

Réseaux de neurones

IFT 780

Réseaux de neurones par graphe

Par
Pierre-Marc Jodoin

1


Documentation

Wu et al. A **Comprehensive Survey on Graph Neural Networks**.
IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. 32, NO. 1, 2021
<https://arxiv.org/pdf/1901.00596.pdf>

W. Hamilton **Graph Representation Learning**, Morgan & Claypool
www.cs.mcgill.ca/~wlu/grl_book/files/GRL_Book.pdf

Michael M. Bronstein, Joan Bruna, Taco Cohen, Petar Veličković, **Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges**, [arXiv:2104.13478](https://arxiv.org/abs/2104.13478)
geometricdeeplearning.com/

Petar Velićkovic, *Everything is Connected: Graph Neural Networks*
<https://arxiv.org/pdf/2301.08210.pdf>



<https://www.youtube.com/playlist?list=PLNfyQfwJuLqBqXVqPF2UvILRqQF1uO4PM>

2

Les graphes sont partout


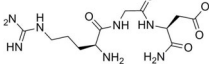
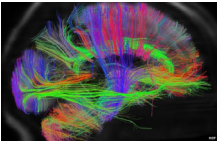



Image by David Foray and Vincent Bevan-Powles



pieryl.com



montreal **map**

Scale: 1:100,000

Legend:

- 1. Districts
- 2. Major roads
- 3. Minor roads
- 4. Water bodies
- 5. Parks and green spaces
- 6. Public transit stations
- 7. Landmarks
- 8. Other points of interest

3

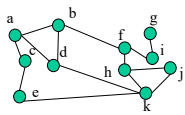
Les graphes sont partout

De plus en plus utilisés grâce à un riche écosystème de bibliothèques



4

Concepts fondamentaux



Un graph (\mathcal{G}) est constitué de nœuds (\mathcal{V}) et d'arêtes (\mathcal{E}) :

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

$$\mathcal{V} = \{a, b, c, \dots, k\}$$

$$\mathcal{E} = \{ab, ad, bf, \dots, kj\}$$

Matrice d'adjacence $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$

5

Matrice d'adjacence

$A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ tel que $a_{ij} = \begin{cases} 1, & (i, j) \in \mathcal{E} \\ 0, & \text{sinon} \end{cases}$

$$A = \begin{pmatrix} & i \\ & \vdots \\ a_{ij} & \dots \dots \\ & j \end{pmatrix}$$

$|\mathcal{V}|$: nombre de nœuds dans le graphe

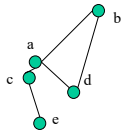
A est symétrique pour tout graphe non orienté

6

6

Concepts fondamentaux

Exemple



$$A = \begin{pmatrix} & \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} & \mathbf{e} \\ \mathbf{a} & 1 & 1 & 1 & 1 & 0 \\ \mathbf{b} & 1 & 1 & 0 & 1 & 0 \\ \mathbf{c} & 1 & 0 & 1 & 0 & 1 \\ \mathbf{d} & 1 & 1 & 0 & 1 & 0 \\ \mathbf{e} & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

7

7

Différentes configurations

Orienté vs non orienté

- Non orienté : $A = A^T$
- Orienté : A est assymétrique

Ex: dans un réseau social, les relations d'amis peuvent être bidirectionnelles ou non.

Poids: Une arête peut avoir un poids $w_{ij} \in \mathbb{R}$

Ex: dans une molécule, un poids peut être la force d'attraction entre 2 atomes.

Caractéristiques: les nœuds/arêtes peuvent avoir des caractéristique $\vec{x}_i \in \mathbb{R}^d$

Ex: dans une molécule, un noeud peut contenir le nombre d'électrons, protons, neutrons (donc 3 caractéristiques).

Multi-relationnel: différents types d'arêtes $(i, j, t) \in \mathcal{E}$

Ex: dans un graph social, t est la relation (parent, époux, etc)

Hétérogène: différents types de nœuds

Ex: dans un réseau d'interactions sociales (personnes, institutions, objets, etc)

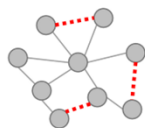
8

Différentes tâches

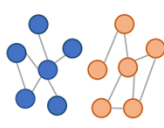
Node Classification



Link Prediction



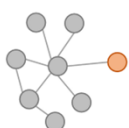
Graph Classification



Community Detection



Anomaly Detection

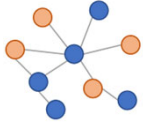


<https://www.anddata-science.com/graph-neural-networks-with-pytorch-on-node-classification-link-prediction-and-anomaly-detection-14a38f61275>

9

Différentes tâches

Node Classification



Identifier les **personnes** intéressées
par la **science fiction** en analysant leurs
amitiés dans un **réseau social**

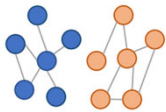
Graphe
Nœuds
Arrêtes
Étiquettes

<https://www.audiodatascience.com/graph-neural-networks-with-pytorch-on-node-classification-link-prediction-and-anomaly-detection-14a38f61275>

10

Différentes tâches

Graph Classification



Déterminer si une **molécule** est **soluble** dans
l'eau basé sur les **atomes** qui la constitue
et des **liens entre eux**.

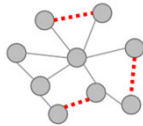
Graphe
Nœuds
Arrêtes
Étiquettes

<https://www.audiodatascience.com/graph-neural-networks-with-pytorch-on-node-classification-link-prediction-and-anomaly-detection-14a38f61275>

11

Différentes tâches

Link Prediction



Déterminer dans un **atlas** routier si une
rue entre deux **endroits** est **bloquée** (ou
non)

Graphe
Nœuds
Arrêtes
Étiquettes

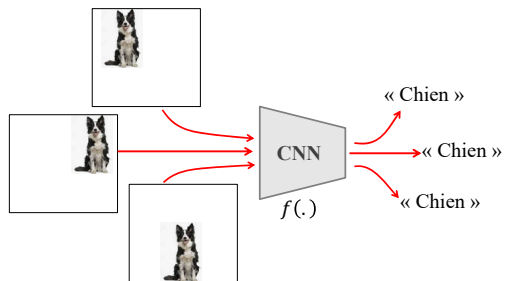
<https://www.audiodatascience.com/graph-neural-networks-with-pytorch-on-node-classification-link-prediction-and-anomaly-detection-14a38f61275>

12

Invariance et équivariance

13

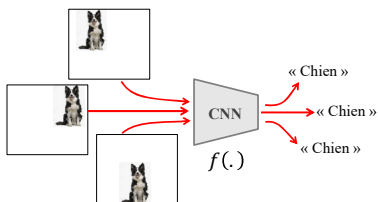
Les CNNs sont invariants à la translation



14

Les classifieurs CNNs sont invariants à la translation

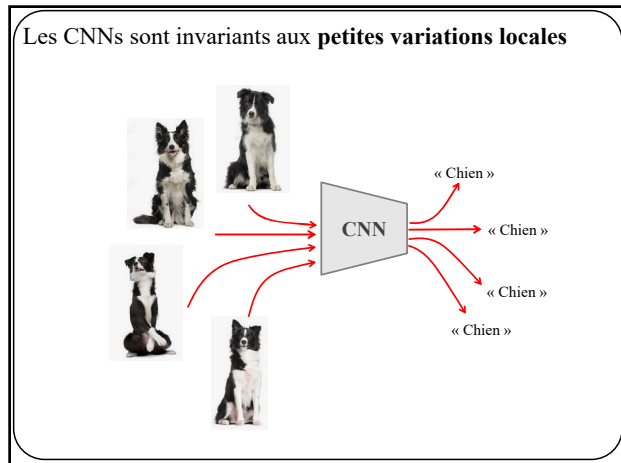
(classification)



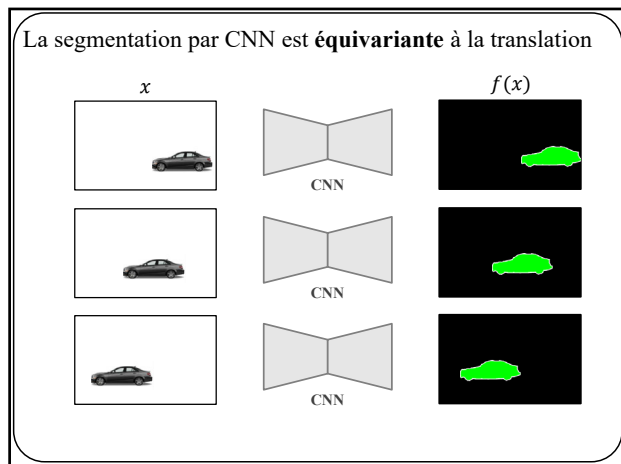
$$f(T(x)) = f(x)$$

où $T(.)$ est l'opérateur de translation

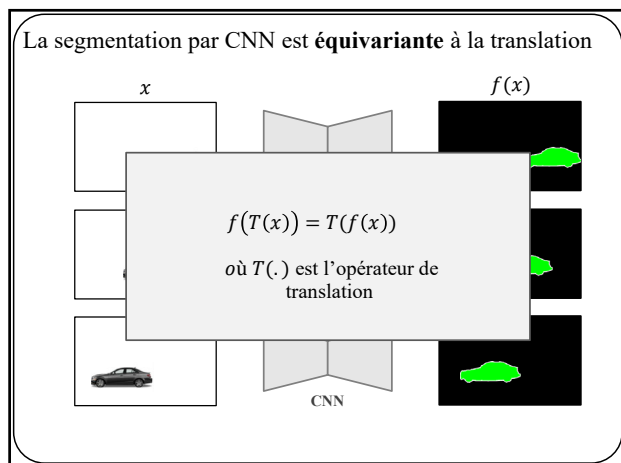
15



16



17



18

Pourquoi les CNNs ont ces propriétés?

19

Convolution

$$(x * h)(i, j) = \sum_n \sum_m x(i - n, j - m) h(n, m)$$

- implique un **produit scalaire** + une **translation**
- les **poids du filtre** sont **fixes** pour l'ensemble des pixels

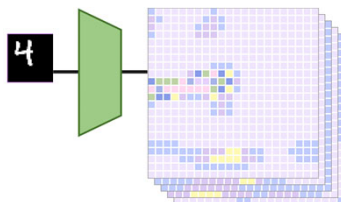
| | | | | |
|----------------|----------------|----------------|---|---|
| 3 ₀ | 3 ₁ | 2 ₂ | 1 | 0 |
| 0 ₂ | 0 ₂ | 1 ₀ | 3 | 1 |
| 3 ₀ | 1 ₁ | 2 ₂ | 2 | 3 |
| 2 | 0 | 0 | 2 | 2 |
| 2 | 0 | 0 | 0 | 1 |

| | | |
|------|------|------|
| 12.0 | 12.0 | 17.0 |
| 10.0 | 17.0 | 19.0 |
| 9.0 | 6.0 | 14.0 |

<https://www.jie-tao.com/types-of-convolutiontranslation/>

20

Convolution



<https://chriswolfvision.medium.com/what-is-translation-equivariance-and-why-do-we-use-convolutions-to-get-it-6f18139d4c59>

21

Convolution

$$(x * h)(i, j) = \sum_n \sum_m x(i - n, j - m) h(n, m)$$

Par conséquent

- La sortie d'une convolution (et CNN) est **prévisible lorsque le signal est traduit**
- Les calculs sont **locaux** (fenêtre NxN)
 - Les petites variations locales ne contaminent pas la sortie
 - C'est l'intérêt d'avoir des petites convolutions (3x3) à travers plusieurs couches d'un réseau profond

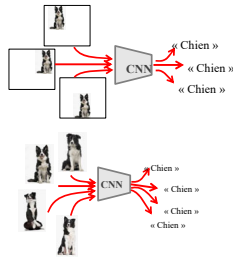
<https://www.jie-tao.com/types-of-convolutiontranslation/>

22

En résumé

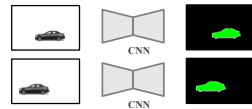
Invariance

$$f(T(x)) = f(x)$$



Équivariance

$$f(T(x)) = T(f(x))$$



23

Le graphe le plus simple:

un ensemble de points

(un graphe sans arête)

24

Très pertinent : conduite autonome, reconstruction 3D, Lidar...



25

Graphes sans arête

Que des nœuds

$$\mathcal{G} = (\mathcal{V}, \cancel{\mathcal{E}})$$

Soit $\vec{x}_i \in R^d$ le vecteur de caractéristiques du nœud i .

Soit $X \in R^{|\mathcal{V}| \times d}$ la matrice de caractéristiques contenant les vecteurs de tous les nœuds i

$$\mathbf{X} = (\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n)^T$$

26

Ensemble de points

Ex.: ensemble de 5 nœuds avec des caractéristiques 2D

Figure 1 shows a 2D scatter plot of five data points, x_a, x_b, x_c, x_d, x_e , each represented by a gray circle and a corresponding vector from the origin. The vectors are labeled with their coordinates in a 2x1 matrix. To the right of the plot, the matrix X is shown, which is a 5x2 matrix where each row is the vector for a data point. The rows are $\begin{bmatrix} 8 & -23 \\ 12 & 25 \\ -4 & 33 \\ 1 & 2 \\ 7 & 33 \end{bmatrix}$.

27

Graphes sans arête

Remarque: en faisant cela, nous avons **imposé un ordre** dans la matrice X : \vec{x}_e est en premier et \vec{x}_d est en dernier

$$\begin{array}{c} \vec{x}_b = \begin{pmatrix} -4 \\ 33 \end{pmatrix} \quad \vec{x}_a = \begin{pmatrix} 12 \\ 25 \end{pmatrix} \\ \vec{x}_c = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad \vec{x}_d = \begin{pmatrix} 7 \\ 33 \end{pmatrix} \end{array} \quad X = \begin{bmatrix} 8 & -23 \\ 12 & 25 \\ -4 & 33 \\ 1 & 2 \\ 7 & 33 \end{bmatrix}$$

28

Graphes sans arête

Remarque: l'**ordre** dans lequel les nœuds sont inscrits **ne change pas la nature** de l'ensemble des nœuds.

$$X = \begin{bmatrix} 8 & -23 \\ 12 & 25 \\ -4 & 33 \\ 1 & 2 \\ 7 & 33 \end{bmatrix} \quad X = \begin{bmatrix} -4 & 33 \\ 12 & 25 \\ 8 & -23 \\ 1 & 2 \\ 7 & 33 \end{bmatrix} \quad X = \begin{bmatrix} -4 & 33 \\ 12 & 25 \\ 8 & -23 \\ 7 & 33 \\ 1 & 2 \end{bmatrix}$$

Toujours le même ensemble de nœuds

29

Ce qu'on souhaite

(pour la classification)

$$f\left(\begin{array}{c} \text{red} \\ \text{blue} \\ \text{yellow} \\ \text{green} \end{array}\right) = f\left(\begin{array}{c} \text{red} \\ \text{yellow} \\ \text{blue} \\ \text{green} \end{array}\right)$$

30

Ce qu'on souhaite
(pour la classification)

$$f\left(\begin{bmatrix} 8 & -23 \\ 12 & 25 \\ -4 & 33 \\ 1 & 2 \\ 7 & 33 \end{bmatrix}\right) = f\left(P\left(\begin{bmatrix} 8 & -23 \\ 12 & 25 \\ -4 & 33 \\ 1 & 2 \\ 7 & 33 \end{bmatrix}\right)\right)$$

On veut que le réseau de neurones soit
invariant à la permutation des lignes de X

31

Permutation matricielle

Une matrice de permutation contient des 0 et des 1 permettant de permuter des lignes.

$$P_{21345}X = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 & -23 \\ 12 & 25 \\ -4 & 33 \\ 1 & 2 \\ 7 & 33 \end{bmatrix} = \begin{bmatrix} 12 & 25 \\ 8 & -23 \\ -4 & 33 \\ 1 & 2 \\ 7 & 33 \end{bmatrix}$$

$$P_{32514}X = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 8 & -23 \\ 12 & 25 \\ -4 & 33 \\ 1 & 2 \\ 7 & 33 \end{bmatrix} = \begin{bmatrix} -4 & 33 \\ 12 & 25 \\ 7 & 33 \\ 8 & -23 \\ 1 & 2 \end{bmatrix}$$

32

Réseau invariant à la permutation

Un réseau de neurones est **invariant à la permutation** si pour toute matrice de permutation P

$$f(PX) = f(X)$$

33

Réseau invariant à la permutation

Un réseau de neurones est **invariant à la permutation** si pour toute matrice de permutation \mathbb{P}

Matrice de permutation

Matrice de caractéristiques

$$f(\mathbb{P}X) = f(X)$$

Réseau de neurones

34

Deep Sets

Un réseau très générique et invariant à la permutation est celui proposé par [Zaheer et al 2017]

$$f(X) = \phi(\sum_{i \in \mathcal{V}} \psi(\vec{x}_i))$$

$\phi(\cdot)$ et $\psi(\cdot)$ sont des réseaux de neurones, e.g. Perceptron

$$\begin{aligned} \rightarrow \phi(\vec{v}) &= \sigma(\mathbf{W}_\phi \vec{v} + \vec{b}_\phi) \\ \rightarrow \psi(\vec{v}) &= \sigma(\mathbf{W}_\psi \vec{v} + \vec{b}_\psi) \end{aligned}$$

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, A.J. Smola *Deep Sets*, NeurIPS 2017.

35

Deep Sets

Un réseau très générique et invariant à la permutation est celui proposé par [Zaheer et al 2017]

$$f(X) = \phi(\sum_{i \in \mathcal{V}} \psi(\vec{x}_i))$$

$\phi(\cdot)$ et $\psi(\cdot)$ sont des réseaux de neurones, e.g. MLP 2 couches (ou plus)

$$\begin{aligned} \rightarrow \phi(\vec{v}) &= \sigma(\mathbf{W}_\phi^2 \sigma(\mathbf{W}_\phi^1 \vec{v} + \vec{b}_\phi^1) + \vec{b}_\phi^2) \\ \rightarrow \psi(\vec{v}) &= \sigma(\mathbf{W}_\psi^2 \sigma(\mathbf{W}_\psi^1 \vec{v} + \vec{b}_\psi^1) + \vec{b}_\psi^2) \end{aligned}$$

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, A.J. Smola *Deep Sets*, NeurIPS 2017.

36

Deep Sets

Ex: MLP à 3 couches + 1 somme

$$f(X) = \phi(\sum_{i \in \mathcal{V}} \psi_2(\psi_1(\vec{x}_i)))$$



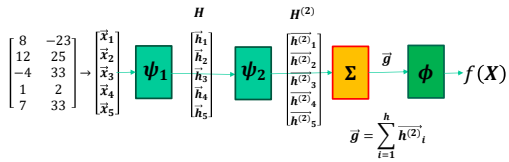
$$\begin{aligned}\psi_1(\vec{x}_i) &= W_1 \vec{x}_i + \vec{b}_1 \\ \psi_2(\vec{h}_i) &= W_2 \vec{h}_i + \vec{b}_2 \\ \phi(\vec{g}) &= W_3 \vec{g} + \vec{b}_3\end{aligned}$$

37

Deep Sets

Ex: MLP à 3 couches + 1 somme

$$\begin{aligned}\psi_1(\vec{x}_i) &= W_1 \vec{x}_i + \vec{b}_1 \\ \psi_2(\vec{h}_i) &= W_2 \vec{h}_i + \vec{b}_2 \\ \phi(\vec{g}) &= W_3 \vec{g} + \vec{b}_3\end{aligned}$$



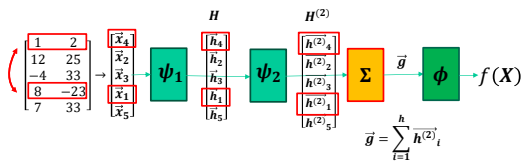
38

Deep Sets, illustration (réseau à 4 couches)

$$\begin{aligned}\psi_1(\vec{x}_i) &= W_1 \vec{x}_i + \vec{b}_1 \\ \psi_2(\vec{h}_i) &= W_2 \vec{h}_i + \vec{b}_2 \\ \phi(\vec{g}) &= W_3 \vec{g} + \vec{b}_3\end{aligned}$$



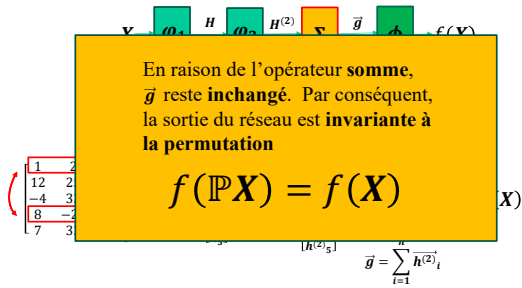
Permutation de 2 entrées



39

Deep Sets, illustration (réseau à 4 couches)

$$f(X) = \phi(\sum_{i \in \mathcal{V}} \varphi_2(\varphi_1(\vec{x}_i)))$$



40

Deep Sets

$$f(X) = \phi(\sum_{i \in \mathcal{V}} \psi(\vec{x}_i))$$

Comme la somme, d'autres opérateurs sont invariants aux permutations (ex. **min**, **max**, **moyenne**, etc.)

41

Deep Sets

$$f(X) = \phi(\oplus_{i \in \mathcal{V}} \psi(\vec{x}_i))$$

Comme la somme, d'autres opérateurs sont invariants aux permutations (ex. **min**, **max**, **moyenne**, etc.).

Il est d'usage de représenter l'opérateur d'agrégation par $\oplus_{i \in \mathcal{V}}$

42

Graphes sans arête : classification des nœuds (segmentation)

On cherche un réseau de neurones **équivariant à la permutation**

$$f(\mathbb{P}X) = \mathbb{P} f(X)$$

Matrice de caractéristiques

Matrice de permutation

Réseau de neurones

43

Graphes sans arête : classification des nœuds (segmentation)

$$f(X) = \phi(\oplus_{i \in \mathcal{V}} \psi(\vec{x}_i))$$

On a qu'à **retirer l'opérateur d'agrégation**

$$f(X) = \begin{pmatrix} \psi(\vec{x}_1) \\ \psi(\vec{x}_2) \\ \dots \\ \psi(\vec{x}_N) \end{pmatrix}$$

et c'est le mieux qu'on puisse faire avec des graphes sans arête.

44

Deep Sets, illustration (MLP 3 couches)

$$f(X) = \phi(\psi_2(\psi_1(\vec{x}_i)))$$

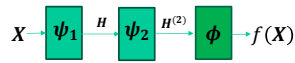
$$X \rightarrow \psi_1 \xrightarrow{H} \psi_2 \xrightarrow{H^{(2)}} \phi \rightarrow f(X)$$

$$\begin{bmatrix} 8 & -23 \\ 12 & 25 \\ -4 & 33 \\ 1 & 2 \\ 7 & 33 \end{bmatrix} \rightarrow \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vec{x}_3 \\ \vec{x}_4 \\ \vec{x}_5 \end{bmatrix} \xrightarrow{H} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \end{bmatrix} \xrightarrow{H^{(2)}} \begin{bmatrix} h^{(2)}_1 \\ h^{(2)}_2 \\ h^{(2)}_3 \\ h^{(2)}_4 \\ h^{(2)}_5 \end{bmatrix} \xrightarrow{\phi} \begin{bmatrix} f(\vec{x}_1) \\ f(\vec{x}_2) \\ f(\vec{x}_3) \\ f(\vec{x}_4) \\ f(\vec{x}_5) \end{bmatrix} = f(X)$$

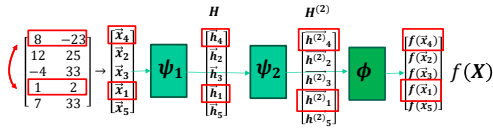
45

Deep Sets, illustration (MLP 3 couches)

$$f(X) = \phi(\psi_2(\psi_1(\vec{x}_i)))$$



Permutation de 2 entrées



46

En résumé

l'apprentissage sur des ensembles de points se fait en appliquant une **fonction équivariante** ψ sur chaque nœud indépendamment

$$\vec{h}_i = \psi(\vec{x}_i)$$

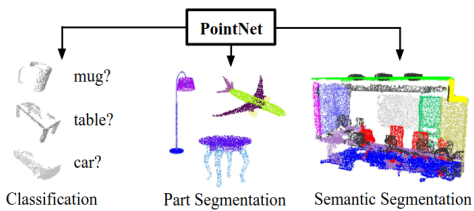
auquel on ajoute un **agrégateur invariant** si nécessaire

$$f(X) = \phi(\oplus_{i \in V} \psi(\vec{x}_i))$$

47

PointNet [Qi et al 2017]

Une parfaite illustration



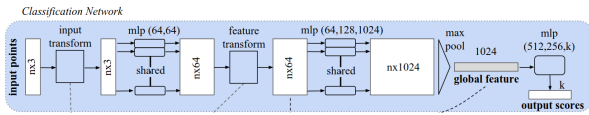
C. Qi, H. Su, K. Mo, L. Guibas *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*, CVPR 2017

48

PointNet [Qi et al 2017]

Une parfaite illustration

Réseau de classification **invariant** à la permutation



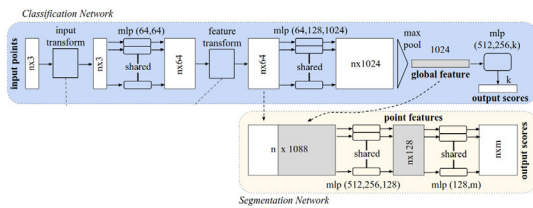
C. Qi, H. Su, K. Mo, L. Guibas *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*, CVPR 2017

49

PointNet [Qi et al 2017]

Une parfaite illustration

Réseau de segmentation **équivalent** à la permutation



C. Qi, H. Su, K. Mo, L. Guibas *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*, CVPR 2017

50

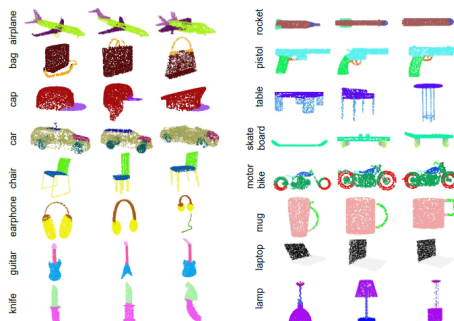


Figure 21. PointNet segmentation results on complete CAD models.

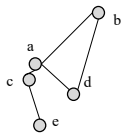
51

Apprentissage sur des graphes

52

Apprentissage sur des graphes

$G = (\mathcal{V}, \mathcal{E})$



Matrice d'adjacence

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix}$$

On souhaite toujours avoir un réseau **invariant/équivariant**

53

Ce qu'on souhaite

Ensemble de points

$$f\left(\begin{matrix} \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \end{matrix}\right) = f\left(\begin{matrix} \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \end{matrix}\right)$$

Graphe

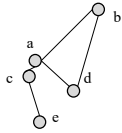
$$f\left(\begin{matrix} \bullet & \bullet \\ & \bullet \\ \bullet & \bullet \end{matrix}\right) = f\left(\begin{matrix} \bullet & \bullet \\ & \bullet \\ \bullet & \bullet \end{matrix}\right)$$

54

Apprentissage sur des graphes

Matrice de caractéristiques

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$



$$X = \begin{bmatrix} \vec{x}_a \\ \vec{x}_b \\ \vec{x}_c \\ \vec{x}_d \\ \vec{x}_e \end{bmatrix}$$

Matrice d'adjacence

$$A = \begin{pmatrix} & a & b & c & d & e \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix}$$

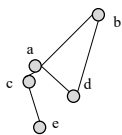
On souhaite toujours avoir un réseau **invariant/équivariant**

55

Permutation des nœuds a et c

Matrice de caractéristiques

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$



$$X = \begin{bmatrix} \vec{x}_c \\ \vec{x}_b \\ \vec{x}_a \\ \vec{x}_d \\ \vec{x}_e \end{bmatrix}$$

Matrice d'adjacence

$$A = \begin{pmatrix} \boxed{c} & b & \boxed{a} & d & e \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} \boxed{c} \\ b \\ \boxed{a} \\ d \\ e \end{matrix}$$

On souhaite toujours avoir un réseau **invariant/équivariant**

56

Permutation

La permutation de 2 (ou plus) éléments dans un graphe implique
la **permutation de la matrice X**

$$\mathbb{P}X$$

la double permutation de la matrice d'adjacence

$$\mathbb{P}A\mathbb{P}^T$$

57

$$X = \begin{bmatrix} \vec{x}_c \\ \vec{x}_b \\ \vec{x}_a \\ \vec{x}_d \\ \vec{x}_e \end{bmatrix}$$

Matrice de permutation

Matrice A d'origine

$$\mathbb{P}_{32145} A \mathbb{P}_{32145}^T = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrice A permutée

58

Permutation sur les graphes

Invariance (pour la classification du graphe)

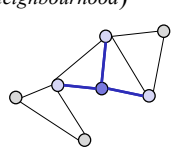
$$f(\mathbb{P}X, \mathbb{P}A\mathbb{P}^T) = f(X, A)$$

Équivariance (pour la classification des nœuds)

$$f(\mathbb{P}X, \mathbb{P}A\mathbb{P}^T) = \mathbb{P}f(X, A)$$

59

Voisinage (*neighbourhood*)



Le voisinage d'un nœud i dans un graphe est généralement constitué de nœuds avec lesquels il partage une arête

$$\mathcal{N}_i = \{j : (i, j) \in \mathcal{E} \vee (j, i) \in \mathcal{E}\}$$

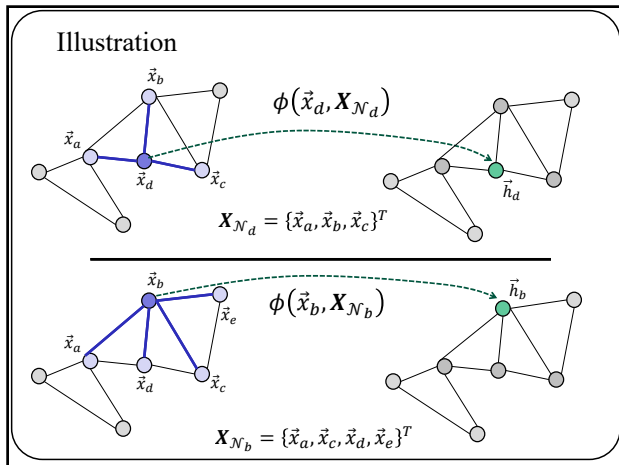
On peut mettre les **vecteurs de caractéristiques du voisinage** dans une matrice

$$X_{\mathcal{N}_i} = \{\vec{x}_j : j \in \mathcal{N}_i\}$$

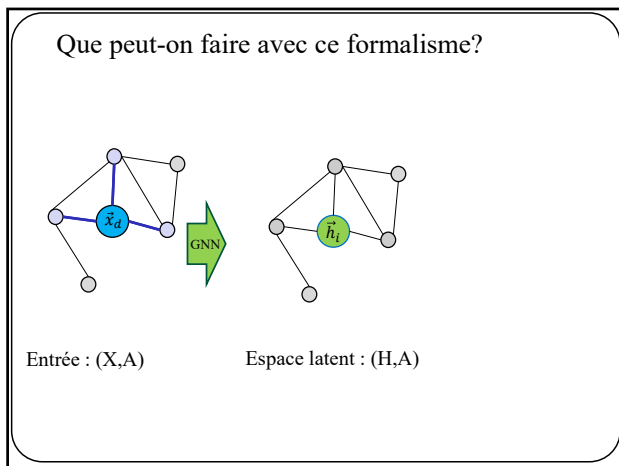
Et définir une fonction sur le voisinage

$$\phi(\vec{x}_i, X_{\mathcal{N}_i})$$

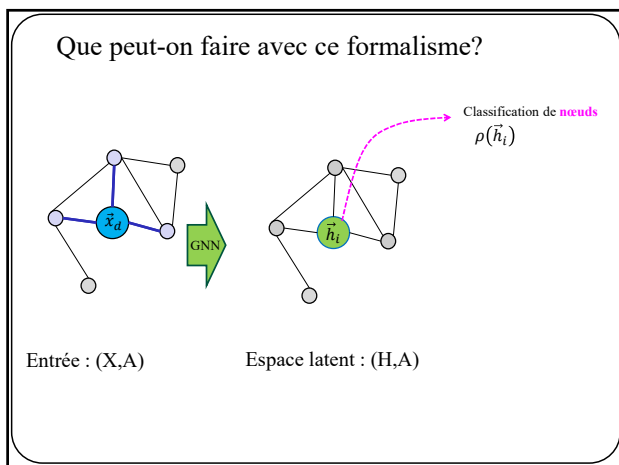
60



61

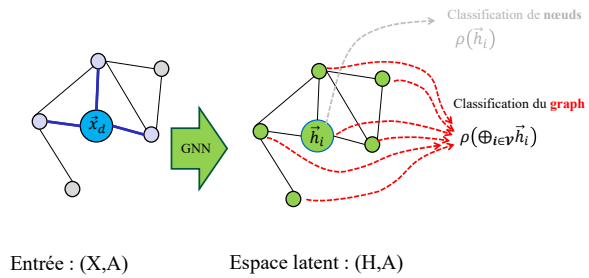


62



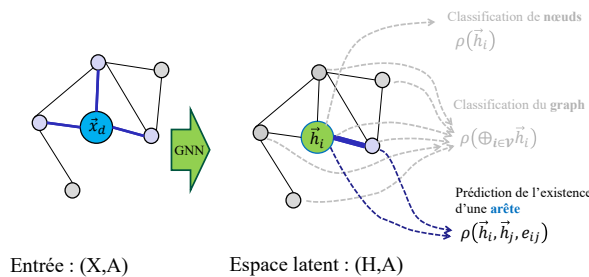
63

Que peut-on faire avec ce formalisme?



64

Que peut-on faire avec ce formalisme?



Dans tous les cas : $\rho(\cdot)$ est un **réseau de neurones**

65

Quel est le contenu de

$$\phi(\vec{x}_i, \mathbf{X}_{\mathcal{N}_i})$$

?

66

Rappel

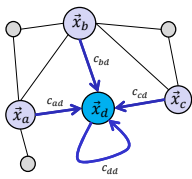
$\phi(\cdot), \psi(\cdot)$ et $\rho(\cdot)$ sont des réseaux de neurones, e.g.

$$\begin{aligned} \rightarrow \phi(\vec{v}) &= \sigma(\mathbf{W}_\phi \vec{v} + \vec{b}_\phi) & \phi(\vec{u}, \vec{v}) &= \sigma(\mathbf{W}_\phi \vec{u} + \mathbf{V}_\phi \vec{v} + \vec{b}_\phi) \\ \rightarrow \psi(\vec{v}) &= \sigma(\mathbf{W}_\psi \vec{v} + \vec{b}_\psi) & \psi(\vec{u}, \vec{v}) &= \sigma(\mathbf{W}_\psi \vec{u} + \mathbf{V}_\psi \vec{v} + \vec{b}_\psi) \\ \rightarrow \rho(\vec{v}) &= \sigma(\mathbf{W}_\rho \vec{v} + \vec{b}_\rho) & \rho(\vec{u}, \vec{v}) &= \sigma(\mathbf{W}_\rho \vec{u} + \mathbf{V}_\rho \vec{v} + \vec{b}_\rho) \end{aligned}$$

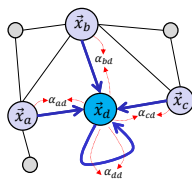
67

Trois familles de méthodes $\phi(\vec{x}_i, \mathbf{X}_{\mathcal{N}_i})$

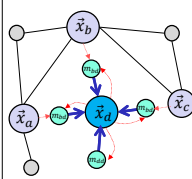
Convolution



Attention



Passage de messages

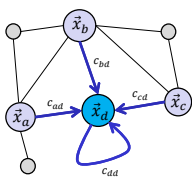


$$\mathbf{X}_{\mathcal{N}_d} = \{\vec{x}_a, \vec{x}_b, \vec{x}_c, \vec{x}_d\}$$

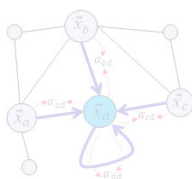
68

Commençons avec l'opération qui nous est la plus familière : **la convolution**

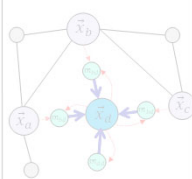
Convolution



Attention



Passage de messages



69

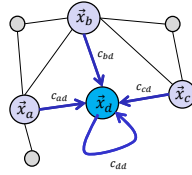
GNN à convolution

$$\vec{h}_d = \phi \left(\vec{x}_d, \bigoplus_{i \in \mathcal{N}_d} c_{id} \psi(\vec{x}_i) \right)$$

avec $c_{id} \in \mathbb{R}$

En général,

- c_{id} est une **constante** issue de A **non apprise par le réseau**
- $\bigoplus_{i \in \mathcal{N}_d}$ est une **sommation/moyenne**



M. Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*, NeurIPS 2016
F. Wu et al. *Simplifying Graph Convolutional Networks*, ICLR 2019

70

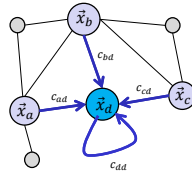
GNN à convolution

$$\vec{h}_d = \phi \left(\vec{x}_d, \bigoplus_{i \in \mathcal{N}_d} c_{id} \psi(\vec{x}_i) \right)$$

avec $c_{id} \in \mathbb{R}$

Exemple d'une solution viable

$$\vec{h}_d = \phi \left(c_{ad} \psi(\vec{x}_a) + c_{bd} \psi(\vec{x}_b) + c_{cd} \psi(\vec{x}_c) + c_{dd} \psi(\vec{x}_d) \right)$$



M. Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*, NeurIPS 2016
F. Wu et al. *Simplifying Graph Convolutional Networks*, ICLR 2019

71

GNN à convolution

Plusieurs **variantes** possibles...

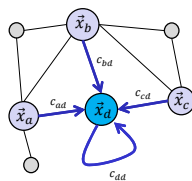
$$\vec{h}_d = \phi \left(\underbrace{\sum_{i \in \mathcal{N}_d} c_{id} \vec{x}_i}_{\text{Somme pondérée des voisins}} \right)$$

et si $c_{id} = A_{id}$ (contenu de la matrice d'adjacence)

$$\vec{h}_d = \phi(A_d X)$$

Pour le graph au complet

$$H = \phi(AX)$$



Thomas N. Kipf and Max Welling *Semi-Supervised Classification with Graph Convolutional Networks*, 2016

72

GNN à convolution

Plusieurs variantes possibles...

\vec{h}_d

Très utile pour les **graphes homophiles**
(le poids des arêtes encodent la similarité)

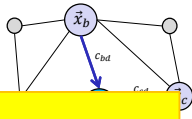
Et si

Très utilisé industriellement

Pour le graph au complet

$$H = \phi(AX)$$

M. Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering*, NeurIPS 2016
F. Wu et al. *Simplifying Graph Convolutional Networks*, ICLR 2019



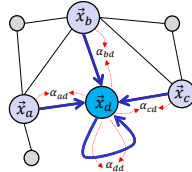
73

GNN d'attention

$$\vec{h}_d = \phi(\vec{x}_d, \bigoplus_{i \in N_d} \alpha(\vec{x}_i, \vec{x}_d) \psi(\vec{x}_i))$$

avec $\alpha(\vec{x}_i, \vec{x}_d) \in \mathbb{R}$

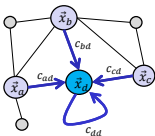
$\alpha(\vec{x}_i, \vec{x}_d)$ est une fonction qui détermine
l'intensité du lien entre les nœuds i et d .



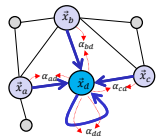
M. Velićković et al. *Graph Attention Network*, ICLR 2018
S. Brody et al. *How Attentive are Graph Attention Networks?* ICLR 2022

74

GNN convolutif vs GNN d'attention



$$\vec{h}_d = \phi(\vec{x}_d, \bigoplus_{i \in N_d} c_{di} \psi(\vec{x}_i))$$



$$\vec{h}_d = \phi(\vec{x}_d, \bigoplus_{i \in N_d} \alpha(\vec{x}_i, \vec{x}_d) \psi(\vec{x}_i))$$

Dans les 2 cas, on a une **somme pondérée** des vecteurs transformés par un réseau ψ

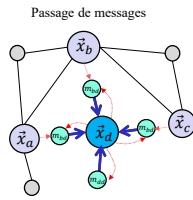
Dans le cas d'un GNN multicouches

- Les coefficients c_{di} **sont fixes** et les mêmes pour toutes les couches
- Les coefficients $\alpha(\vec{x}_i, \vec{x}_d)$ **s'adaptent** au contenu de chaque couche.

76

GNN avec passage de messages

$$\vec{h}_d = \phi \left(\vec{x}_d, \bigoplus_{i \in \mathcal{N}_d} \psi(\vec{x}_i, \vec{x}_d) \right)$$



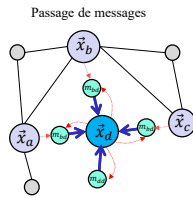
Les arêtes apprennent l'information à combiner à chaque nœud

P. Battaglia et al. *Relational inductive biases, deep learning, and graph networks*, arXiv:1806.01261 2018

77

GNN avec passage de messages

$$\vec{h}_d = \phi \left(\vec{x}_d, \bigoplus_{i \in \mathcal{N}_d} \psi(\vec{x}_i, \vec{x}_d) \right)$$

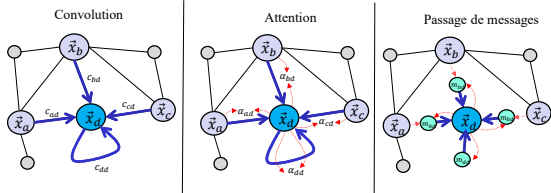


Différences

- **GNN conv** : Les coefficients c_{di} **sont fixes** et sont des **scalaires**
- **GNN attention** : Les coefficients $\alpha(\vec{x}_i, \vec{x}_d)$ **s'adaptent** et sont des **scalaires**
- **GNN PM** : Les éléments $\psi(\vec{x}_i, \vec{x}_d)$ **s'adaptent** et sont des **vecteurs**

P. Battaglia et al. *Relational inductive biases, deep learning, and graph networks*, arXiv:1806.01261 2018

78



Les 3 approches implémentent une **combinaison de MLPs**

79

Convolution

Attention

Passage de messages

Les 3 approches respectent les principes

- d'invariance aux permutations $\rightarrow \oplus_{i \in \mathcal{N}_d}$ opérateur d'agrégation
- d'invariance aux perturbations locales mineures $\rightarrow \mathcal{N}_d$ voisinage
- d'équivariance $\rightarrow \psi(\vec{x}_j, \vec{x}_d)$ est le même pour tous les nœuds

80

Convolution

Attention

Passage de messages

- GNN d'attention : cas particulier de GNN Passage de messages
- GNN convolutif : cas particulier de GNN d'attention
- GNN sans arête : cas particulier de GNN de convolutif
- CNN sur une grille régulière : cas particulier de GNN convolutif

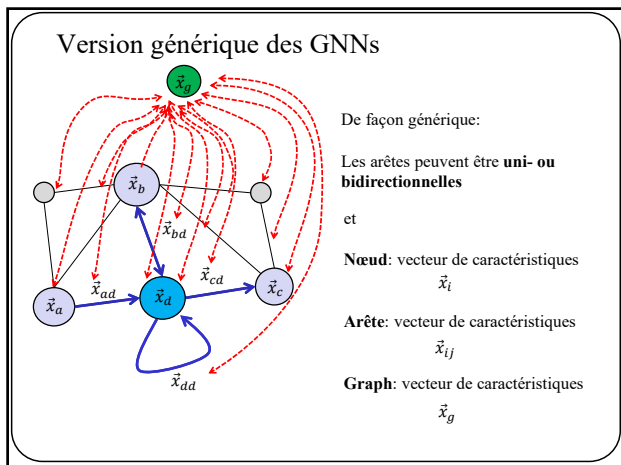
81

Version la plus générique des GNNs
à ce jour

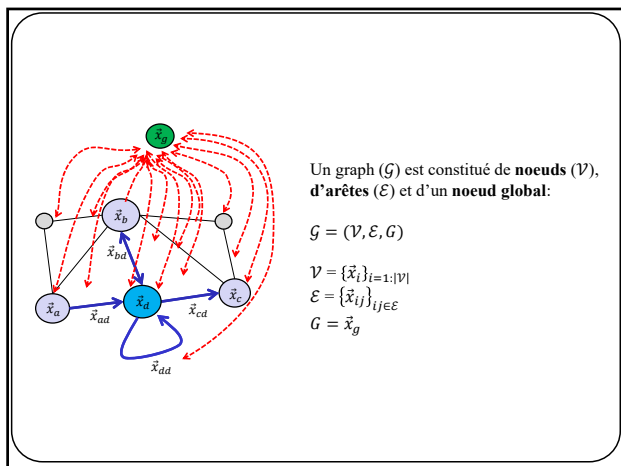
Graph Nets*

* P.W Battaglia et al. Relational inductive biases, deep learning, and graph networks, arXiv:1806.01261 2018
github.com/deepmind/graph_nets

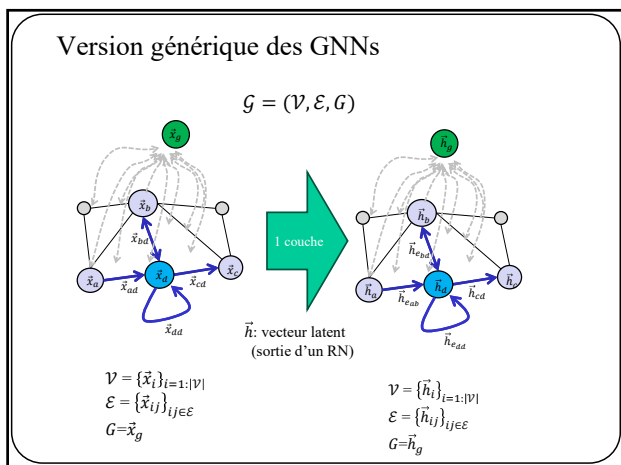
82



83



84

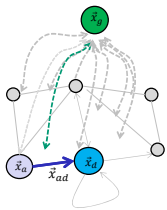


85

Version générique des GNNs

Mise à jour des arêtes

$$\vec{h}_{ij} = \psi(\vec{x}_i, \vec{x}_j, \vec{x}_{ij}, \vec{x}_g), \quad \forall ij \in \mathcal{E}$$



86

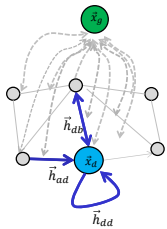
Version générique des GNNs

Mise à jour des arêtes

$$\vec{h}_{ij} = \psi(\vec{x}_i, \vec{x}_j, \vec{x}_{ij}, \vec{x}_g), \quad \forall ij \in \mathcal{E}$$

Agrégation des arêtes de chaque nœud

$$\vec{h}_{N_k} = \oplus_{ij \in N_k} \vec{h}_{ij} \quad \forall k \in \mathcal{V}$$



87

Version générique des GNNs

Mise à jour des arêtes

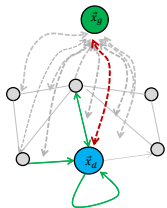
$$\vec{h}_{ij} = \psi(\vec{x}_i, \vec{x}_j, \vec{x}_{ij}, \vec{x}_g), \quad \forall ij \in \mathcal{E}$$

Agrégation des arêtes de chaque nœud

$$\vec{h}_{N_k} = \oplus_{ij \in N_k} \vec{h}_{ij} \quad \forall k \in \mathcal{V}$$

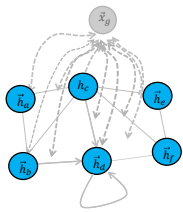
Mise à jour des nœuds

$$\vec{h}_k = \phi(\vec{x}_k, \vec{h}_{N_k}, \vec{x}_g) \quad \forall k \in \mathcal{V}$$



88

Version générique des GNNs



Mise à jour des **arêtes**

$$\vec{h}_{ij} = \psi(\vec{x}_i, \vec{x}_j, \vec{x}_{ij}, \vec{x}_g), \quad \forall ij \in \mathcal{E}$$

Agrégation des arêtes de chaque **nœud**

$$\vec{h}_{N_k} = \bigoplus_{ij \in N_k} \vec{h}_{ij} \quad \forall k \in \mathcal{V}$$

Mise à jour des **nœuds**

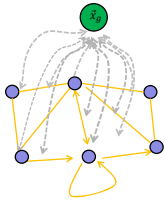
$$\vec{h}_k = \phi(\vec{x}_k, \vec{h}_{N_k}, \vec{x}_g) \quad \forall k \in \mathcal{V}$$

Agrégation des **nœuds**

$$\vec{h}_n = \bigoplus_{k \in \mathcal{V}} \vec{h}_k$$

89

Version générique des GNNs



Mise à jour des **arêtes**

$$\vec{h}_{ij} = \psi(\vec{x}_i, \vec{x}_j, \vec{x}_{ij}, \vec{x}_g), \quad \forall ij \in \mathcal{E}$$

Agrégation des **arêtes** de chaque nœud

$$\vec{h}_{N_k} = \bigoplus_{ij \in N_k} \vec{h}_{ij} \quad \forall k \in \mathcal{V}$$

Mise à jour des **nœuds**

$$\vec{h}_k = \phi(\vec{x}_k, \vec{h}_{N_k}, \vec{x}_g) \quad \forall k \in \mathcal{V}$$

Agrégation des **nœuds**

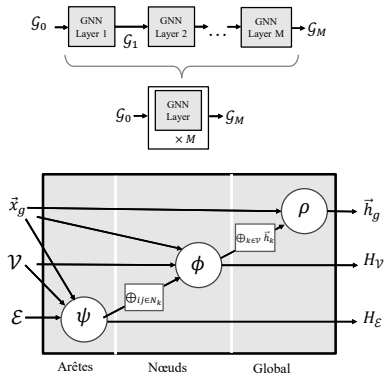
$$\vec{h}_n = \bigoplus_{k \in \mathcal{V}} \vec{h}_k$$

Mise à jour du **graph**

$$\vec{h}_g = \rho(\vec{h}_n, \vec{h}_{N_k}, \vec{x}_g)$$

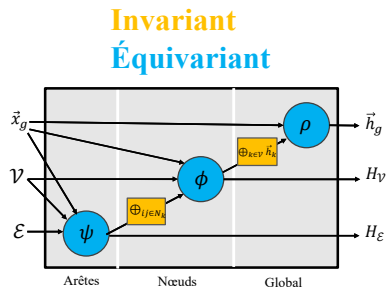
90

Autre représentation d'une couche GNN



91

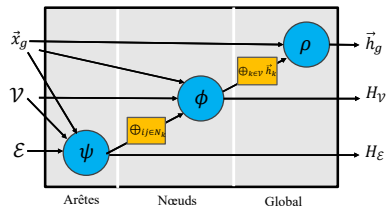
Autre représentation d'une couche GNN



92

Autre représentation d'une couche GNN

Que des MLP!
 ψ, ϕ, ρ



93

En conclusion...

94

Principe d'invariance
Principe d'équivariance

3 familles

- GNN convolutif
- GNN d'attention
- GNN passage de messages

Graph Nets = le GNN le plus générique qui soit

GNN = des **MLP** bien agencés!
