**Projet BELKA**

# Contexte

Le projet BELKA s’inscrit dans la recherche de nouveau médicament. Cette recherche se fait classiquement en plusieurs étapes :

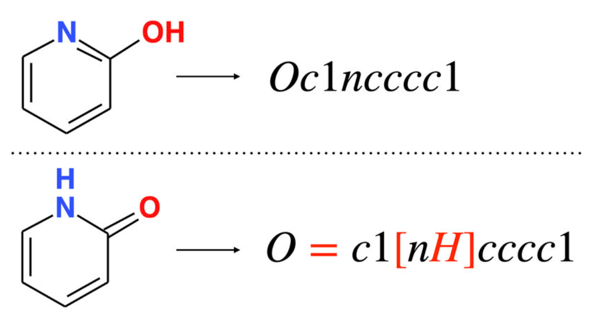
1. Choisir une maladie
2. Faire des dizaines d’années de recherche pour trouver quelle protéine cause cette maladie
3. Essayer de trouver quelles molécules vont inhiber cette protéine, et donc stopperont la maladie

Actuellement, ce sont les biologistes, par leur expertise métier, qui vont essayer de déterminer ces molécules et qui vont les tester manuellement. Ces tests se font en mettant chaque molécule candidate en contact de la protéine cible, dans une éprouvette. Toutefois il y en a des milliers de molécules candidates, c’est donc un processus qui est très long et donc très coûteux.

Et c’est là qu’intervient l’IA !

Dans le principe, si on connait les propriétés physico-chimiques des molécules, on pourrait prédire si elles vont se lier ou non à une protéine cible, ou prédire si elles vont être dangereuses ou non pour l’organisme.

Pour cela on va prendre la représentation la plus classique des molécules, ce qu’on appelle la forme SMILE, et qu’on peut écrire sous forme de string :



L’idée est ensuite de transformer cette forme SMILE de la meilleure manière possible, puis d’utiliser des modèles de ML ou de DL pour faire des prédictions.

# Projet

Rentrons dans le cœur du projet !

Je propose de faire un projet [Kaggle](https://www.kaggle.com/competitions/leash-BELKA/data), ce qui a l’avantage d’être un projet structuré, avec des objectifs clairs, une belle base de données, des discussions et des codes qui sont déjà partagés.

Ce projet s’intéresse à 3 protéines :

1. Une protéine impliquée dans la progression du diabète
2. Une protéine impliquée dans la progression du cancer
3. Une protéine présente dans le sang et qui empêche les médicaments d’être efficaces

L’objectif de ce projet est de prédire quelles molécules (candidates pour être des médicaments) peuvent se lier (donc interagir) avec une ces 3 protéines cibles.

# Base de données

* Train : 98 M molécules par protéine (donc 3 x 98 M = 295 M d’échantillons)
* Validation : 200 k molécules par protéine
* Test : 360 K molécules par protéine

Chaque molécule est divisée en 3 bloques (bleu, rose, gris), qu’on appelle *buildingblock* :

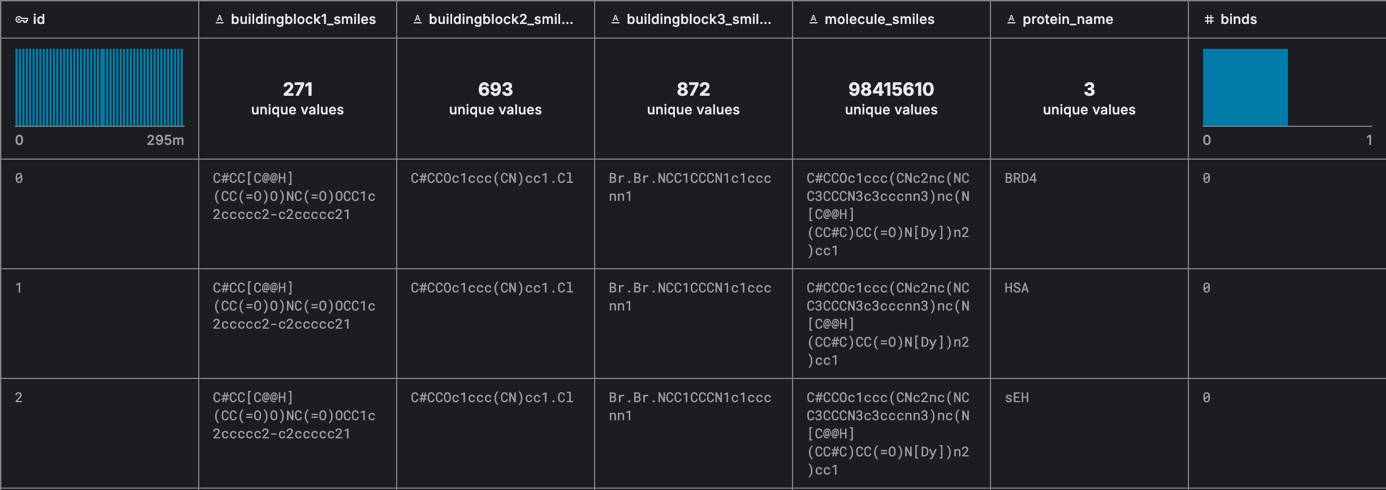
Une image contenant diagramme, cercle, conception

Description générée automatiquement

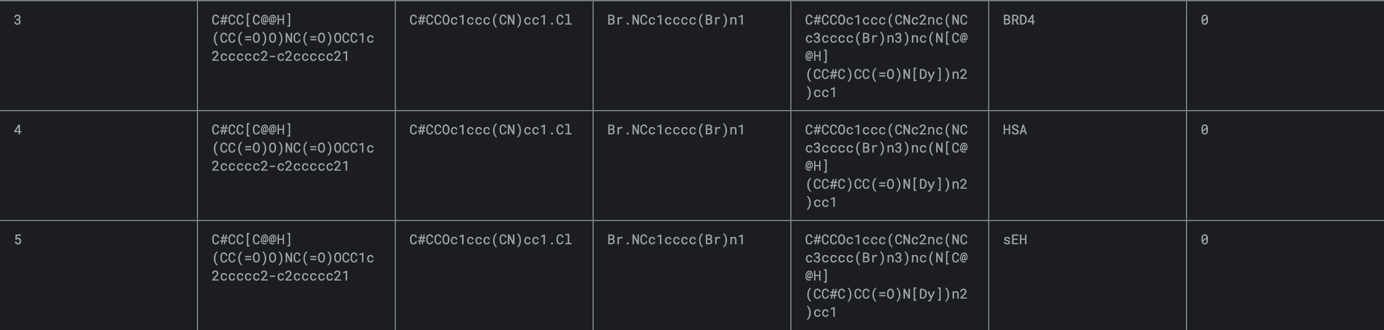
Exemple du fichier train, comportant 7 colonnes :

* *id* :
  + - Un exemple unique que nous utilisons pour identifier la paire molécule-cible de liaison.
* *buildingblock1\_smiles* :
  + - La structure, en SMILES, du premier bloc de construction.
* *buildingblock2\_smiles* :
  + - La structure, en SMILES, du deuxième bloc de construction.
* *buildingblock3\_smiles* :
  + - La structure, en SMILES, du troisième bloc de construction
* *molecule\_smiles* :
  + - La structure de la molécule entièrement assemblée, en SMILES. Elle comprend les trois blocs de construction et le noyau de la triazine. Notez que nous utilisons un [Dy] comme substitut pour l'élément de liaison de l'ADN.
* *nom\_de\_la\_protéine* :
  + - Le nom de la protéine cible (les 3 protéines sont : BRD4, HSA et sEH).
* *binds* :
  + - La colonne de la cible. Une étiquette de classe binaire indiquant si la molécule se lie à la protéine. Non disponible pour l'ensemble de test.

Pour une 1ère molécule :



Pour une 2nd molécule :



! Attention ! Les données sont fortement déséquilibrées donc il faudra surement les équilibrer durant l’entrainement du modèle. Sachant qu’il faudra aussi faire attention au *scaffolds* des molécules (la chaîne principale d'atome de carbone, sans prendre en compte les fonctions chimiques qui y sont attachés).

Objectif de fichier de soumission :



# Transformation

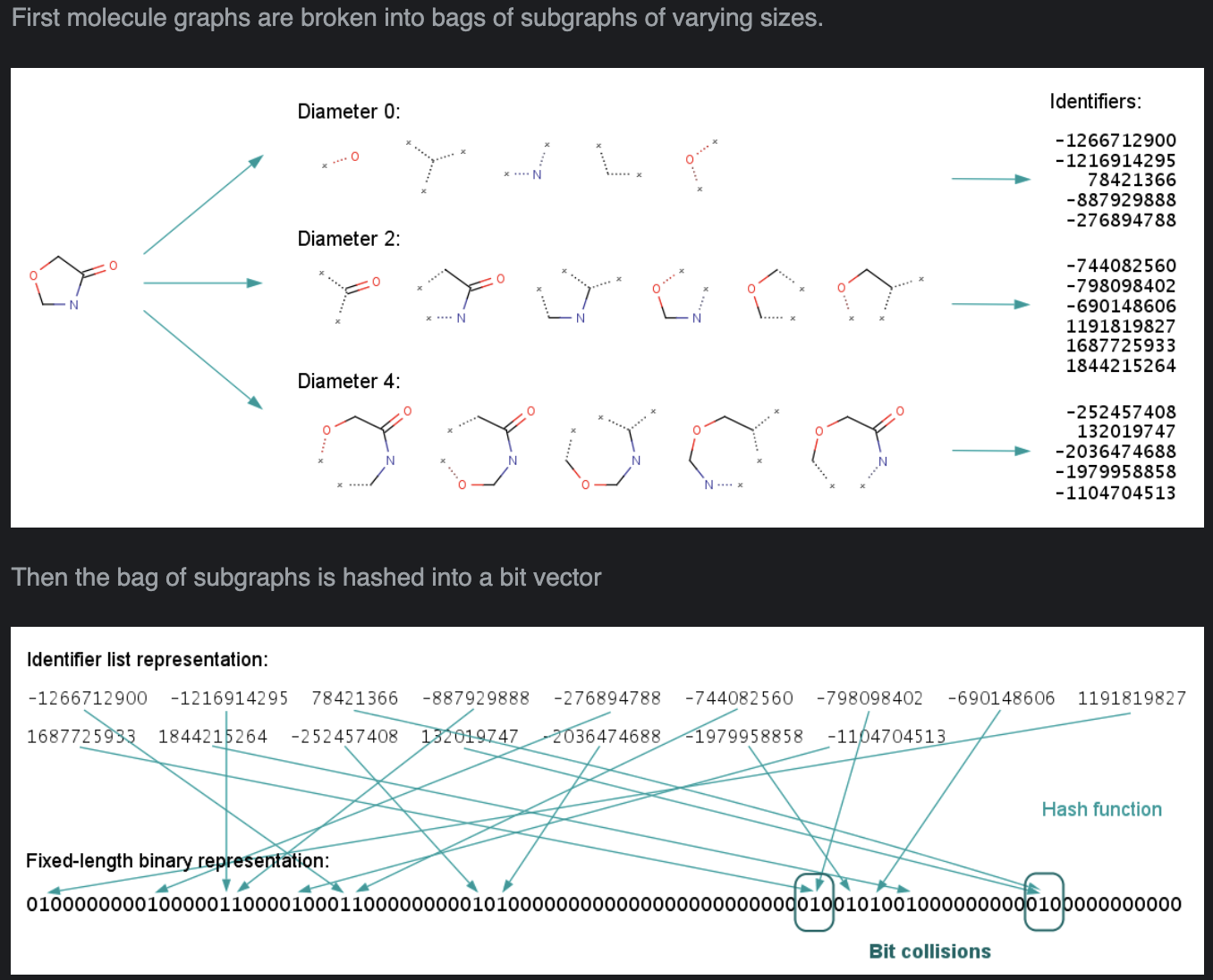
Les molécules sont représentées en format SMILES et peuvent être traduits dans d’autres formes de représentation chimique. Pour cela on utilise la librairie RDKit qui nous permet de générer par exemple 4 formats :

**Les Descripteurs Physico-Chimiques :**

Ce sont les propriétés telles que le poids moléculaire, le nombre d'atomes de carbone, le nombre de liaisons hydrogène, la surface l'interaction, etc.

**Les Fingerprints (**ECFPs)**:**

Ce sont des vecteurs binaires ou de bits qui capturent la présence ou l'absence de sous-structures chimique spécifiques dans la molécule.

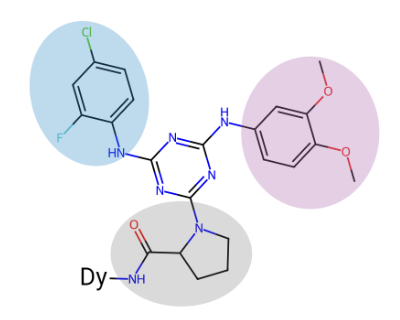


**Les Vecteurs Embeddings :**

Représentations continues et denses de molécules, où chaque molécule est représentée par un vecteur de nombres réels. Ils sont générés par des transformateurs, RNN.

**Les Graphiques :**

Utiliser la représentation graphique des molécules.



# Prédiction

Une fois qu’on a nos données transformées, nous pouvons utiliser des modèles de ML ou DL adaptés à nos entrées pour prédire si les molécules peuvent se lier aux protéines cibles.

* Pour les descripteurs Physico-Chimiques et les Fingerprint :
  + - *Linear Classifier, SVM, K-NN, RandomForestClassifier*
* Pour les Vecteurs Embedding :
  + - *RNN, Transformers*
* Pour les Graphiques :
  + - *GNN (voir annexes)*

# Rendu Final

L’idée est d’aller étape par étape, du plus simple au plus complexe, et de s’arrêter quand on n’aura plus de temps.

Par exemple :

1. Utiliser un Linear Classifier sur des données physico-chimique.
2. Utiliser un Linear Classifier sur des données Fingerprint.
3. Utiliser un RandomForest Classifier ou des BoostedTrees Classifier (AdaBoost ou Gradient Boosting) sur les données Fingerprint.
4. Utiliser des RNN ou des Transformers (encoding) pour générer des vecteurs Embedding pour les utiliser en input de RNN et de Transformers (deconding) pour faire notre prédiction.
5. Utiliser une représentation graphique des molécules et utiliser un GNN (encoding-decoding) pour faire notre prédiction.

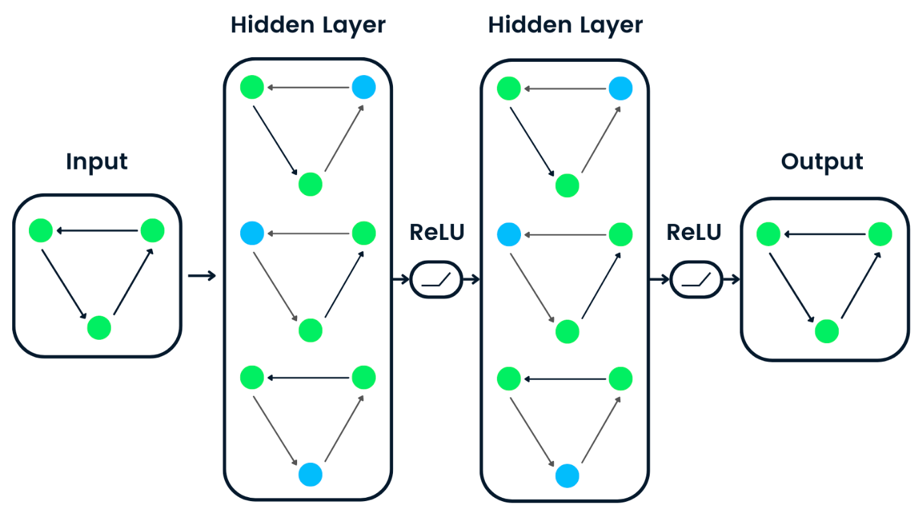
# Annexes

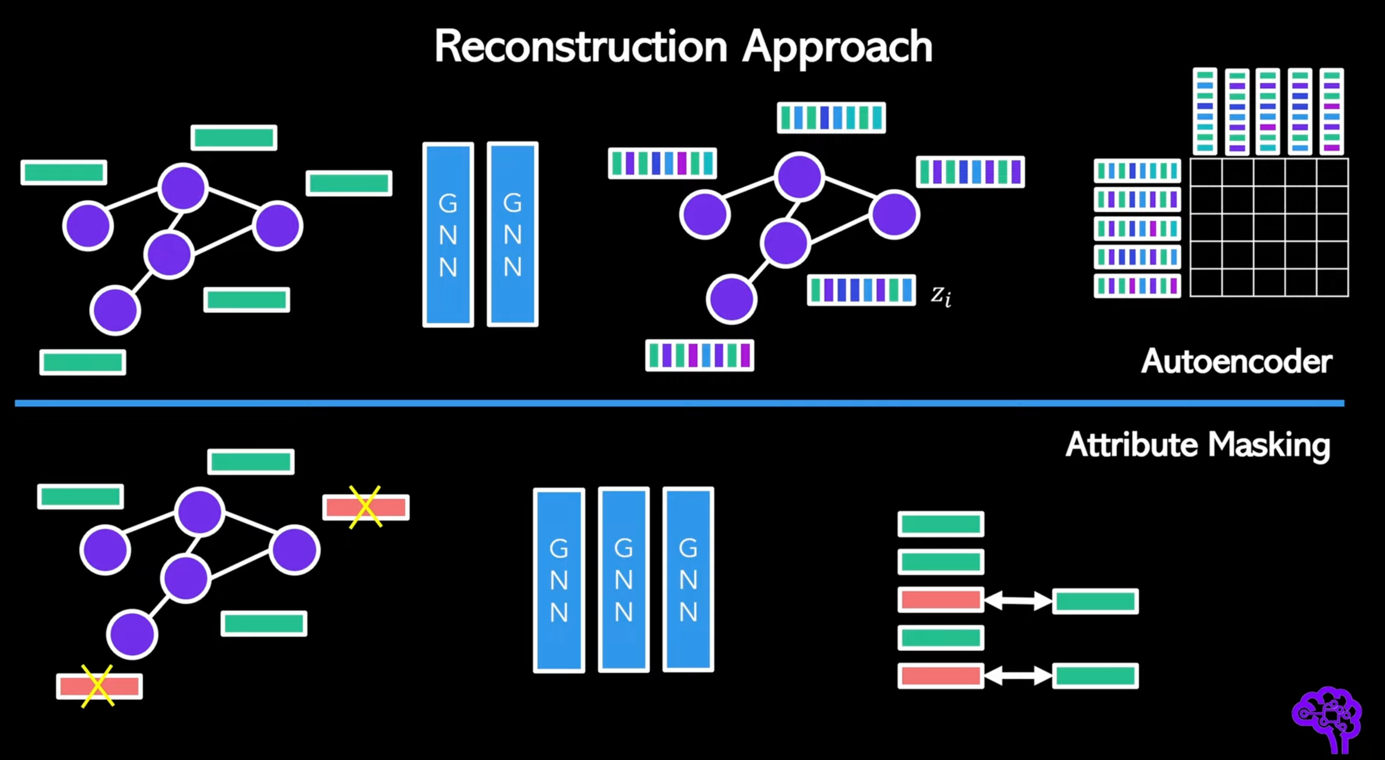
**Graph Neural Networks (GNN) :**

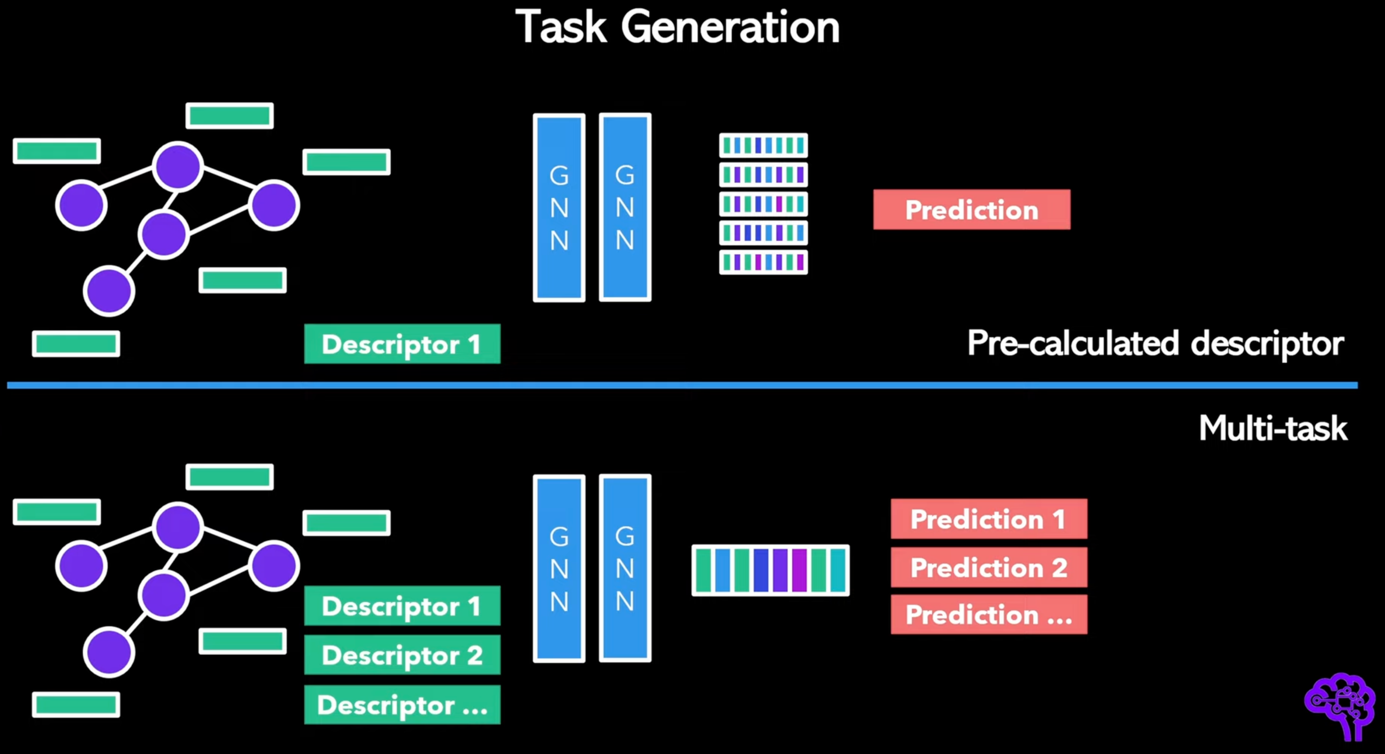
Applique des convolutions sur les graphes moléculaires pour créer des Embeddings des molécules en tenant compte de leur structure topologique.

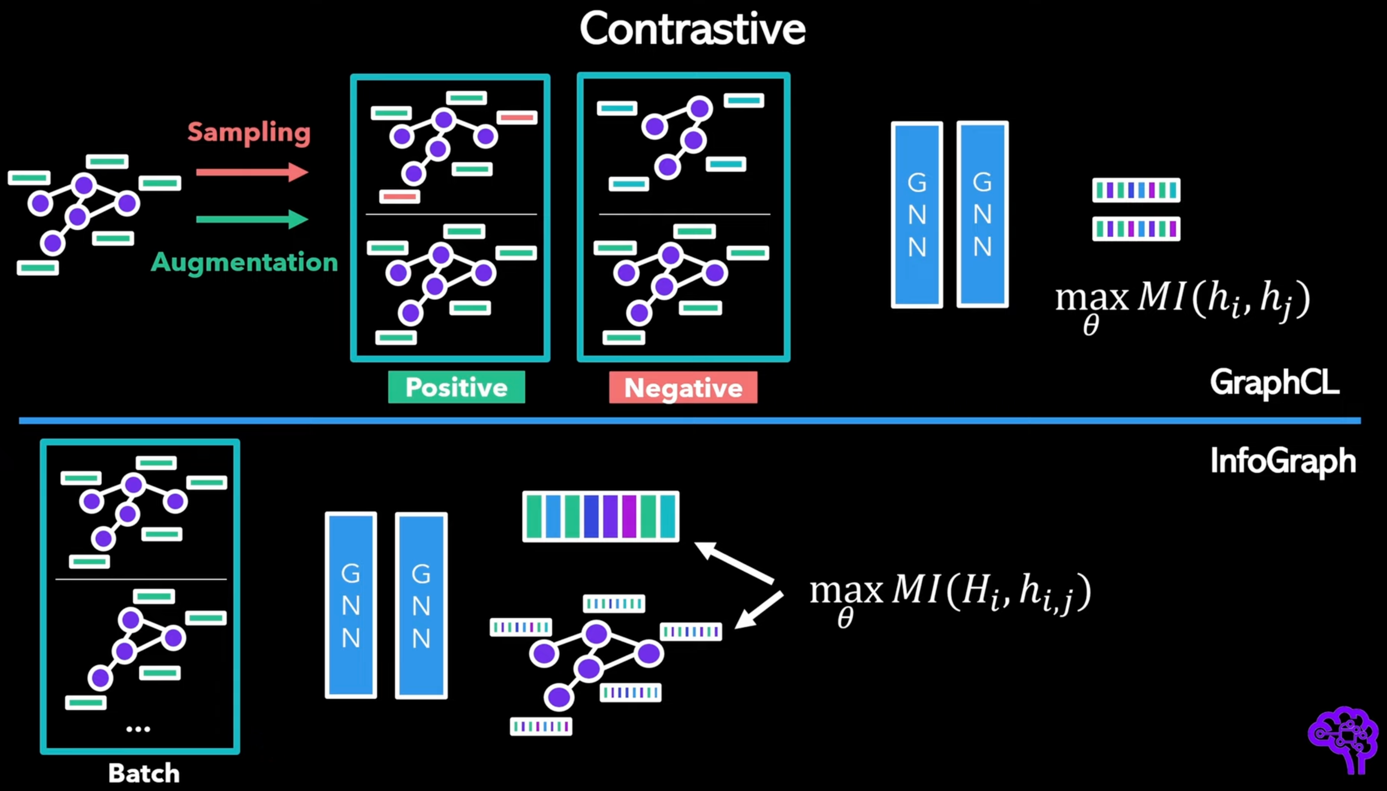
[Graph Neural Networks - a perspective from the ground up](https://www.youtube.com/watch?v=GXhBEj1ZtE8&t=730s)

[A Comprehensive Introduction to Graph Neural Networks (GNNs)](https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial)









Similitude :

