Graph Neural Network pour la reconstruction de traces dans le détecteur ATLAS

Sylvain Caillou
Laboratoire des 2 Infinis - Toulouse (L2IT)
sylvain.caillou@l2it.in2p3.fr

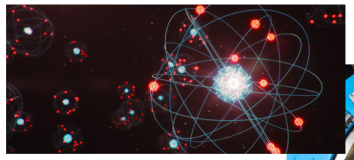
13e Journées Informatique IN2P3/IRFU, Nov 15 – 17, 2021

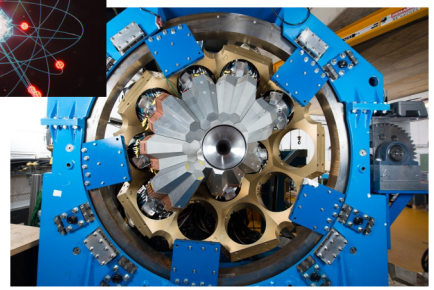
Quelques mots sur moi

- Sylvain Caillou, Ingénieur de Recherche en informatique au CNRS
  - Rentré au CNRS en 2005 en tant que IE
  - A rejoint le L2IT (IN2P3) en Septembre 2020
  - En poste depuis 2008 au LIMSI (INS2I, Institut des Sciences de l'Information et de leurs Interactions) à Orsay
  - Spécialiste des algorithmes d'intelligence artificielle et des librairies associées
  - Formateur en *machine learning* à l'IFSeM (formation CNRS Île de France), membre du comité d'expert pour l'établissement du nouveau catalogue de formation informatique de l'IFSeM 2022-2026

Le L2I - Toulouse, dernier né des laboratoires IN2P3

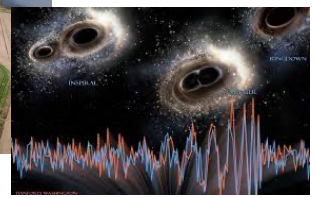
Physique nucléaire





GANIL

Ondes gravitationelles



VIRGO

données

données

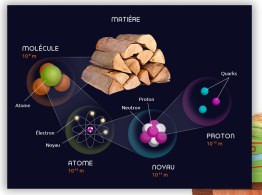
Ressources de calcul

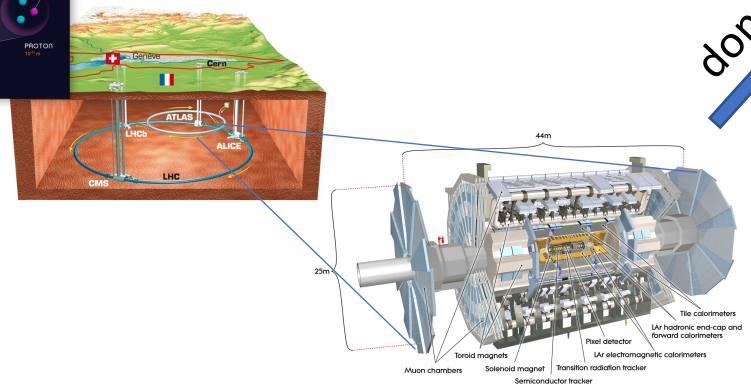
Retour d'expériences / beta tests

L2IT

CCIN2P3

Physique des particules





ATLAS

données

Laboratoire des 2 Infinis – Toulouse

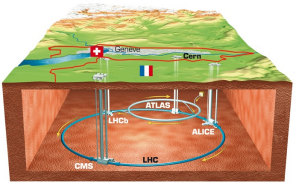
Activité pricipales:

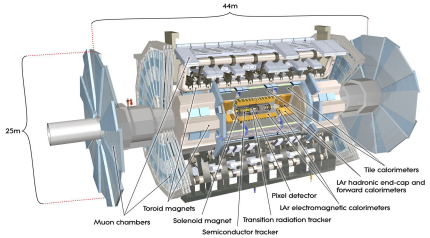
⇒ Analyse de données en physique

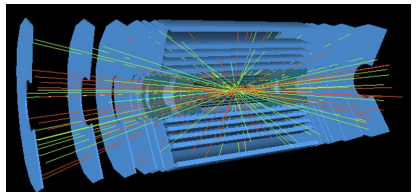
⇒ Développement d'algorithmes innovants, basés entre autres sur le machine learning

Le challenge de computing de HL-LHC

Large Hadron Collider (LHC) au CERN

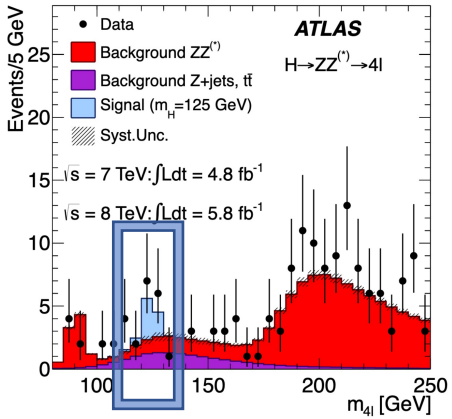


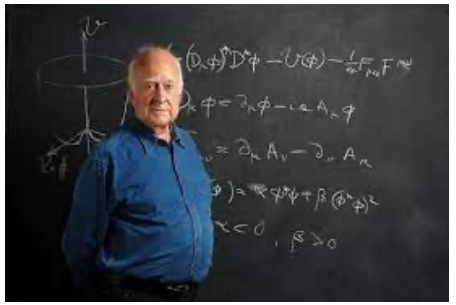




Collision de paquet de proton à très haute énergie pour créer des particules
40 millions de croisement de paquets de protons par seconde !

Découverte du boson de Higgs en 2012





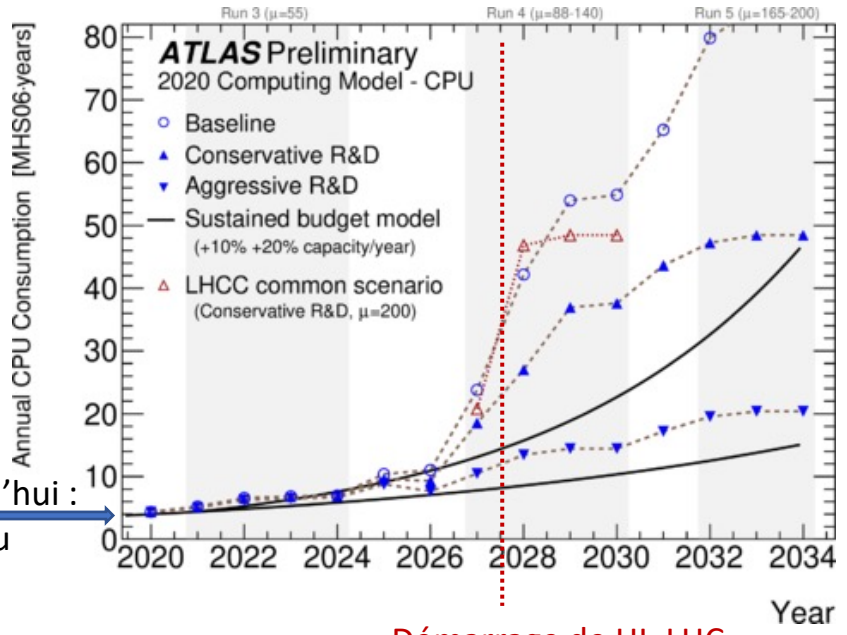
La prochaine grande étape pour la physique du Higgs au LHC

Etudier les propriétés du boson de Higgs!
HL-LHC => Intensité du faisceau multiplié par un facteur 10, démarre en 2027

=> Augmentation du volume et de la *complexité* des données

Avec notre modèle de *computing* actuel, il manquera un facteur important dans les ressources de calcul et de stockage
=> nécessité de changements importants
=> au risque de limiter le programme de physique

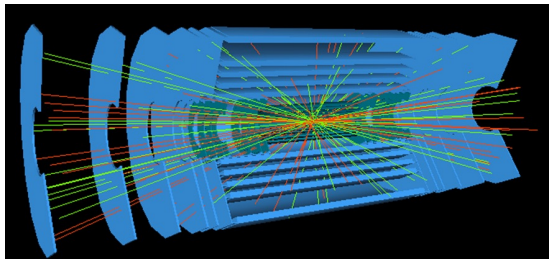
Ressources utilisées aujourd'hui :
- O(10^6) de cœurs en continu
- O(10^{12}) octet de stockage

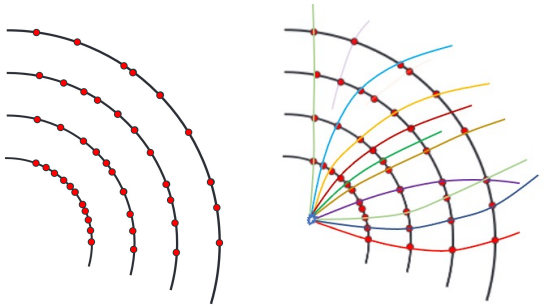


Le défi de HL-LHC pour la reconstruction de traces dans ATLAS

Les particules laissent des « Hits » dans les modules du détecteur

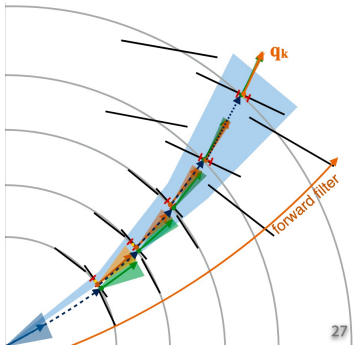
- Trouver les trajectoires (traces) des particules chargées produites dans les collisions p-p
- Nécessaire pour étudier les propriétés des particules produites et faire de la physique





Reconstruction actuellement opérée par des algorithmes basés sur des filtres de Kalman

- Estimation des paramètres de la trajectoire hélicoïdal
- Très bonne performance et optimisé depuis des années
- La partie **la plus coûteuse en complexité de calcul (donc en ressources CPU)** dans la reconstruction d'un événement



1. propagate p_{k-1} and its covariance C_{k-1} :

$$q_{k|k-1} = f_{k|k-1}(q_{k-1|k-1})$$
$$C_{k|k-1} = F_{k|k-1} C_{k-1|k-1} F_{k|k-1}^{T} + Q_k$$

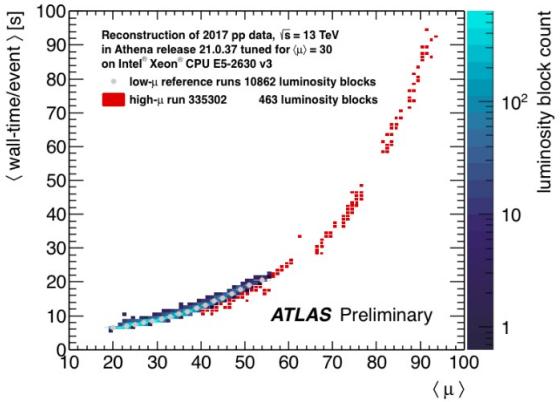
with Q_k ~ noise term (M.S.)

2. update prediction to get $q_{k|k}$ and $C_{k|k}$:

$$q_{k|k} = q_{k|k-1} + K_k[m_k - h_k(q_{k|k-1})]$$
$$C_{k|k} = (I - K_k H_k) C_{k|k-1}$$

with K_k ~ gain matrix :

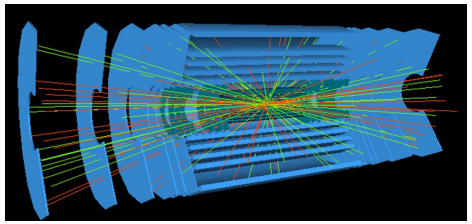
$$K_k = C_{k|k-1} H_k^{T} (G_k + H_k C_{k|k-1} H_k^{T})^{-1}$$



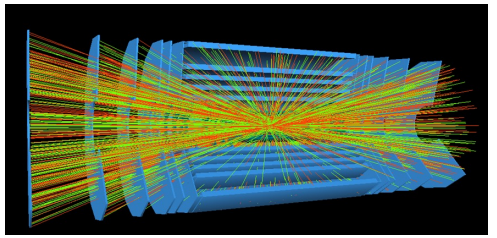
Démarrage de HL-LHC en 2027

- La combinatoire va exploser (pileup ~20 => pileup ~200)
- Augmentation très importante du volume et de la complexité des données (10^5 hits et 10^4 traces par évènement)
- Les algorithmes actuels ne suffiront pas

=> Il nous faut algorithme fondamentalement nouveau



LHC (pileup ~20)

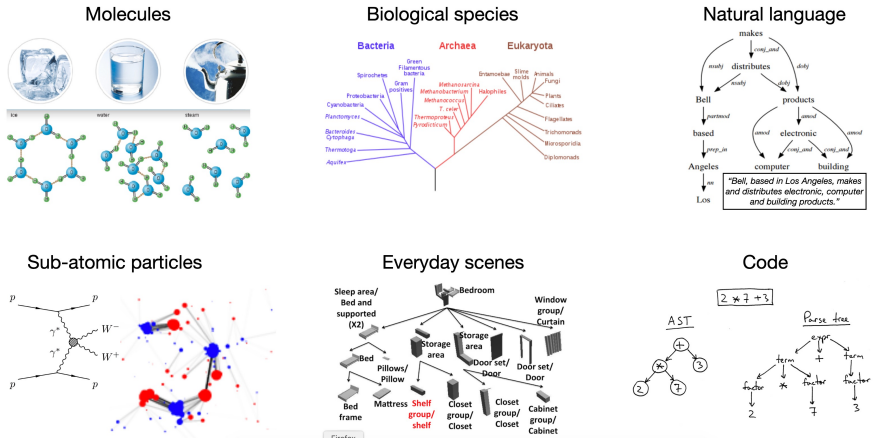


HL-LHC (pileup ~200)

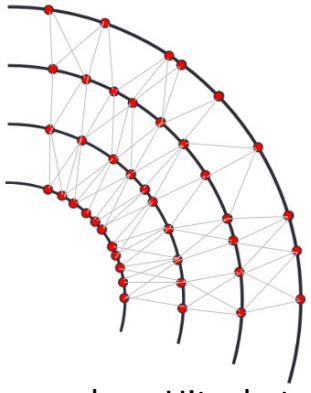
Imaginer un nouvel algorithme basé sur l'IA pour répondre au défi de HL-LHC

Peut-on utiliser l'IA pour aider à reconstruire les traces ?

Beaucoup de systèmes complexes sont structurés et peuvent être représentés sous forme de graphe

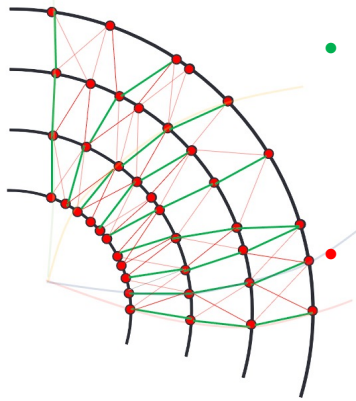


Représentation des données (les hits) sous forme de graphe



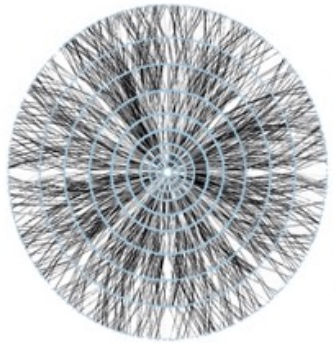
Nœuds du graphe = Hits du tracker
Arcs du graphe = Connexions considérées comme possibles

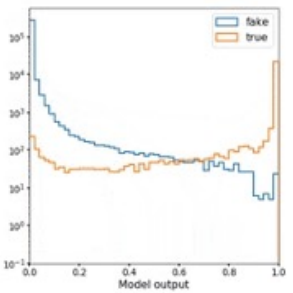
Ce que l'on voudrait faire :
Classifier les arcs du graphe



- Score haut = **forte probabilité** que l'arc appartiennent à une trace
- Score faible = **faible probabilité** que l'arc appartienne à une trace

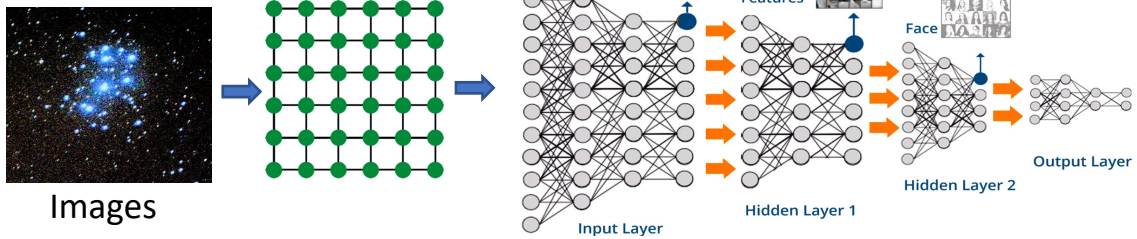
Exa.Tkrx project (2019 -)
« **Graph Neural Networks for Particle Reconstruction in High Energy Physics detectors** » arXiv:2003.11603
=> Entrainer des réseaux de neurones avec une architecture adaptée les **« Graph Neural Networks » (GNNs)**
=> Preuve de principe sur le dataset du TrackML Data Challenge





Les Graph Neural Networks (GNNs)

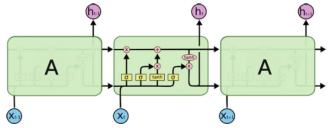
Convolutional Network CNN (~2012)



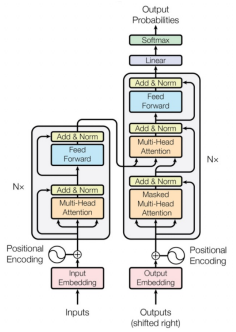
time series

«bla bla… »

Texte

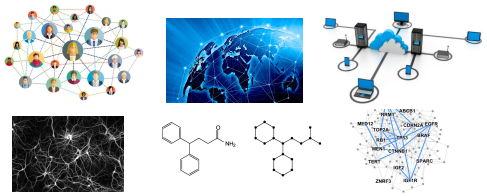


Recurent Neural Network (RNN, LSTM) (~2014)



Architectures Transformer

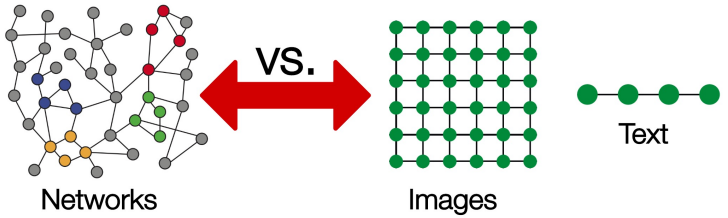
« **Attention Is All You Need** » (2017)



- Graphes permettent de représenter des données éparses avec des relations
- Les GNNs apprennent à partir des features des éléments du graphe mais également à partir des relations qui relient ces éléments entre eux

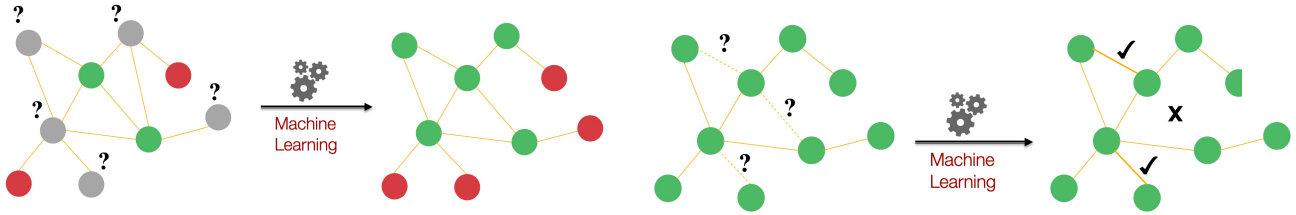
Apprentissage sur les graphes difficile

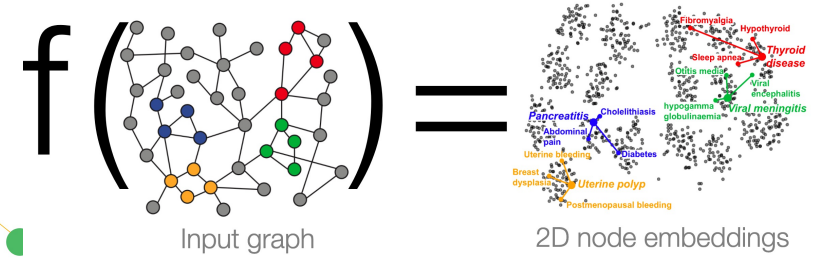
- Taille arbitraire et structure topologique complexe
- Pas de notion d'ordre ou de point référent



Quelles tâches avec un GNN ?

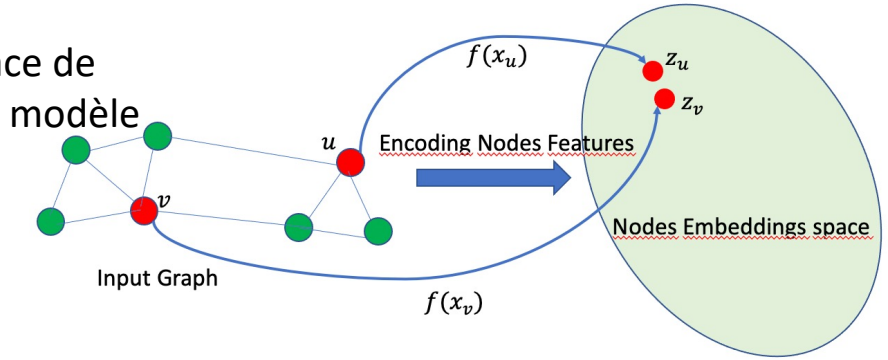
- Classification de nœuds
- Classification des arcs
- Prédiction de liens
- Classification de graphes





Les Graph Neural Networks (GNNs)

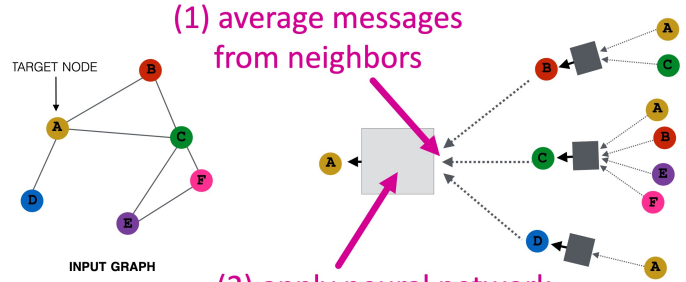
Encoder les nodes (ou les edges) du graphe de manière à ce qu'ils soient proches dans l'espace de projection si ils sont de la même classe et linéairement séparables par la dernière couche du modèle

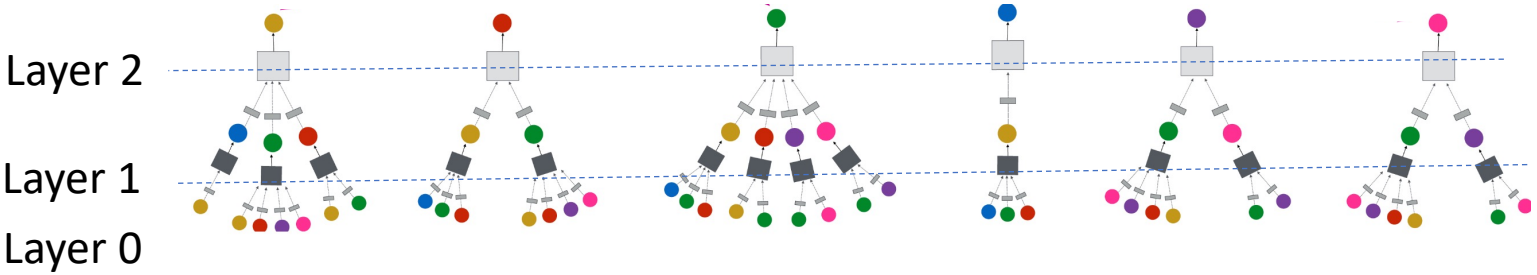


Pour i allant de 1 à N message passing:

Pour chaque nœud et/ou pour chaque arc du graphe:

1) Agréger les informations du voisinage
2) Appliquer un réseaux de neurones multicouches non linéaire





Trainable weight matrices (i.e., what we learn)

$$\mathrm{h}_v^{(0)} = \mathrm{x}_v$$

$$\mathrm{h}_v^{(l+1)} = \sigma\left(\mathrm{W}_l \sum_{u \in \mathrm{N}(v)} \frac{\mathrm{h}_u^{(l)}}{|\mathrm{N}(v)|} + \mathrm{B}_l \mathrm{h}_v^{(l)}\right), \forall l \in \{0, \ldots, L-1\}$$

$$\mathrm{z}_v = \mathrm{h}_v^{(L)}$$

Final node embedding

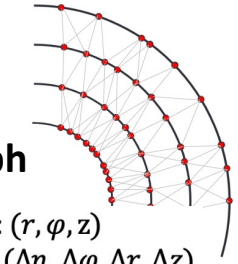
Des bibliothèques python pour construire des architectures GNN existent :

GraphNet (Tensorflow)

Deep Graph Library (DGL) (Tensorflow, Pytorch)

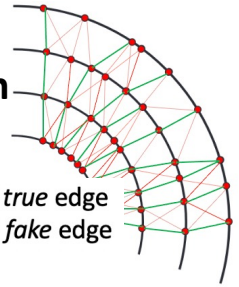
Pytorch Geometrics (PyG) (Pytorch)

Vers un algorithme de reconstruction basé sur les GNNs



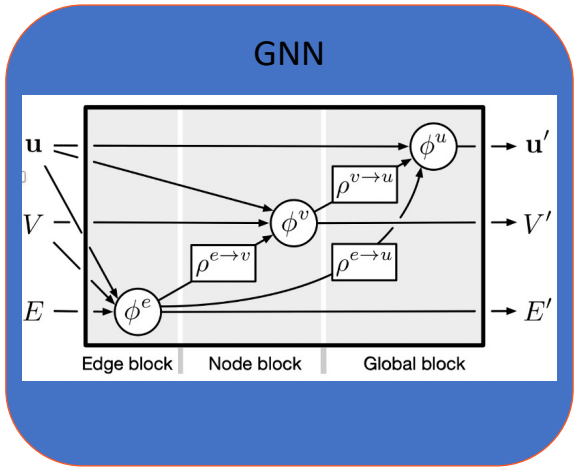
Input graph

Nodes features: (r, φ, z)
Edges features: $(\Delta\eta, \Delta\varphi, \Delta r, \Delta z)$

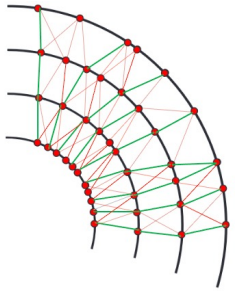


Target graph

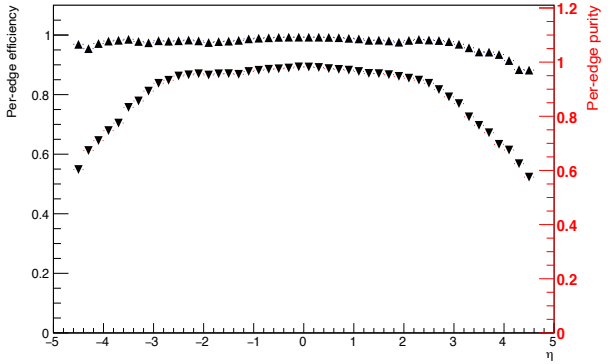
Target: { label = 1 if *true* edge
label = 0 if *fake* edge



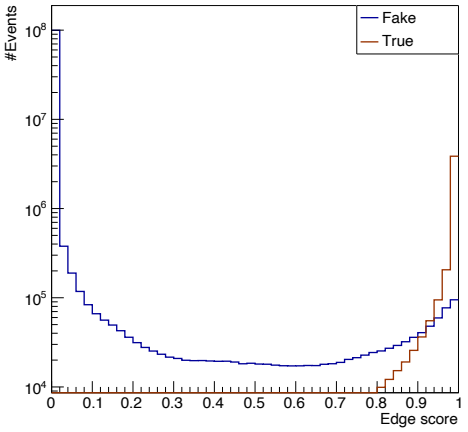
Predict graph



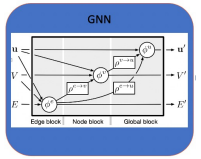
Edges scores

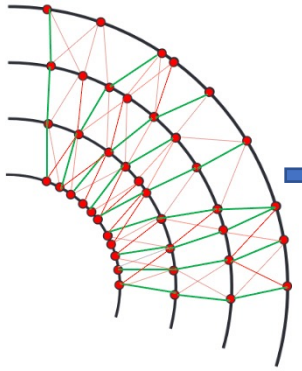


Edges classification

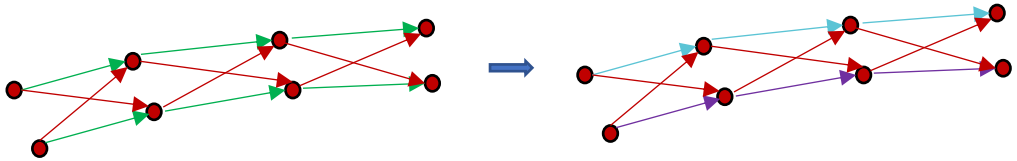


Predict graph

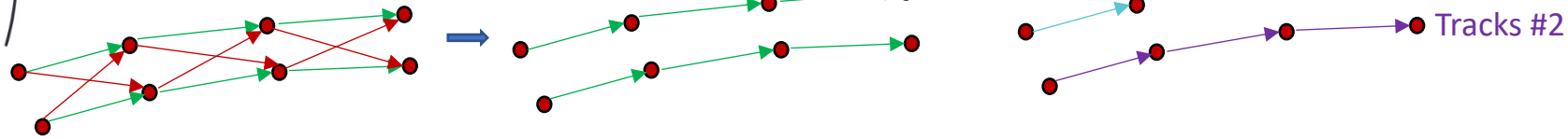


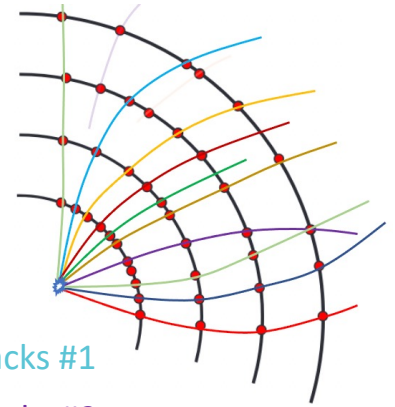


Algorithme de type « *walkthrough* »

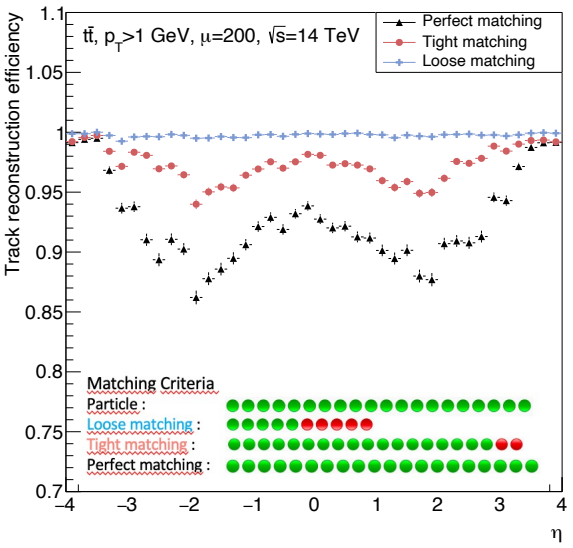


Algorithme de type « *labelling components* »





Premiers résultats et nouveaux challenges



=> Premier résultats présentés en session plénière à vCHEP 2021

« Towards a realistic track reconstruction algorithm based on graph neural networks for the HL-LHC » (2021)
Catherine Biscarat, Sylvain Caillou, Charline Rougier*, Jan Stark, Jad Zahreddine
arXiv:2103.00916 [physics.ins-det]

Depuis on est passé à des simulations de données hautement réalistes :

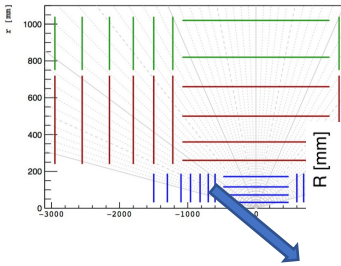
- pileup 200 + diffusion multiple + Ionisation
- Détecteur ITk (futur inner tracker d'ATLAS)

=> Graphes avec O(10^6) edges avec une haute densité et une topologie complexe

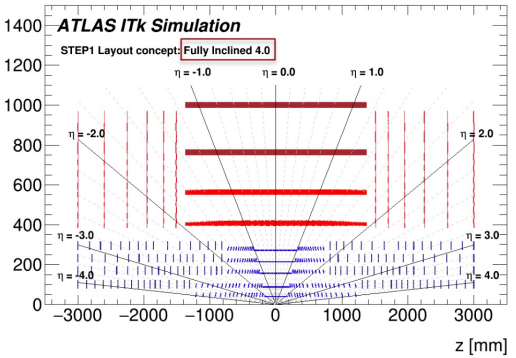
Le problème est dur à résoudre

=> On a besoin d'améliorer nos modèles

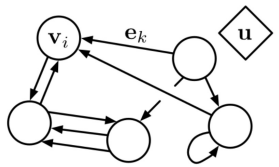
ACTS Generic detector

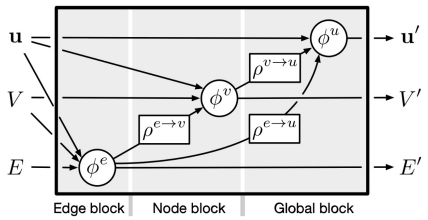


ATLAS ITk

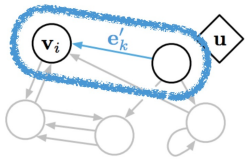


Comprendre et tester différentes architectures GNN

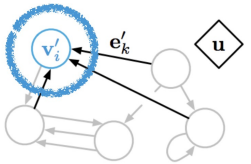




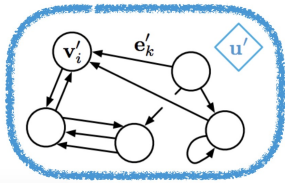
Edge block
For each edge, $\mathbf{e}_k, \mathbf{v}_{s_k}, \mathbf{v}_{r_k}, \mathbf{u}$, are passed to an "edge-wise function":
$\mathbf{e}'_k \leftarrow \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$

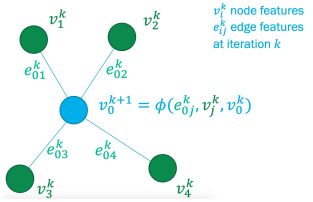


Node block
For each node, $\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}$, are passed to a "node-wise function":
$\mathbf{v}'_i \leftarrow \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$

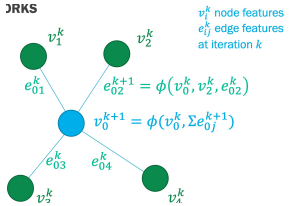


Global block
Across the graph, $\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}$, are passed to a "global function":
$\mathbf{u}' \leftarrow \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)$

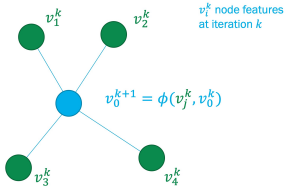




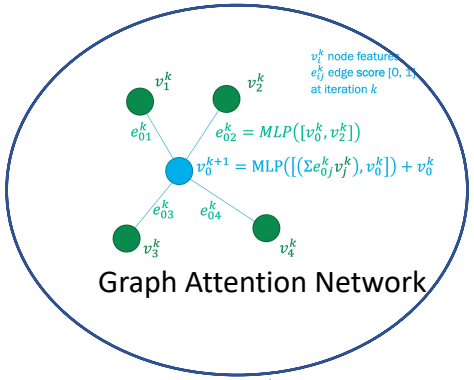
Graph Convolutional Network



Interaction Network



Message passing

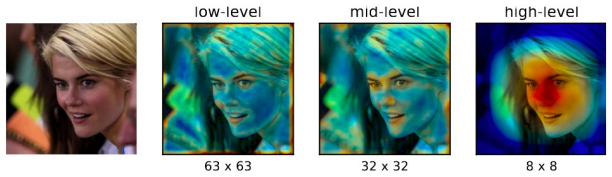


Graph Attention Network

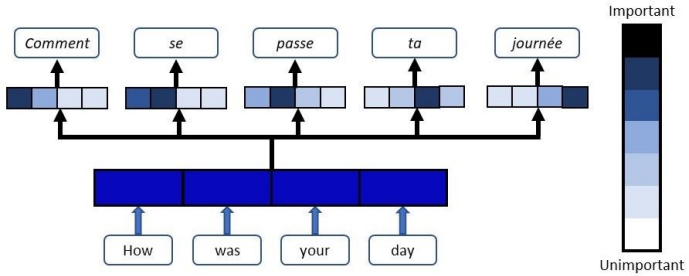
Beaucoup plus encore …

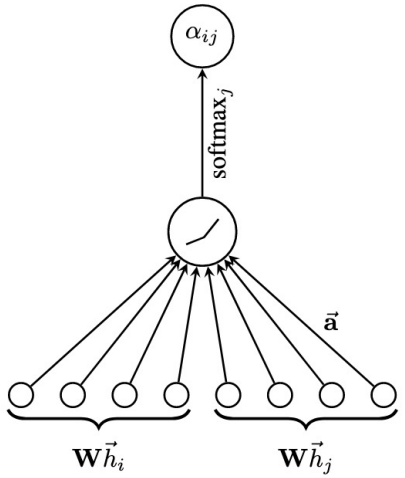
Modèles bioinspirés => Mécanismes d'attention

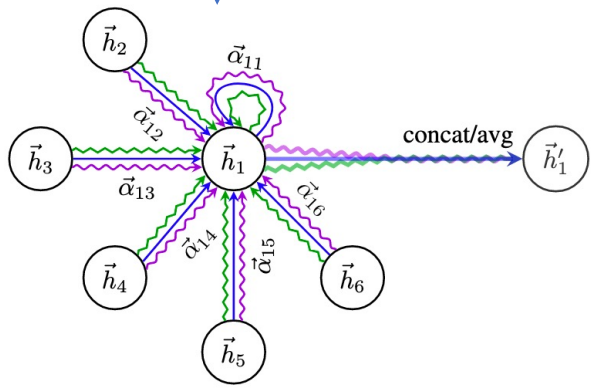
Attention spatiale dans les architectures CNN pour la classification ou la segmentation d'images



Attention séquentielle ou temporelle dans les architectures RNN ou les modèles transformer pour le traitement du language naturel ou les times series







Graph Attention Network (GAT)
Veličković *et al.*,arXiv:1710.10903 [stat.ML]

Out of Memory Error

Récurrence du GNN
Besoin d'accéder au N message passing dernier états du graphe pour mettre à jour les arcs => Nécessité de les garder en mémoire

N edge features
N edges [ ] x N message passing

Rétropropagation du gradient
Chaque arc défini un graphe computationel
Nécessité de garder en mémoire tous les états cachés à la sortie des couches linéaires

N edge features
N edges [ ] x (5xN MLP edge layers
+ 4xN MLP edge layers x N message passing)

Par exemple avec : message passing = 8, N edge features = 128, N edges = O(10^6), N MLP edge layers = 2
On a : 2 x ((2x8) x 10^6 x 128 x 8 + (2 + 4x8 + 2 + 1) x 10^6 x 128 x 8) => 0(100 Gb)!

Techniques investiguées

- IBM TFLMS et Pytorch LMS (swapping entre CPU host et GPU)
- Automatic Mix Precison (AMP) (passer en précision float16 pour les couches linéaires)
- Checkpointing (recalculer le gradient à la volée au moment de la backpropagation)
- stochastic training
- random sampling

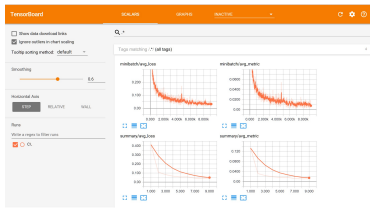
Environnement de développement du projet

Constrution des graphes
Boost Graph (C++)

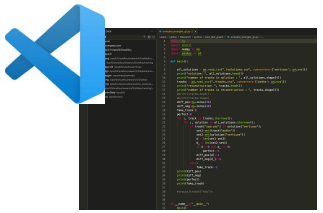
Entrainement GNNs
TensorFlow (Graphnet)
PyTorch (DGL, Pytorch Geometrics)

Opérations sur les graphes
NetworkX
Graph-tool
cuGraph

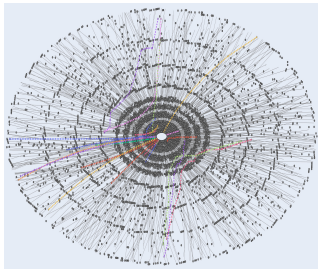
Visualisation entraînement
Tensorboard



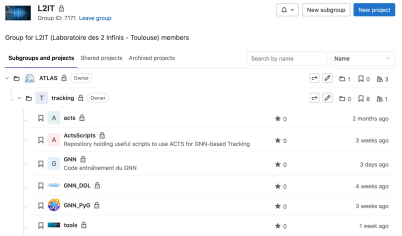
Developpement dans IDE (VSCode)



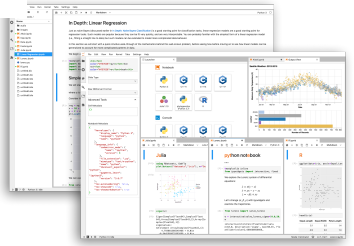
Visualisation données massives



Gestion de version



Analyse de données
Prototype de code



Fermes CPU et GPU

Besoins R&D spécifiques L2IT

2xRTX 8000 + Host CPU 1To RAM
4xV100 + Host 1To RAM

MERCI LE CC !

Travaux actuels

- Amélioration des performances des modèles GNN
- Amélioration de l'algorithme de reconstruction (régression sur les trajectoires hélicoïdales des particules)
- Comparaison en terme de temps de calcul et de performances de reconstruction avec le kalman d'Athena (ATLAS)
- Intégration dans ACTS (C++, boost graph, onnx)

MERCI ;-)