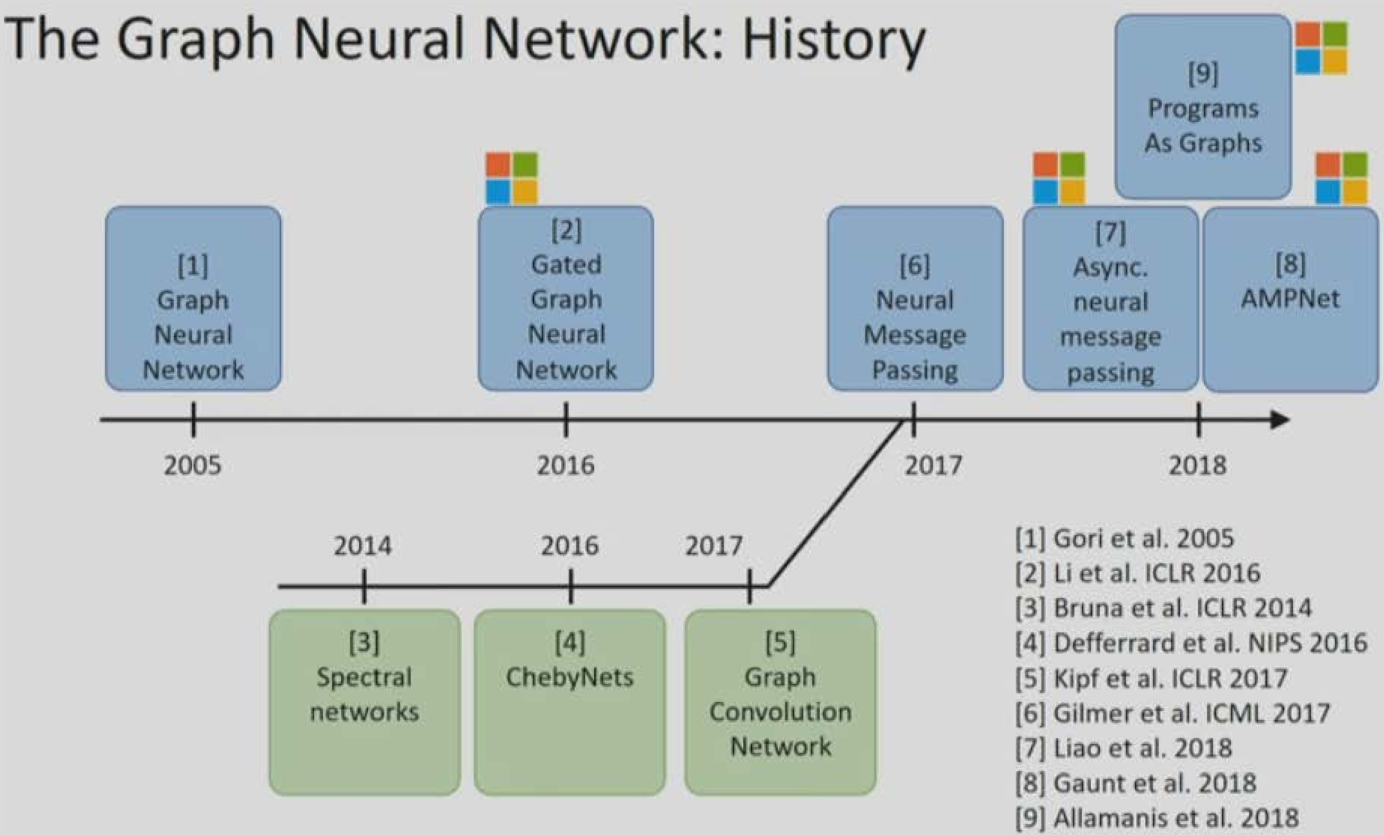


color—particle type
size —energy or tansverse momentum

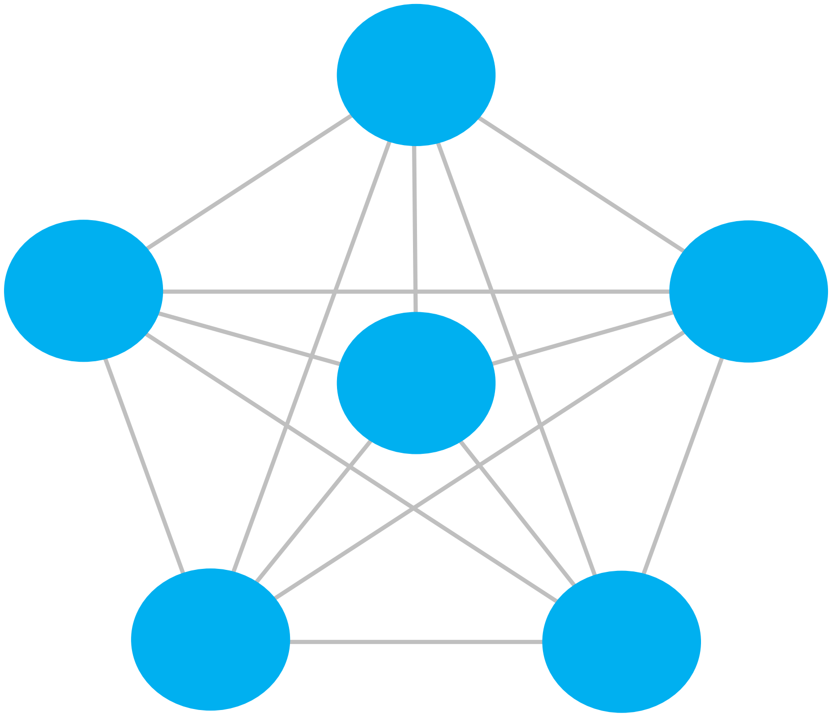
Pros and Cons

- Cut-flow is simple but coarse, hurts signal events.
- BDT can be viewed as an optimized version of cut-flow.
- BDT is explainable, while NNs are hard to explain.
- NNs are more powerful than BDT for non-linear mapping.
- Most machine learning methods use fixed-length of inputs.

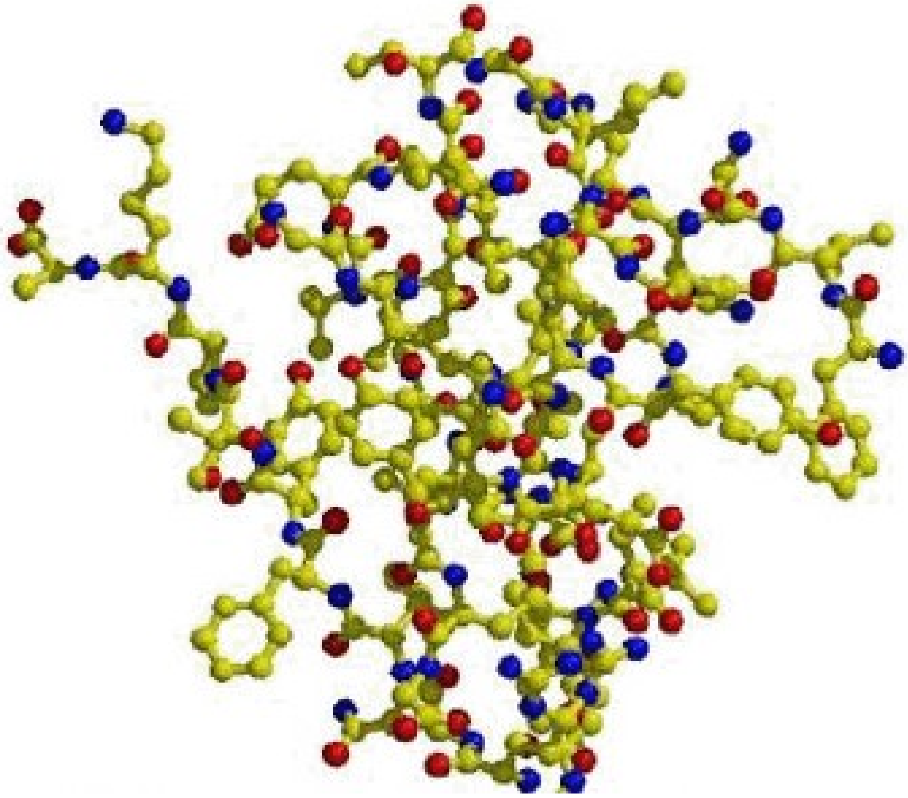
Message Passing Graph Neural Network



Message Passing Graph Neural Network



Message Passing Graph Neural Network



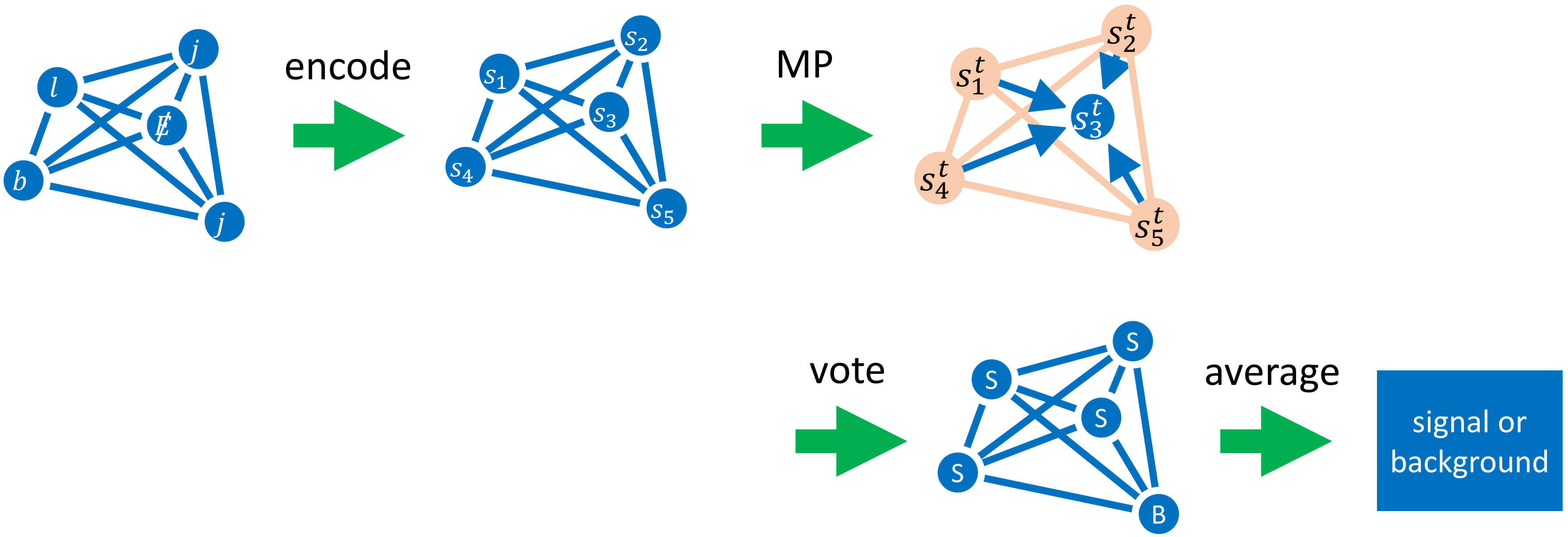
Event as Graph

Our Idea

- Represent an event as a ***graph*** $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$
- Encode each vertex into a ***state vector***
- ***Message passing*** between vertices
- Each vertex ***votes*** the signal/background
- ***Average the votes*** as the final result

$\boldsymbol{E}$: $d_{ij} \equiv \sqrt{\Delta y_{ij}^2 + \Delta\phi_{ij}^2}$

$\boldsymbol{V}$: $(0,0,1,0,m,E,P_T)$



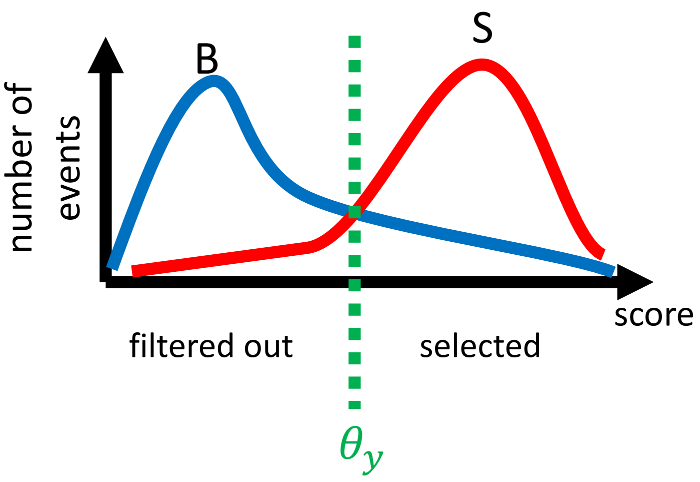
Graph Classification as Event Selection

The output value is called ***Score***, which can be understood as the likeliness of the event being a signal.

Apply a cut on the score:

- If $y \geq \theta_y$, keep the event.
- If $y < \theta_y$, drop the event.

As a result, **most** signal events and **some** background events are selected.



Performance Index

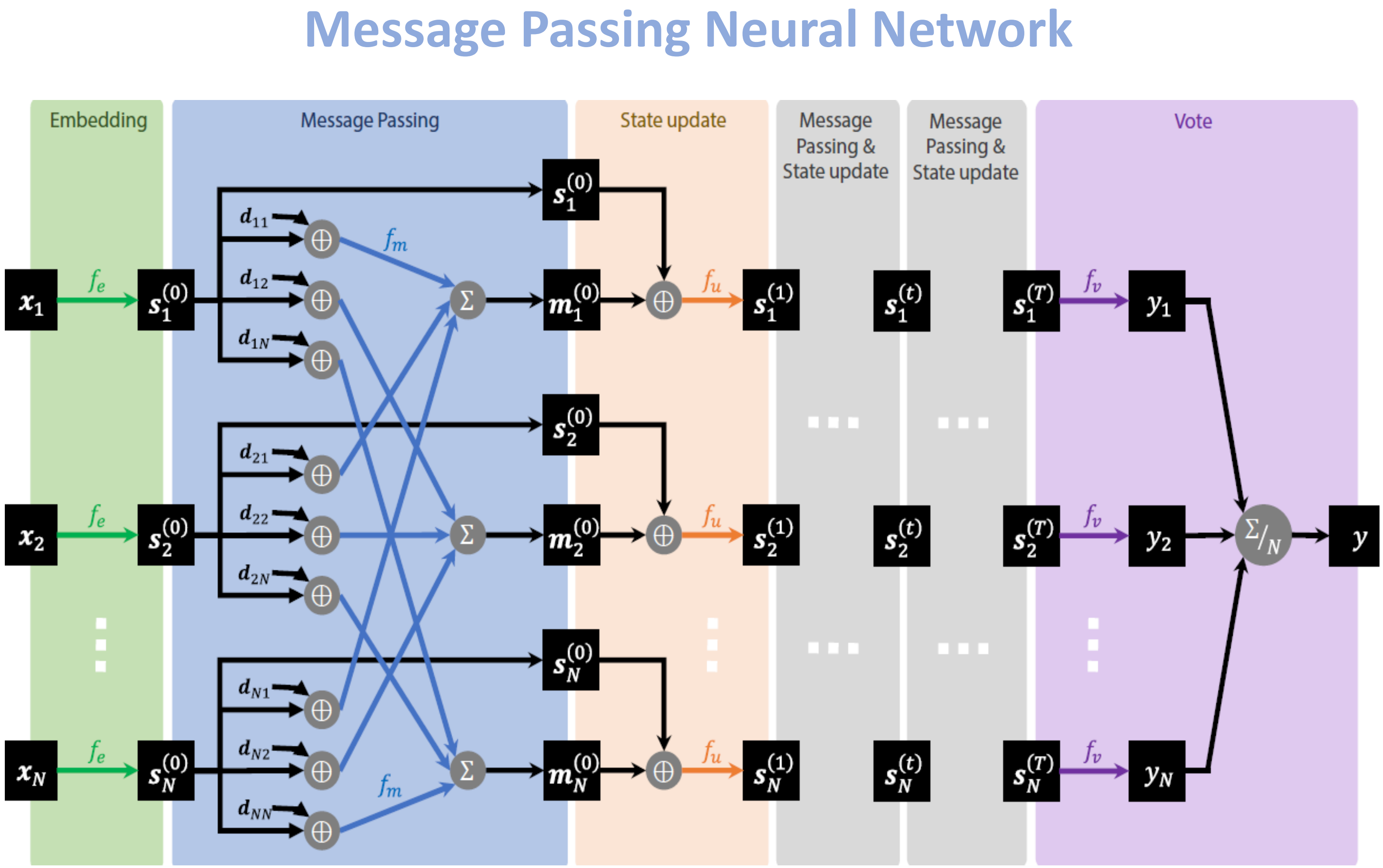
Expected discovery significance is

$$\frac{S}{\sqrt{B}} = \frac{\sigma_S L \epsilon_S^0}{\sqrt{\sigma_B L \epsilon_B^0}} \cdot \frac{\epsilon_S}{\sqrt{\epsilon_B}}$$

- S, B: the number of selected signal and background events
- σ: cross section
- L: integrated luminosity
- ϵ^0, ϵ: efficiencies of preselection cuts and classifier

We define the expected relative discovery significance as $\epsilon_S / \sqrt{\epsilon_B}$

Detailed operation（1）



Detailed operation （2）

Neural Network Model

- Use ***one-hot-like*** encoding for object identity.
- ***30-dim*** feature vectors
- ***Distance*** measure using $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$
- Pair distances are expanded in a Gaussian basis (linearly distributed in [0, 5]) as vectors of length 21.
- Use separate message and update functions for each iteration.

- $f_e(\text{id}, E, p_T) = \text{relu}\left(W_e \begin{bmatrix} \text{onehot(id)} \\ p_T \\ E \end{bmatrix} + b_e\right)$
- $f_m^{(t)}(s, d) = \text{relu}\left(W_m^{(t)} \begin{bmatrix} s \\ \text{expand}(d) \end{bmatrix} + \boldsymbol{b}_m^{(t)}\right)$
- $f_u^{(t)}(s, m) = \text{relu}\left(W_u^{(t)} \begin{bmatrix} s \\ m \end{bmatrix} + \boldsymbol{b}_u^{(t)}\right)$
- $f_v(\boldsymbol{s}) = \sigma(W_v \boldsymbol{s} + \boldsymbol{b}_s)$

relu: rectified linear unit (non linear trans)

expand: expand to Gausian basis to form a vector

$\boldsymbol{b}_e, \boldsymbol{b}_m, \boldsymbol{b}_u, \boldsymbol{b}_s$ **:** parameters

$\boldsymbol{W}_e, \boldsymbol{W}_m, \boldsymbol{W}_u, \boldsymbol{W}_v$ **:** linear transformations

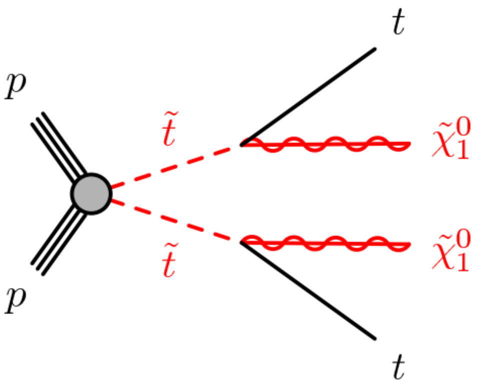
Training

- Binary Cross-Entropy (BCE) as loss function.
- Calculate gradients using error back-propagation.
- Optimize network parameters using Adam algorithm.
- Training with mini-batch of examples.
- Adopt early stopping to prevent overfitting.

Search for stop-pair signal

The diagonal region of phase space ($m_{\tilde{t}_1} \approx m_{\tilde{\chi}_1^0} + m_t$) is hard to hunt

- The momentum transfer from $\tilde{t}_1$ to $\tilde{\chi}_1^0$ is small.
- The stop signal is kinematically very similar to $t\bar{t}$ process.



Search for stop-pair signal

Generate events with ATLAS detector

- MadGraph5 + Pythia8 + Delphes3 + CheckMate2
- Signal events
  - $pp \to \tilde{t}_1 \bar{\tilde{t}}_1$
- Background events
  - $pp \to t\bar{t}$

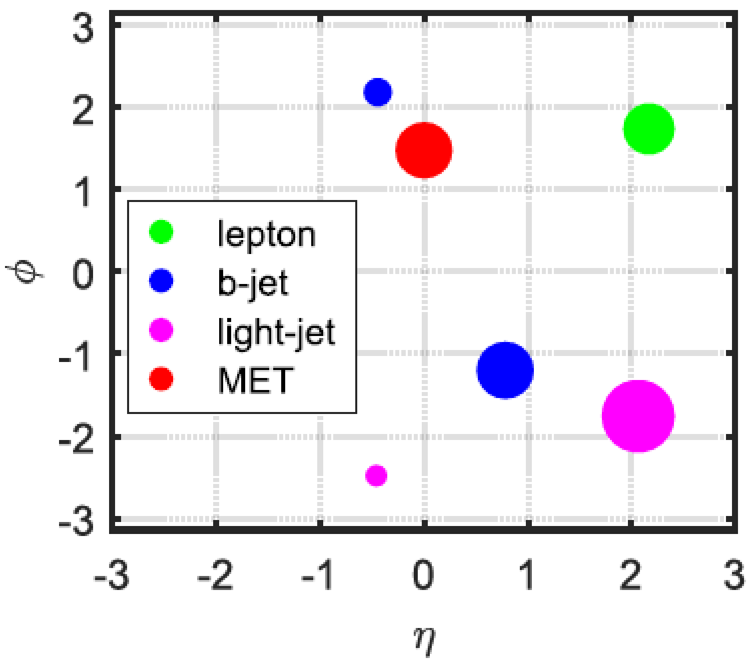
Object reconstruction

- Electron and muon
  - $p_T > 10 \text{ GeV}, |\eta| < 2.5$
- Jet
  - Anti-kt clustering ($R = 0.4$)
  - $p_T > 25 \text{ GeV}, |\eta| < 2.5$
- B-tagging
  - 80% efficiency

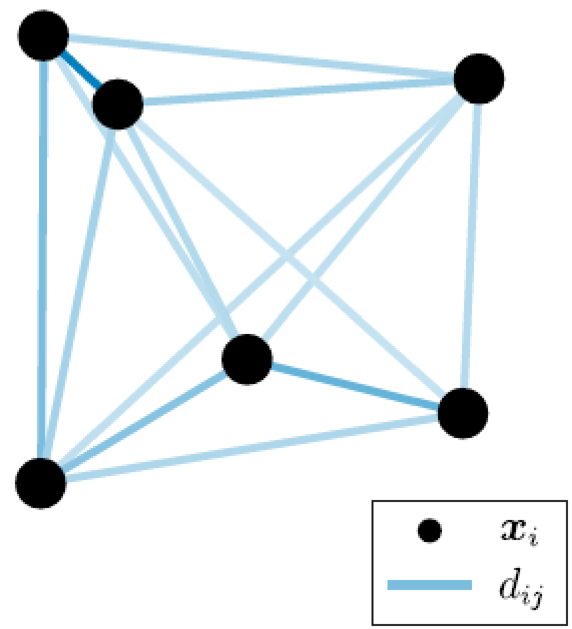
Preselection criteria

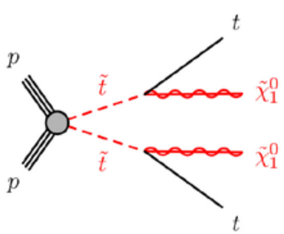
- $N(l) = 1$
- $N(j) \geq 4$
- $N(b) = 2$
- $\text{MET} > 150 \text{ GeV}$

Search for stop-pair signal

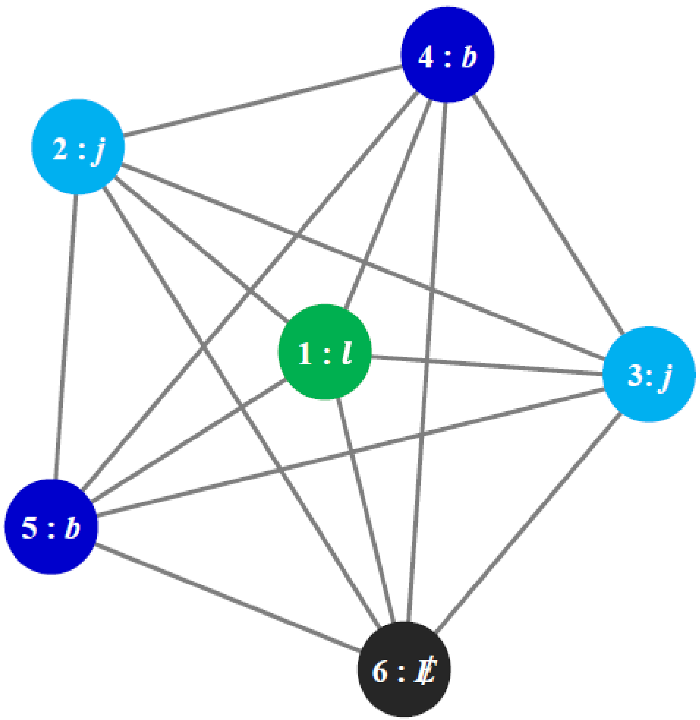


encoding





$$d_{ij} \equiv \sqrt{\Delta y_{ij}^2 + \Delta \phi_{ij}^2}$$



x

	Photon	lepton charge	b-jet or light jet	MET	p_T (TeV)	E (TeV)	m (TeV)
$x_1 = ($	0	-1	0	0	0.0229	0.0289	0.0000 $)$
$x_2 = ($	0	0	-1	0	0.2637	0.3304	0.0373 $)$
$x_3 = ($	0	0	-1	0	0.1003	0.1888	0.0091 $)$
$x_4 = ($	0	0	1	0	0.0980	0.1146	0.0133 $)$
$x_5 = ($	0	0	1	0	0.0689	0.0773	0.0062 $)$
$x_6 = ($	0	0	0	1	0.2107	0.2107	0.0000 $)$

d

	1	2	3	4	5	6
1	0	1.3971	2.5649	1.2801	3.2752	3.0312
2	1.3971	0	1.9019	1.6688	3.0871	3.1717
3	2.5649	1.9019	0	3.4440	1.5805	1.7831
4	1.2801	1.6688	3.4440	0	2.2175	2.1387
5	3.2752	3.0871	1.5805	2.2175	0	0.4912
6	3.0312	3.1717	1.7831	2.1387	0.4912	0

An event graph with detailed node features and distance matrix, built from a Monte Carlo simulated event

Search for stop-pair signal

Training set

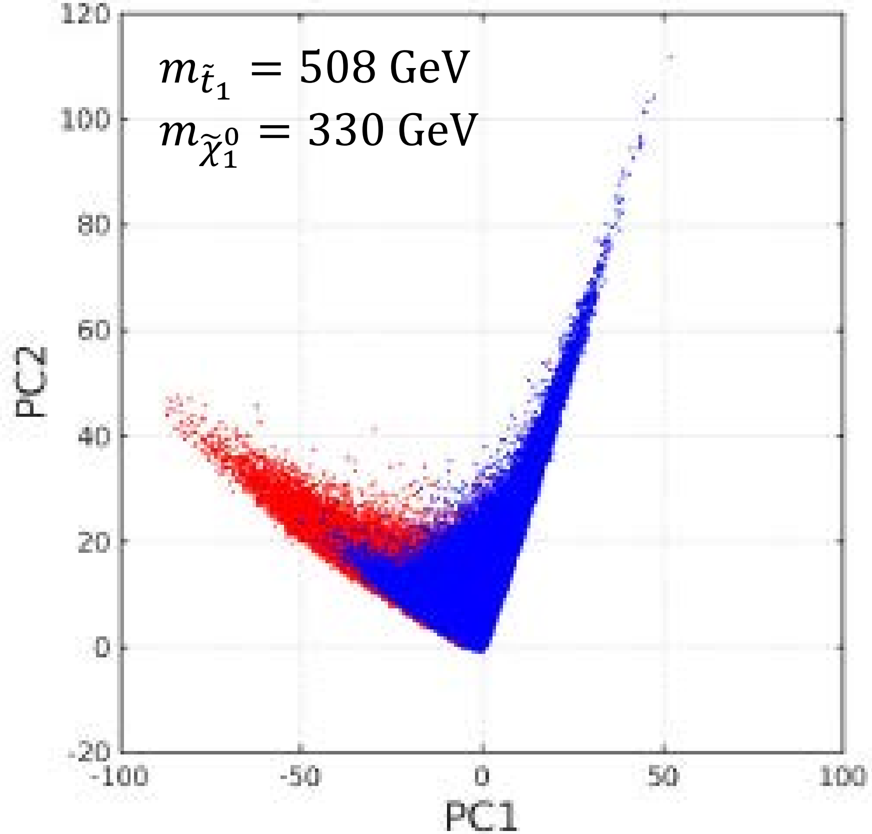
- 300,000 signal events and 300,000 background events

Validation set

- 100,000 signal events and 100,000 background events

Tools

- BDT: scikit-learn
- NN/DNN/MPNN: pytorch

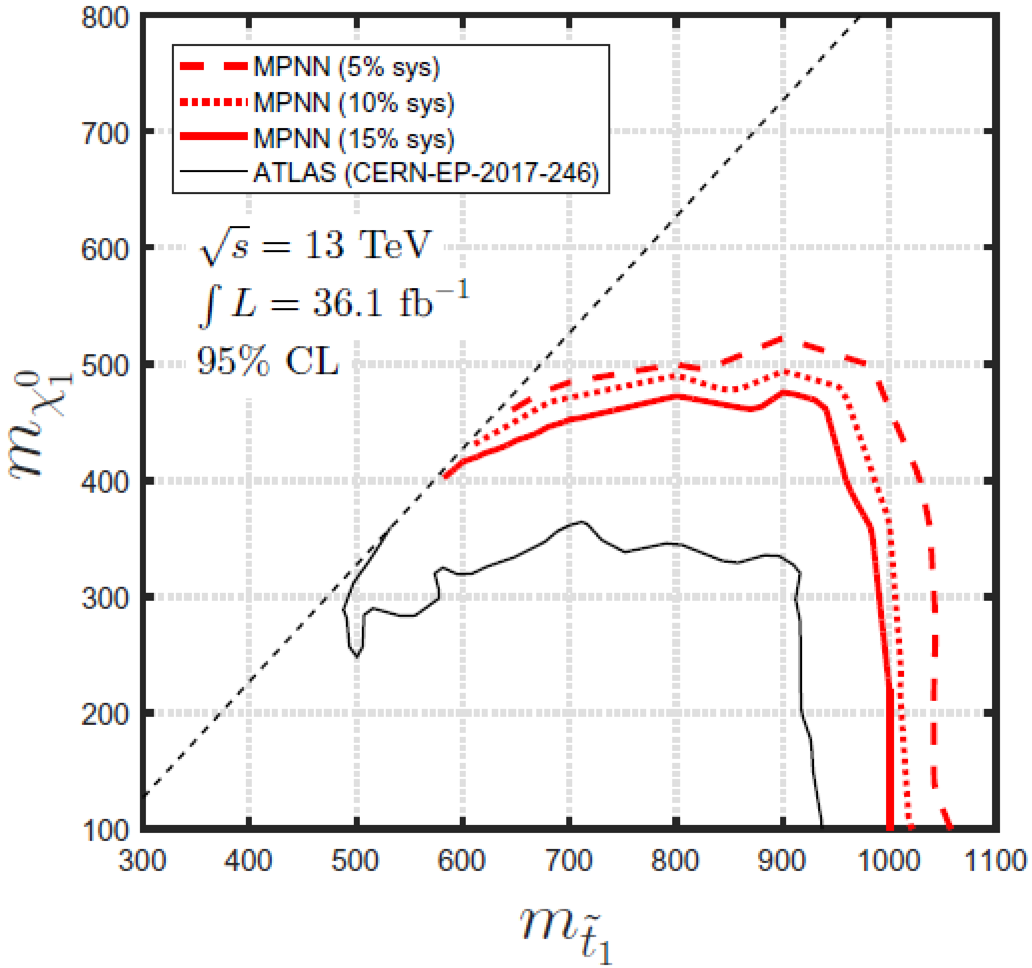


The first two principle components of node state vectors s_i^T of signal (red) and background (blue) events.

Benchmark	A	B
$m_{\tilde{t}_1}$ (GeV)	525	900
$m_{\chi_1^0}$ (GeV)	352	330
Pre-selection yield	380.5	44.9
ATLAS significance	2.0	2.0
MPNN significance	3.3	3.7
Improvement	65%	85%

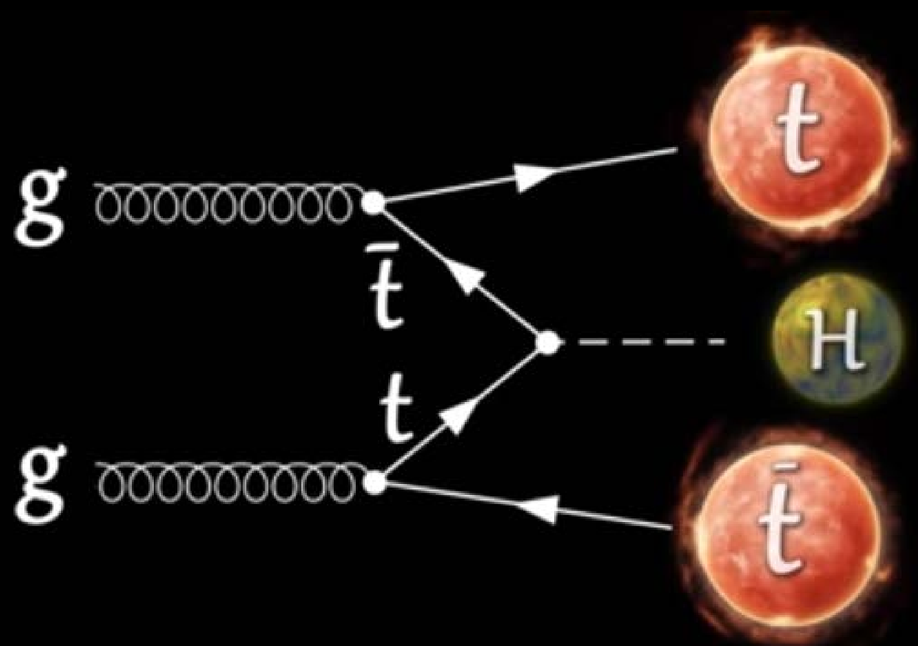
TABLE I. The comparison of MPNN with the available ATLAS results [37] for two benchmark points at 13 TeV LHC with the luminosity of $\mathcal{L} = 36.1\ \text{fb}^{-1}$.

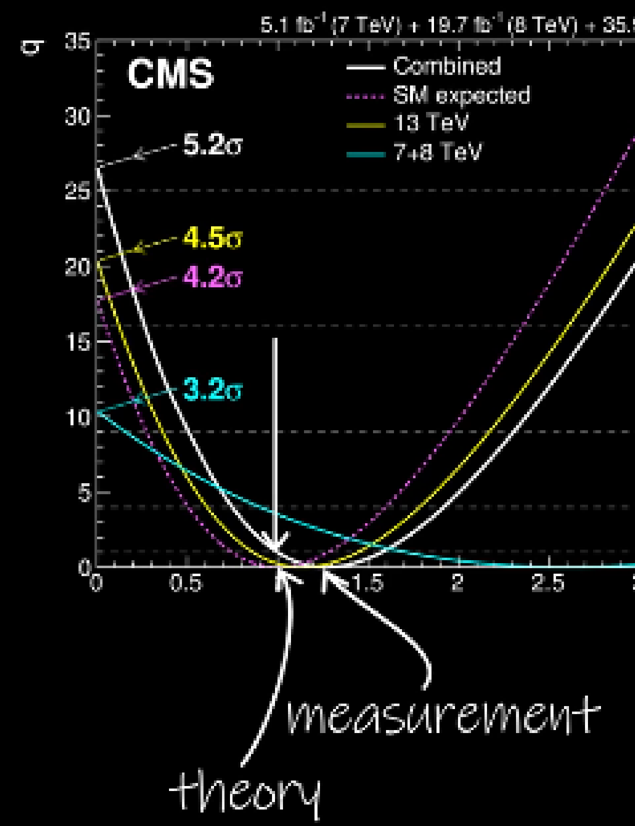
systematical uncertainty of backgrounds： 10%

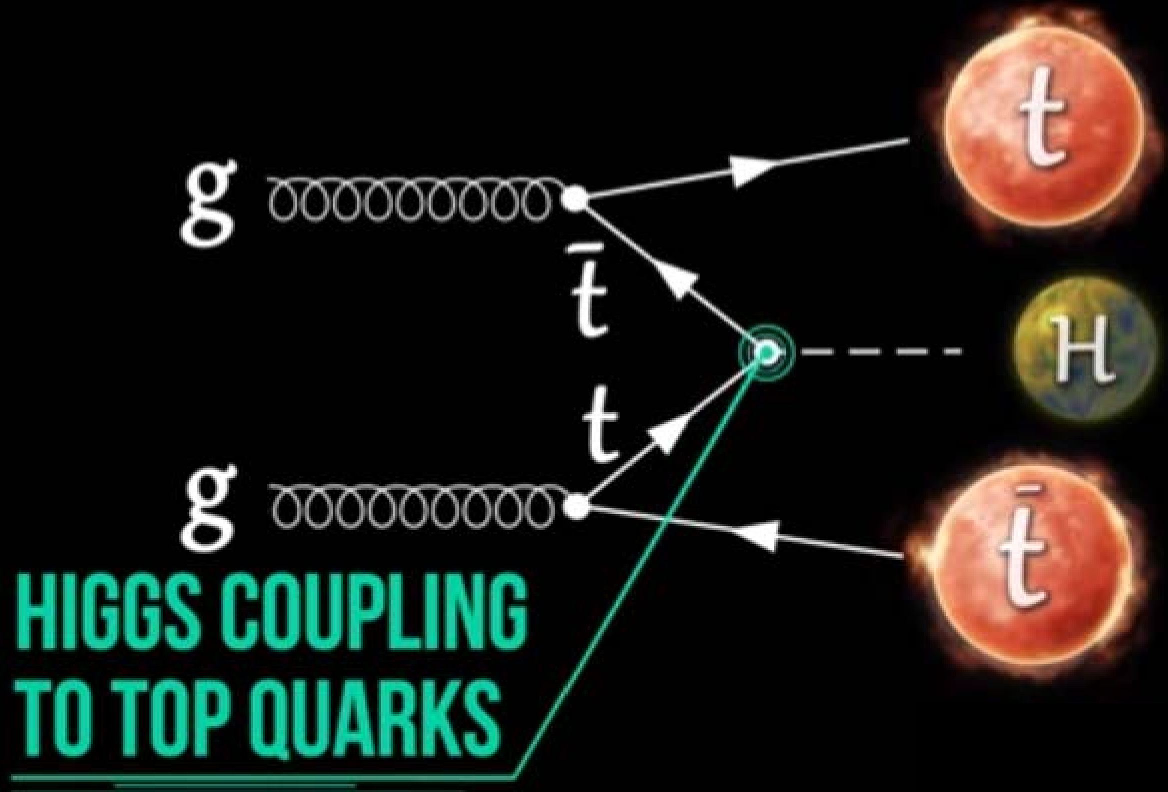


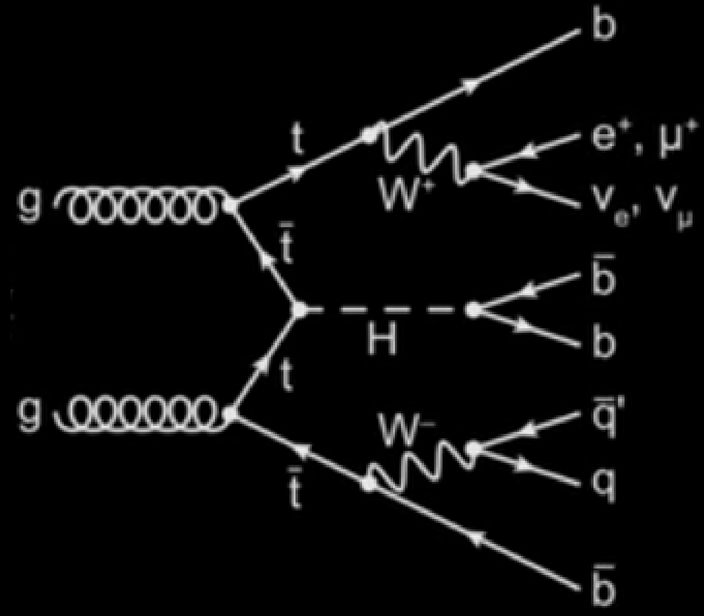
MPNN (5% sys)
MPNN (10% sys)
MPNN (15% sys)
ATLAS (CERN-EP-2017-246)
$\sqrt{s} = 13$ TeV
$\int L = 36.1$ fb^{-1}
95% CL
800
700
600
500
400
300
200
100
300
400
500
600
700
800
900
1000
1100
$m_{\chi_1^0}$
$m_{\tilde{t}_1}$

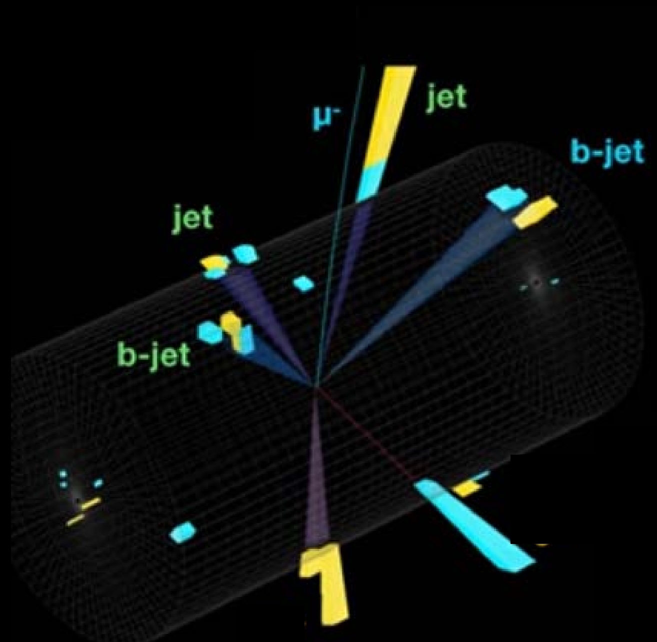
3 Graph neural network for $Ht\bar{t}$ at LHC






g
g
t̄
t
t
H
t̄
HIGGS COUPLING
TO TOP QUARKS

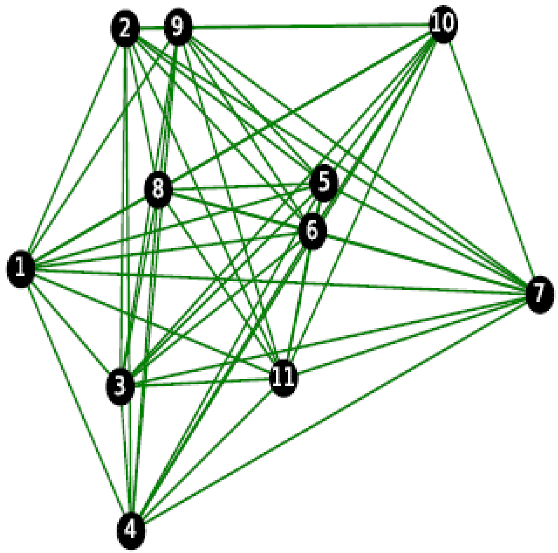

b
t
e⁺, μ⁺
g
W⁺
ν_e, ν_μ
t̄
b̄
H
b
t
g
W⁻
q̄'
q
t̄
b̄


μ⁻
jet
b-jet
jet
b-jet

$$pp \rightarrow t\bar{t}H \ (H = h) \rightarrow \bar{t}t\, b\bar{b}$$

$$pp \rightarrow t\bar{t}H \ (H = A) \rightarrow \bar{t}t\, b\bar{b}$$

$$pp \rightarrow t\bar{t}\ b\bar{b}\ \ \text{(background)}$$



x

1	0	-1	0	0	0.0160	0.0679	-0.0001
2	0	0	1	0	0.3265	0.6360	0.0587
3	0	0	1	0	0.1956	0.3934	0.0187
4	0	0	-1	0	0.1703	0.3179	0.0215
5	0	0	-1	0	0.1520	0.1605	0.0113
6	0	0	-1	0	0.1442	0.1491	0.0174
7	0	0	-1	0	0.0827	0.3333	0.0110
8	0	0	1	0	0.0452	0.0709	0.0068
9	0	0	1	0	0.0373	0.0519	0.0059
10	0	0	-1	0	0.0282	0.0552	0.0043
11	0	0	0	1	0.2154	0.2154	0.0000

d

	1	2	3	4	5	6	7	8	9	10	11
1	0.0000	2.8578	1.5566	3.1012	2.6385	2.3965	4.2066	1.4305	3.0275	4.4020	2.4592
2	2.8578	0.0000	2.2198	0.5844	2.3785	2.7535	4.5119	1.8490	0.4306	2.5726	2.6467
3	1.5566	2.2198	0.0000	1.6393	2.8377	2.3509	3.5506	2.2553	2.2532	3.3985	1.3257
4	3.1012	0.5844	1.6393	0.0000	2.8085	3.2339	4.2561	2.4220	0.6836	2.5808	2.1275
5	2.6385	2.3785	2.8377	2.8085	0.0000	0.5545	2.1564	1.3419	2.1250	2.0419	2.2379
6	2.3965	2.7535	2.3509	3.2339	0.5545	0.0000	1.9747	1.3322	2.5569	2.5738	1.6836
7	4.2066	4.5119	3.5506	4.2561	2.1564	1.9747	0.0000	3.3068	4.2135	3.1631	2.2780
8	1.4305	1.8490	2.2553	2.4220	1.3419	1.3322	3.3068	0.0000	1.8527	2.9715	2.3663
9	3.0275	0.4306	2.2532	0.6836	2.1250	2.5569	4.2135	1.8527	0.0000	2.1421	2.4522
10	4.4020	2.5726	3.3985	2.5808	2.0419	2.5738	3.1631	2.9715	2.1421	0.0000	2.6109
11	2.4592	2.6467	1.3257	2.1275	2.2379	1.6836	2.2780	2.3663	2.4522	2.6109	0.0000

FIG. 1. Event graph with detailed node features and edge weights for a specific simulated $t\bar{t}h$ event.

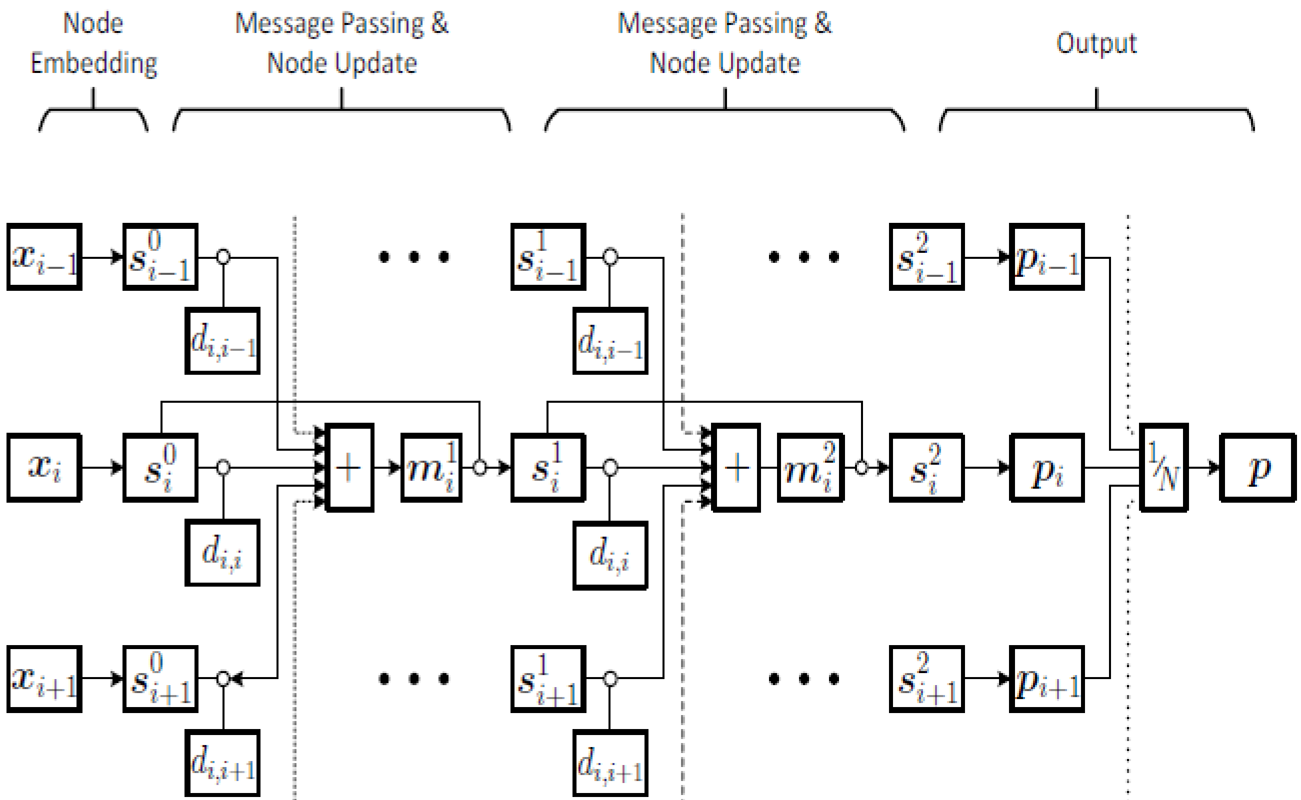


FIG. 2. The architecture of MPNN designed for classify $t\bar{t}h$, $t\bar{t}A$ and $t\bar{t}b\bar{b}$ events. It has one node embedding layer, two message passing and node update layers and one output layer. The small circles denote vector concatenation. The arrows denote applying non-linear functions. The summation and average run over all nodes.

For each event:

- each node i gives 3 probabilities $(p_i)_k$ for $t\bar{t}h$, $t\bar{t}A$ and $t\bar{t}b\bar{b}$
- average over all the nodes as the final output

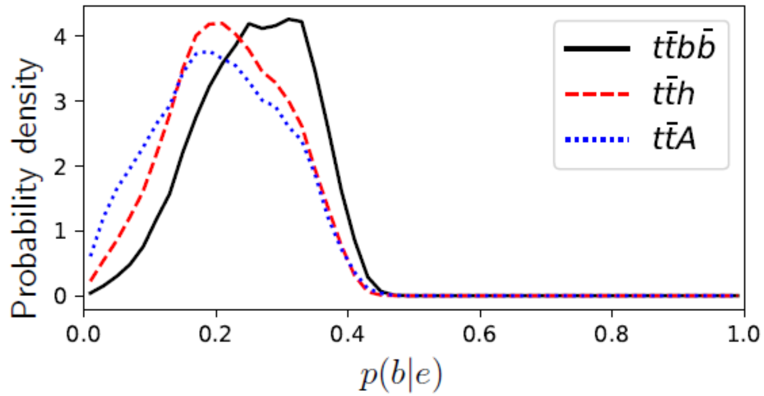
$$\frac{1}{N}\sum_i (p_i)_k \rightarrow \begin{cases} p(h|e) \\ p(A|e) \\ p(b|e) \end{cases}$$

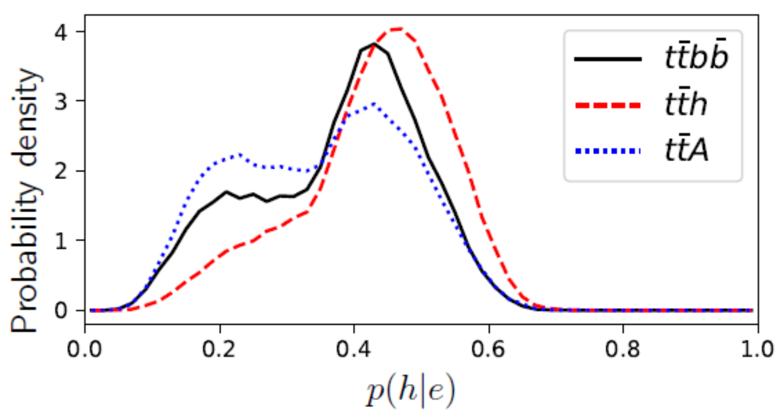
For each event sample D:

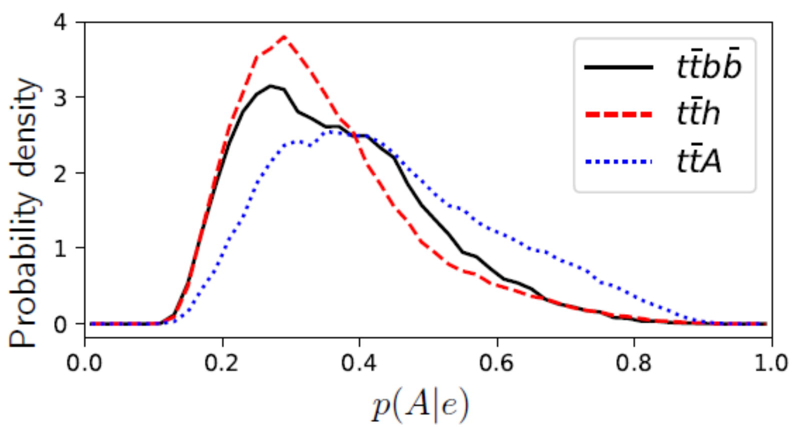
$$L_h(D) = \prod_{e\in D}{}' p(h|e)$$

$$L_A(D) = \prod_{e\in D}{}' p(A|e)$$

$$Q(D) = \frac{L_A(D)}{L_h(D)}$$







- each node i gives 3 probabilities $(p_i)_k$ for $t\bar{t}h$, $t\bar{t}A$ and $t\bar{t}b\bar{b}$
- average over all the nodes as the final output

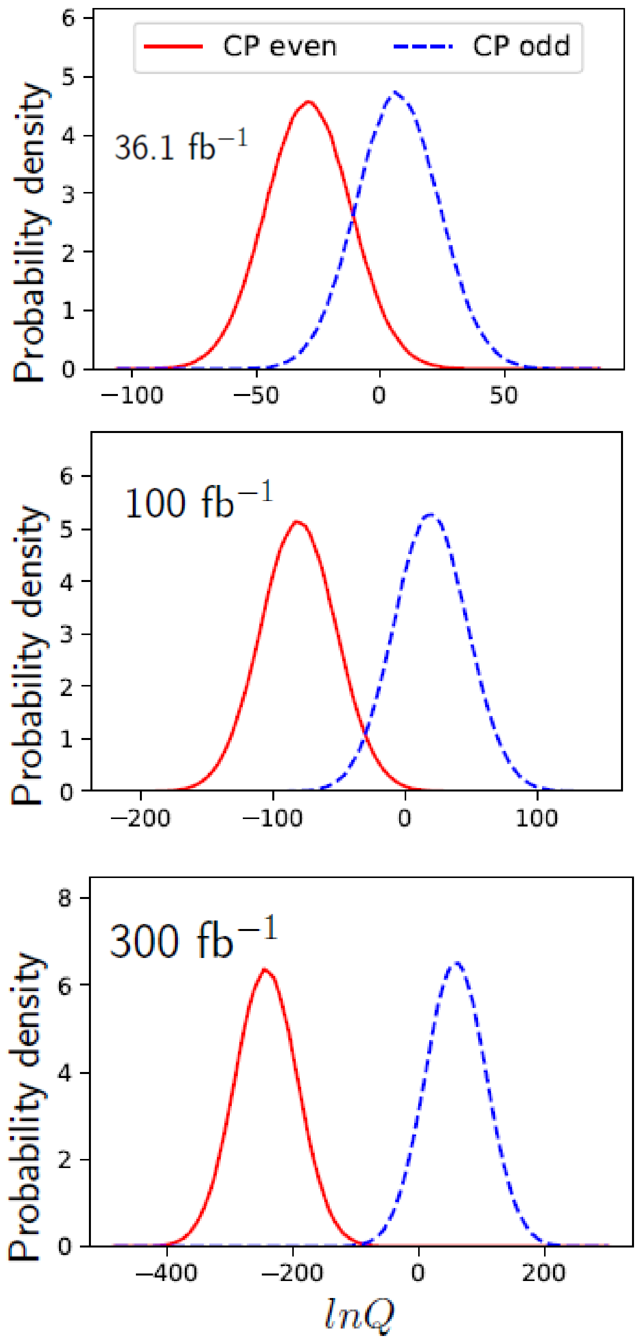
$$\frac{1}{N}\sum_i (p_i)_k \quad \begin{cases} p(h|e) \\ p(A|e) \\ p(b|e) \end{cases}$$

The MPNN has indeed learned some discriminative features for different processes:

The background $t\bar{t}b\bar{b}$ events tend to have higher p(b|e);

The $t\bar{t}h$ events tend to have higher p(h|e);

The $t\bar{t}A$ events tend to have higher p(A|e)



For each event:

- each node i gives 3 probabilities $(p_i)_k$ for $t\bar{t}h$, $t\bar{t}A$ and $t\bar{t}b\bar{b}$
- average over all the nodes as the final output

$$\frac{1}{N}\sum_i (p_i)_k \rightarrow \begin{cases} p(h|e) \\ p(A|e) \\ p(b|e) \end{cases}$$

For each event sample D:

$$L_h(D) = \prod_{e\in D}{}' p(h|e)$$

$$L_A(D) = \prod_{e\in D}{}' p(A|e)$$

$$Q(D) = \frac{L_A(D)}{L_h(D)}$$

The overlap between the two distributions reduces with increasing luminosity. When the luminosity is 300 fb^{-1} , the two distributions have nearly no overlap, which means that the CP nature of top-Higgs coupling can be determined.

Conclusion

We apply graph neural network to

- stop pair production at LHC

  to dig out stops from background

- $Ht\bar{t}$ production at LHC

  to distinguish CP-even h from CP-odd A

Thanks for your attention !