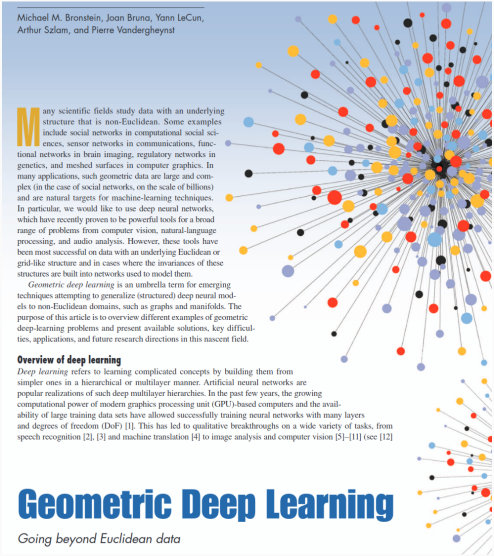
Exploiting Graph Invariants in Deep Learning

Marc Lelarge
INRIA, DI/ENS, PSL Research University, PRAIRIE

PRAIRIE 2022 - Janv

Geometric deep learning



Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst

Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. In many applications, such geometric data are large and complex (in the case of social networks, on the scale of billions) and are natural targets for machine-learning techniques. In particular, we would like to use deep neural networks, which have recently proven to be powerful tools for a broad range of problems from computer vision, natural-language processing, and audio analysis. However, these tools have been most successful on data with an underlying Euclidean or grid-like structure and in cases where the invariances of these structures are built into networks used to model them.

Geometric deep learning is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available solutions, key difficulties, applications, and future research directions in this nascent field.

Overview of deep learning

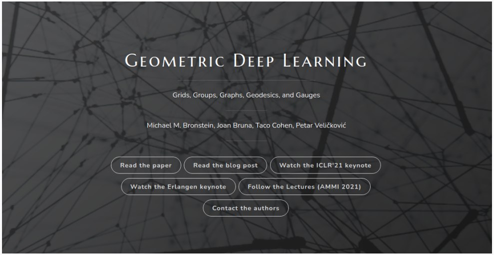
Deep learning refers to learning complicated concepts by building them from simpler ones in a hierarchical or multilayer manner. Artificial neural networks are popular realizations of such deep multilayer hierarchies. In the past few years, the growing computational power of modern graphics processing unit (GPU)-based computers and the availability of large training data sets have allowed successfully training neural networks with many layers and degrees of freedom (DoF) [1]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition [2], [3] and machine translation [4] to image analysis and computer vision [5]–[11] (see [12]

Geometric Deep Learning

Going beyond Euclidean data

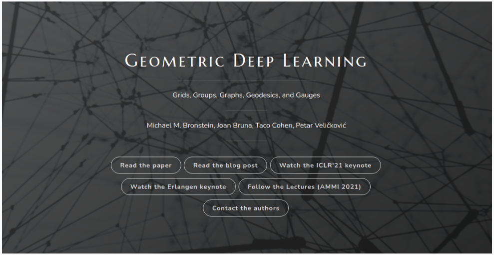
(Bronstein et al., 2017)

Geometric deep learning



`https://geometricdeeplearning.com/`

Geometric deep learning

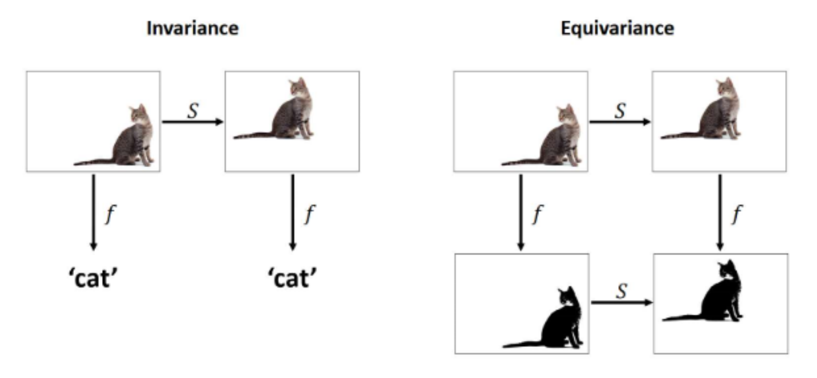


`https://geometricdeeplearning.com/`

In this talk : Symmetries in Machine Learning

What can be done when trying to learn a task that is known to be invariant to some group of symmetries ?

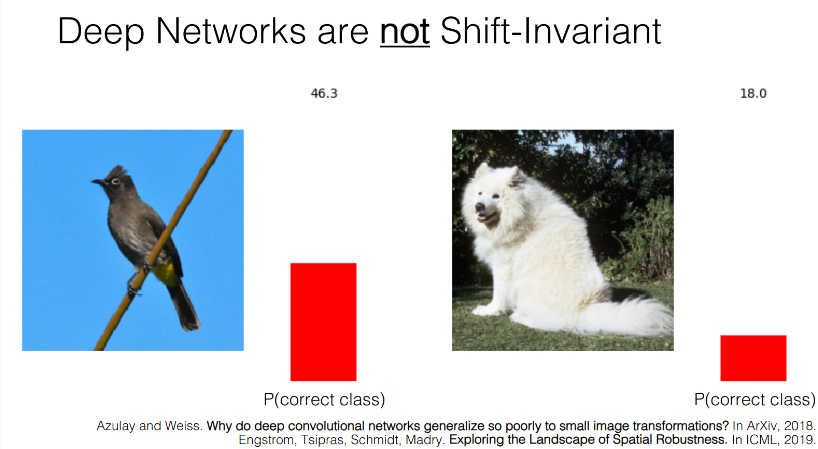
Invariance and Equivariance



source : image from Bernhard Kainz

Why Invariance and Equivariance?

Why Invariance and Equivariance?



taken from Zhang (2019)

How to make your algorithm invariant

Motivation for invariant/equivariant algorithms : by restricting the class of functions we are learning, we lower the complexity of the model and improve its robustness and generalization.

How to make your algorithm invariant

Motivation for invariant/equivariant algorithms : by restricting the class of functions we are learning, we lower the complexity of the model and improve its robustness and generalization.

From an arbitrary function f, an easy way to construct an invariant version :

$$\frac{1}{|T|}\sum_{t\in T} f(T(\mathcal{I}))$$

How to make your algorithm invariant

Motivation for invariant/equivariant algorithms : by restricting the class of functions we are learning, we lower the complexity of the model and improve its robustness and generalization.

From an arbitrary function f, an easy way to construct an invariant version :

$$\frac{1}{|T|}\sum_{t\in T} f(T(\mathcal{I}))$$

so that the learned function f_θ is such that

$$f_\theta(T(\mathcal{I})) \approx f_\theta(\mathcal{I})$$

Convolutions from first principles

How to build an equivariant layer?

Convolutions from first principles

How to build an equivariant layer ?

Th : Shift equivariance + Linear = Convolution

more details at : `https://dataflowr.github.io/website/modules/extras/Convolutions_first/`

Convolutions from first principles

How to build an equivariant layer ?

Th : Shift equivariance + Linear = Convolution

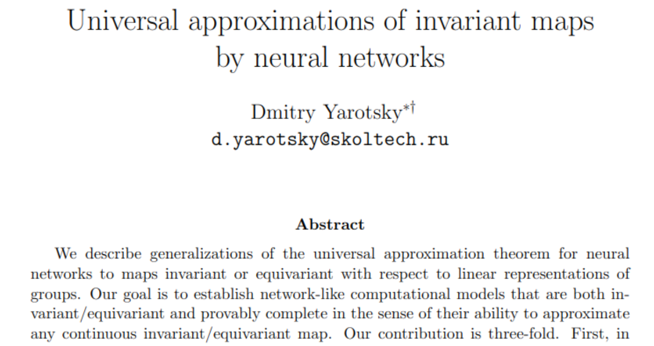
more details at : `https://dataflowr.github.io/website/modules/extras/Convolutions_first/`

Pb with practical CNN : max pooling breaks the equivariance property.

Expressive power of CNNs

To learn a function f that is known to be invariant to some symmetries, we use linear layers that respect this symmetry. Can such a network approximate an arbitrary continuous invariant function?

To learn a function f that is known to be invariant to some symmetries, we use linear layers that respect this symmetry. Can such a network approximate an arbitrary continuous invariant function?


Universal approximations of invariant maps
by neural networks

Dmitry Yarotsky*†
d.yarotsky@skoltech.ru

Abstract

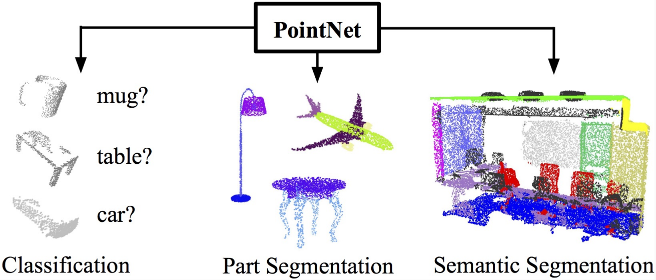
We describe generalizations of the universal approximation theorem for neural networks to maps invariant or equivariant with respect to linear representations of groups. Our goal is to establish network-like computational models that are both invariant/equivariant and provably complete in the sense of their ability to approximate any continuous invariant/equivariant map. Our contribution is three-fold. First, in

(Yarotsky, 2021)

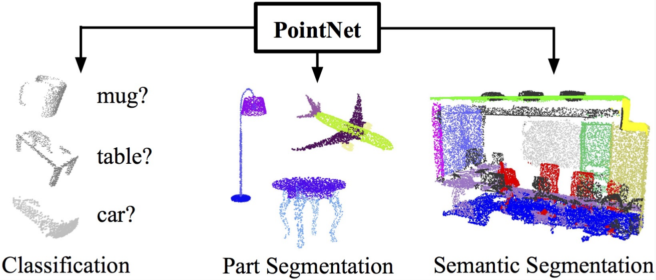
Summary for CNNs

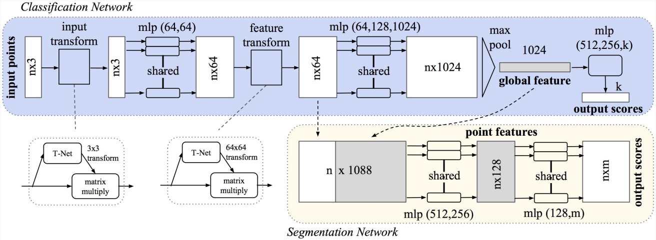
- Symmetries in ML : Invariance and Equivariance
- for CNNs, using equivariant layers does not restrict the Expressive Power of the network
- in practice, architectures are not invariant and we use other techniques like data augmentation...

Learning with point clouds : PointNet



Learning with point clouds : PointNet





(Qi et al., 2017)

PointNet is not equivariant universal

PointNet architecture can be abstracted as :
$PointNet(x_1, \ldots, x_n) \mapsto (f(x_1), \ldots, f(x_n))$

PointNet is not equivariant universal

PointNet architecture can be abstracted as :
$PointNet(x_1, \ldots, x_n) \mapsto (f(x_1), \ldots, f(x_n))$

It is equivariant as
$PointNet(x_{\sigma(1)}, \ldots, x_{\sigma(n)}) = \big(f(x_{\sigma(1)}), \ldots, f(x_{\sigma(n)})\big)$ for any permutation $\sigma \in \mathcal{S}_n$.

PointNet is not equivariant universal

PointNet architecture can be abstracted as :
$PointNet(x_1, \dots, x_n) \mapsto (f(x_1), \dots, f(x_n))$

It is equivariant as
$PointNet(x_{\sigma(1)}, \dots, x_{\sigma(n)}) = \big(f(x_{\sigma(1)}), \dots, f(x_{\sigma(n)})\big)$ for any permutation $\sigma \in \mathcal{S}_n$.

For $n = 2$, whatever f, you cannot approximate the following equivariant function :
$(x_1, x_2) \mapsto \left(\frac{x_1+x_2}{2}, \frac{x_1+x_2}{2}\right)$

PointNet is not equivariant universal

PointNet architecture can be abstracted as :
$\textit{PointNet}(x_1, \ldots, x_n) \mapsto (f(x_1), \ldots, f(x_n))$

It is equivariant as
$\textit{PointNet}(x_{\sigma(1)}, \ldots, x_{\sigma(n)}) = \big(f(x_{\sigma(1)}), \ldots, f(x_{\sigma(n)})\big)$ for any permutation $\sigma \in \mathcal{S}_n$.

For $n = 2$, whatever f, you cannot approximate the following equivariant function :
$(x_1, x_2) \mapsto \left(\frac{x_1+x_2}{2}, \frac{x_1+x_2}{2}\right)$

Indeed this obstruction is the only one for universality and DeepSets and PointNetSeg are equivariant universal :
$(x_1, \ldots, x_n) \mapsto \big(f(x_1, \sum_i \Phi(x_i)), \ldots, f(x_n, \sum_i \Phi(x_i))\big)$

(Zaheer et al., 2017), for more details `https://dataflowr.github.io/website/modules/extras/invariant_equivariant/`

Learning with graph symmetries

Why symmetries matter with graphs ?

Start with a linear regression : your task is to estimate a linear model $\beta_1 x_1 + \cdots + \beta_n x_n$ from noisy observations $(\mathbf{x}, y)$.

Q : How many parameters do you need to estimate if you know in addition that the model is invariant to permutation of the input $(x_1, \ldots, x_n)$?

Why symmetries matter with graphs ?

Start with a linear regression : your task is to estimate a linear model $\beta_1 x_1 + \cdots + \beta_n x_n$ from noisy observations $(\mathbf{x}, y)$.

Q : How many parameters do you need to estimate if you know in addition that the model is invariant to permutation of the input $(x_1, \ldots, x_n)$?

A : there is only one parameter to estimate because invariance implies $\beta_1 = \cdots = \beta_n$.

Why symmetries matter with graphs ?

Start with a linear regression : your task is to estimate a linear model $\beta_1 x_1 + \cdots + \beta_n x_n$ from noisy observations $(\mathbf{x}, y)$.

Q : How many parameters do you need to estimate if you know in addition that the model is invariant to permutation of the input $(x_1, \ldots, x_n)$?

A : there is only one parameter to estimate because invariance implies $\beta_1 = \cdots = \beta_n$.

Q : a linear regression on graphs : estimate a linear function of the adjacency matrix in $\mathbb{R}^{n \times n}$, how many parameters to estimate ?

Why symmetries matter with graphs ?

Start with a linear regression : your task is to estimate a linear model $\beta_1 x_1 + \cdots + \beta_n x_n$ from noisy observations $(\mathbf{x}, y)$.

Q : How many parameters do you need to estimate if you know in addition that the model is invariant to permutation of the input $(x_1, \ldots, x_n)$?

A : there is only one parameter to estimate because invariance implies $\beta_1 = \cdots = \beta_n$.

Q : a linear regression on graphs : estimate a linear function of the adjacency matrix in $\mathbb{R}^{n\times n}$, how many parameters to estimate ?

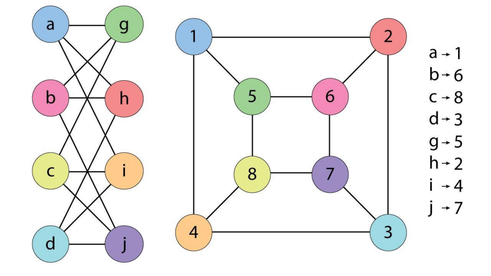
A : there are only two parameters to estimate for a linear function $f : \mathbb{R}^{n\times n} \to \mathbb{R}$ invariant to permutation of the rows and columns :

$$f(A) = \alpha \sum_{i=j} A_{i,j} + \beta \sum_{i\neq j} A_{i,j},$$

whatever the value of n !

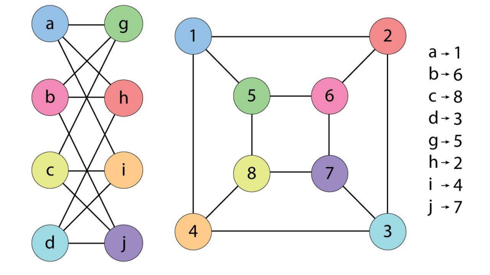
Graph isomorphism

$G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there is a bijection $V_1 \longrightarrow V_2$ which preserves edges.



Graph isomorphism

$G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there is a bijection $V_1 \longrightarrow V_2$ which preserves edges.



Idea : design a machine learning algorithm whose result does not depend on the representation of the input.

Invariant and Equivariant GNNs

Invariant and equivariant functions

For a permutation $\sigma \in \mathcal{S}_n$, we define ($\mathbb{F} = \mathbb{R}^p$ feature space) :

- for $X \in \mathbb{F}^n$, $(\sigma \star X)_{\sigma(i)} = X_i$
- for $G \in \mathbb{F}^{n \times n}$, $(\sigma \star G)_{\sigma(i_1),\sigma(i_2)} = G_{i_1,i_2}$

Invariant and equivariant functions

For a permutation $\sigma \in \mathcal{S}_n$, we define ($\mathbb{F} = \mathbb{R}^p$ feature space) :

- for $X \in \mathbb{F}^n$, $(\sigma \star X)_{\sigma(i)} = X_i$
- for $G \in \mathbb{F}^{n \times n}$, $(\sigma \star G)_{\sigma(i_1),\sigma(i_2)} = G_{i_1,i_2}$

G_1, G_2 are isomorphic iff $G_1 = \sigma \star G_2$.

Invariant and equivariant functions

For a permutation $\sigma \in \mathcal{S}_n$, we define ($\mathbb{F} = \mathbb{R}^p$ feature space) :

- for $X \in \mathbb{F}^n$, $(\sigma \star X)_{\sigma(i)} = X_i$
- for $G \in \mathbb{F}^{n \times n}$, $(\sigma \star G)_{\sigma(i_1), \sigma(i_2)} = G_{i_1, i_2}$

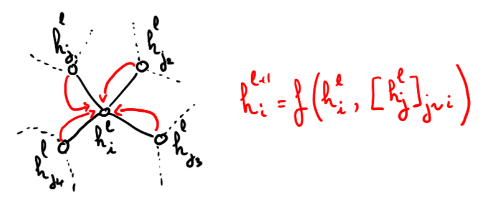
G_1, G_2 are isomorphic iff $G_1 = \sigma \star G_2$.

Definition
($k = 1$ or $k = 2$)

A function $f : \mathbb{F}^{n^k} \to \mathbb{F}$ is said to be invariant if $f(\sigma \star G) = f(G)$.

A function $f : \mathbb{F}^{n^k} \to \mathbb{F}^n$ is said to be equivariant if $f(\sigma \star G) = \sigma \star f(G)$.

Practical GNNs are not universal

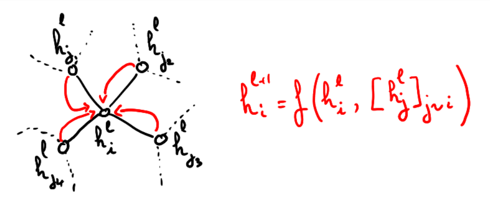


MGNN takes as input a discrete graph $G = (V, E)$ with n nodes and are defined inductively as : $h_i^\ell \in \mathbb{F}$ being the features at layer ℓ associated with node i, then

$$h_i^{\ell+1} = f\left(h_i^\ell, \left\{\left\{h_j^\ell\right\}\right\}_{j\sim i}\right) = f_0\left(h_i^\ell, \sum_{j\sim i} f_1\left(h_j^\ell\right)\right),$$

where f or f_0 and f_1 are learnable functions.

A first example : Message passing GNN (MGNN)



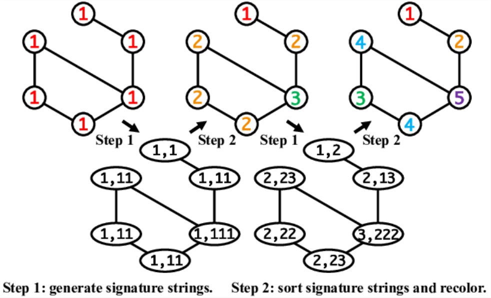
MGNN takes as input a discrete graph $G = (V, E)$ with n nodes and are defined inductively as : $h_i^{\ell} \in \mathbb{F}$ being the features at layer ℓ associated with node i, then

$$h_i^{\ell+1} = f\left(h_i^{\ell}, \left\{\left\{h_j^{\ell}\right\}\right\}_{j\sim i}\right) = f_0\left(h_i^{\ell}, \sum_{j\sim i} f_1\left(h_j^{\ell}\right)\right),$$

where f or f_0 and f_1 are learnable functions.

Prop : The message passing layer is equivariant and both expressions above are equivalent (i.e. for each f, there exists f_0 and f_1).

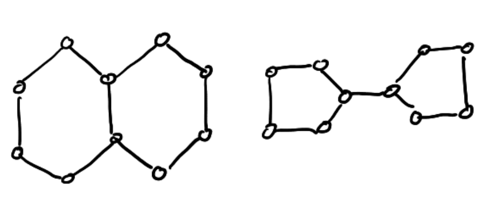
For $k \geq 2$, k-WL(G) are invariants based on the Weisfeiler-Lehman tests designed for the graph isomorphism problem.



MGNN are not universal

Prop : MGNN are useless on d-regular graphs (without features).

Another example of a problematic pair for MGNN :



Learning with (practical i.e. $k = 2$) FGNN

(Maron et al., 2019) adapted the Folklore version of the Weisfeiler-Lehman test to propose the folklore graph layer (FGL) :

$$h_{i\to j}^{\ell+1} = f_0\left(h_{i\to j}^{\ell}, \sum_{k\in V} f_1\left(h_{i\to k}^{\ell}\right) f_2\left(h_{k\to j}^{\ell}\right)\right),$$

where f_0, f_1 and f_2 are learnable functions.

For FGNNs, messages are associated with pairs of vertices as opposed to MGNN where messages are associated with vertices.

FGNN : a FGNN is the composition of FGLs and a final invariant/equivariant reduction layer from $\mathbb{F}^{n^2}$ to $\mathbb{F}/\mathbb{F}^n$.

Separation (Maron et al., 2019) : FGL is equivariant and the same power of separation as 3-WL.

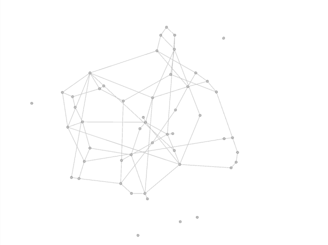
Expressiveness (Azizian and Lelarge, 2020) :
FGNN has the best power of approximation among all architectures working with tensors of order 2 (MGNN or LGNN).

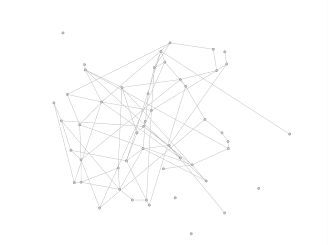
Proof : from separation to approximation via a Stone-Weierstrass theorem

Alignment of Graphs

From graph 1 (on the left), put indices on its vertices, perturb the graph by adding and removing a few edges and remove indices to obtain graph 2 (on the right).

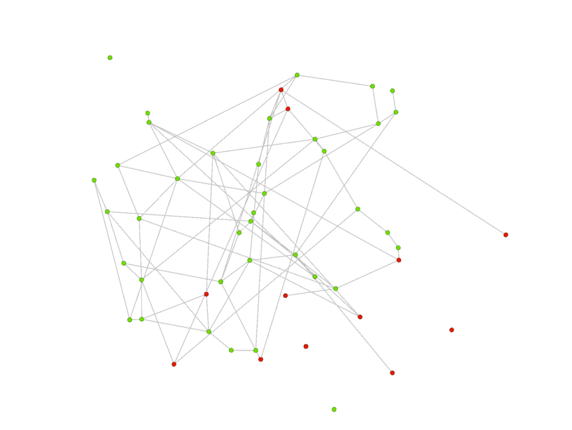
Task : recover the indices on vertices of graph 2.





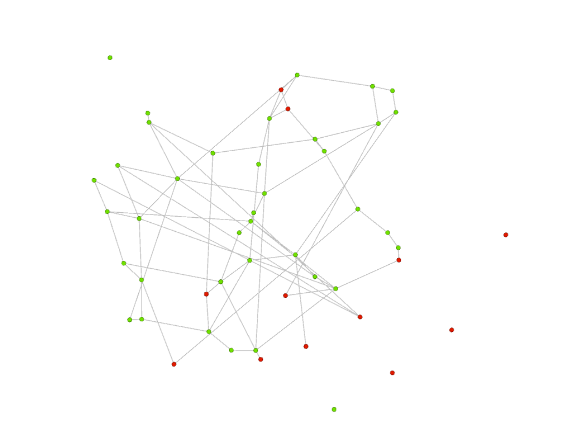
Result with FGNN

Green vertices are good predictions. Red vertices are errors (graph 2).



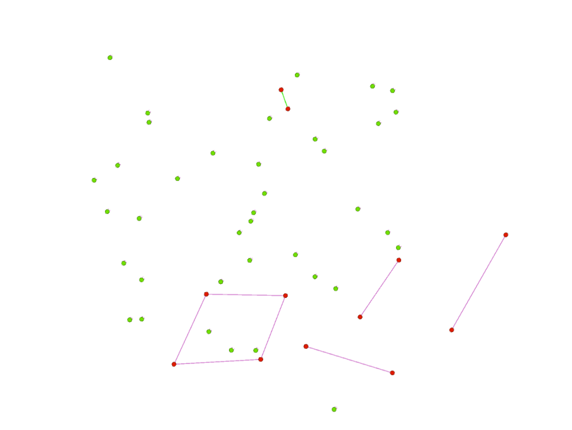
Result with FGNN

Green vertices are good predictions. Red vertices are errors (graph 1).



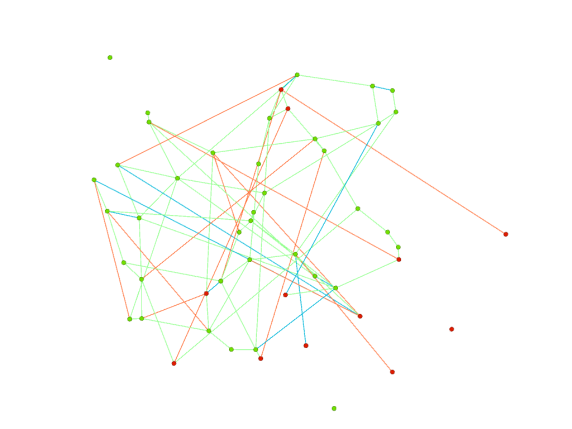
Result with FGNN

Here are the 'wrong' matchings or cycles.



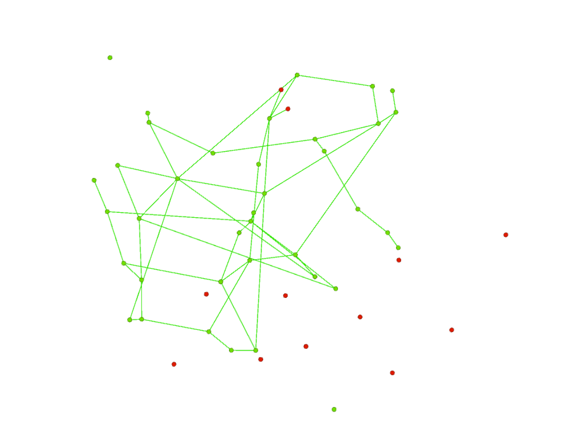
Result with FGNN

Superposing the 2 graphs : green edges in both, orange and blue edges in graph 1 and 2 resp.



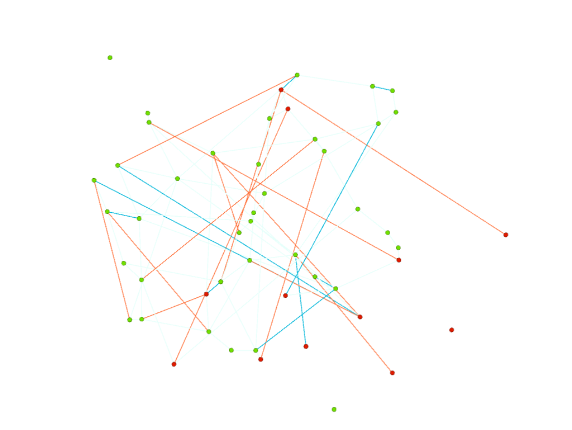
Result with FGNN

Matched edges.



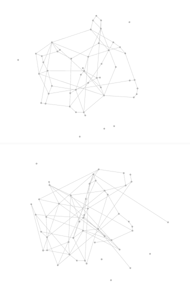
Result with FGNN

Green vertices are well paired vertices. Red vertices are errors.



A learning algorithm

Learning the graph alignment problem with Siamese FGNNs



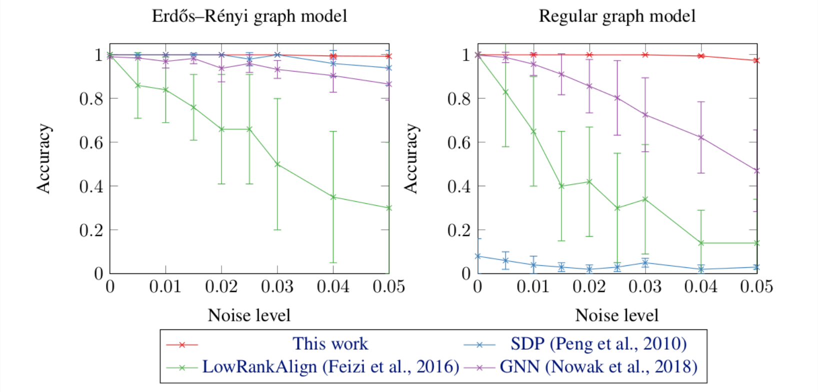
$$G_1 \in \{0,1\}^{n^2} \xrightarrow{\text{FGNN}} E_1 \in \mathbb{R}^{n \times b}$$

$$G_2 \in \{0,1\}^{n^2} \xrightarrow{\text{FGNN}} E_2 \in \mathbb{R}^{n \times b}$$

$$E_1 E_2^T \in \mathbb{R}^{n^2}$$

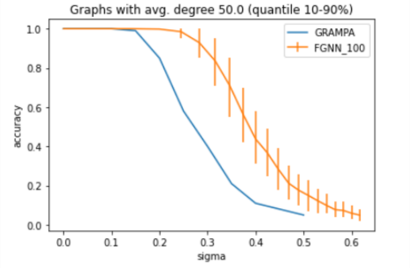
- The same FGNN is used for both graphs.
- From the node similarity matrix $E_1 E_2^T$, we extract a mapping from nodes of G_1 to nodes of G_2 (using a LAP solver to get a permutation).

Results on synthetic data



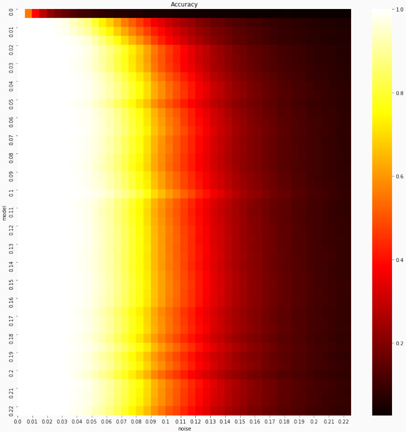
- Graphs : $n = 50$, density = 0.2
- Training set : 20000 samples
- Validation and Test sets : 1000 samples

Comparison FGNN vs GRAMPA



Overlap as a function of $\sigma = \sqrt{\frac{1-s}{1-\lambda/n}}$ for correlated E-R with average degree 50 (number of nodes : 100). Fan et al. (2019)

Generalization for regular graphs



Each line corresponds to a model trained at a given noise level and shows its accuracy across all noise levels.

Conclusion

- we can deal with symmetries in ML thanks to invariance and equivariance
- images : geometric invariants - graphs : combinatorial invariants

Conclusion

- we can deal with symmetries in ML thanks to invariance and equivariance
- images : geometric invariants - graphs : combinatorial invariants
- when the group of invariants is huge, data augmentation is not possible and we need to use architectures that respect these invariants

Conclusion

- we can deal with symmetries in ML thanks to invariance and equivariance
- images : geometric invariants - graphs : combinatorial invariants
- when the group of invariants is huge, data augmentation is not possible and we need to use architectures that respect these invariants
- as a result, we lower the expressive power of our network and practical GNNs are not universal

Conclusion

- we can deal with symmetries in ML thanks to invariance and equivariance
- images : geometric invariants - graphs : combinatorial invariants
- when the group of invariants is huge, data augmentation is not possible and we need to use architectures that respect these invariants
- as a result, we lower the expressive power of our network and practical GNNs are not universal
- other applications of GNNs in graph theory, combinatorial optimization...
- Transformers are built with equivariant blocks and positional encoding is used to recover expressiveness!

Thank You!

Références

W. Azizian and M. Lelarge. Expressive power of invariant and equivariant graph neural networks. *arXiv preprint arXiv :2006.15646*, 2020.

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning : going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4) :18–42, 2017.

Z. Fan, C. Mao, Y. Wu, and J. Xu. Spectral graph matching and regularized quadratic relaxations i : The gaussian model. *arXiv preprint arXiv :1907.08880*, 2019.

H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman. Provably powerful graph networks. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32 : Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 2153–2164, 2019. URL `http://papers.nips.cc/paper/8488-provably-powerful-graph-networks`.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet : Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.