Expressive Power of Graph Neural Networks

Pablo Barceló[1], Egor Kostylev[2], **Mikaël Monet**[1], Jorge Pérez[1], Juan Reutter[1]

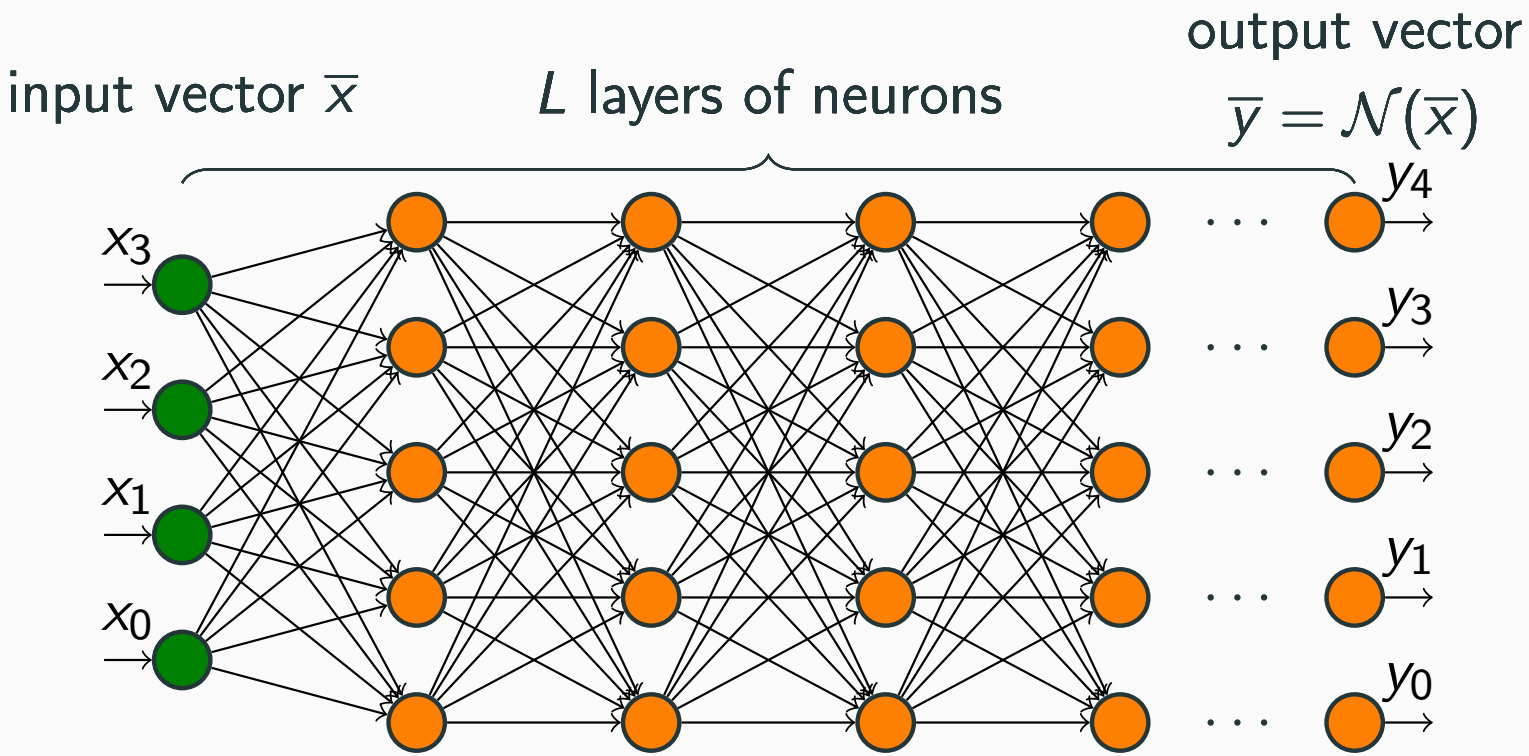
July 26th, 2019

[1]Millennium Institute for Foundational Research on Data, Chile

[2]Oxford University, UK

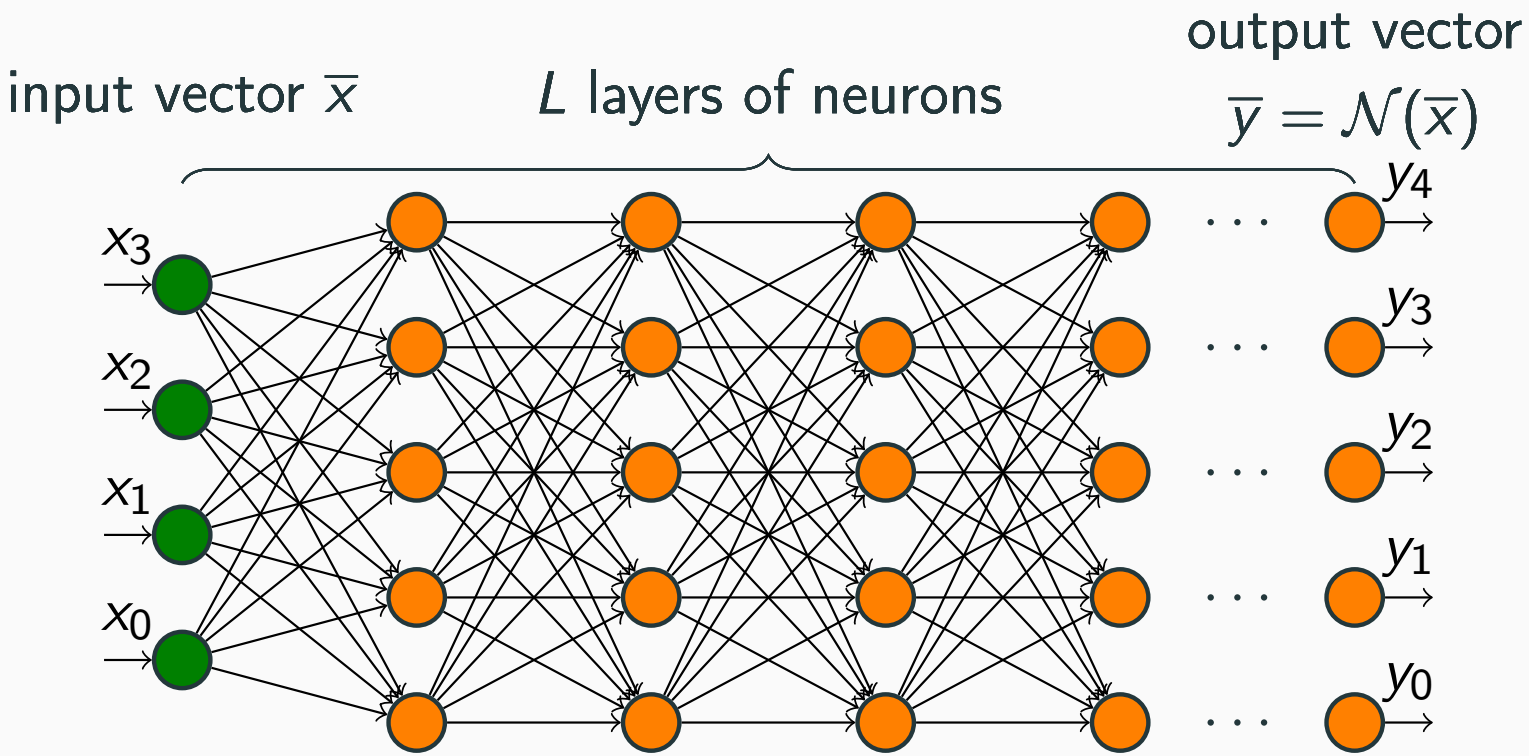
Motivation: why *graph* neural networks?

Neural Networks (NNs)



A fully connected neural network $\mathcal{N}$.

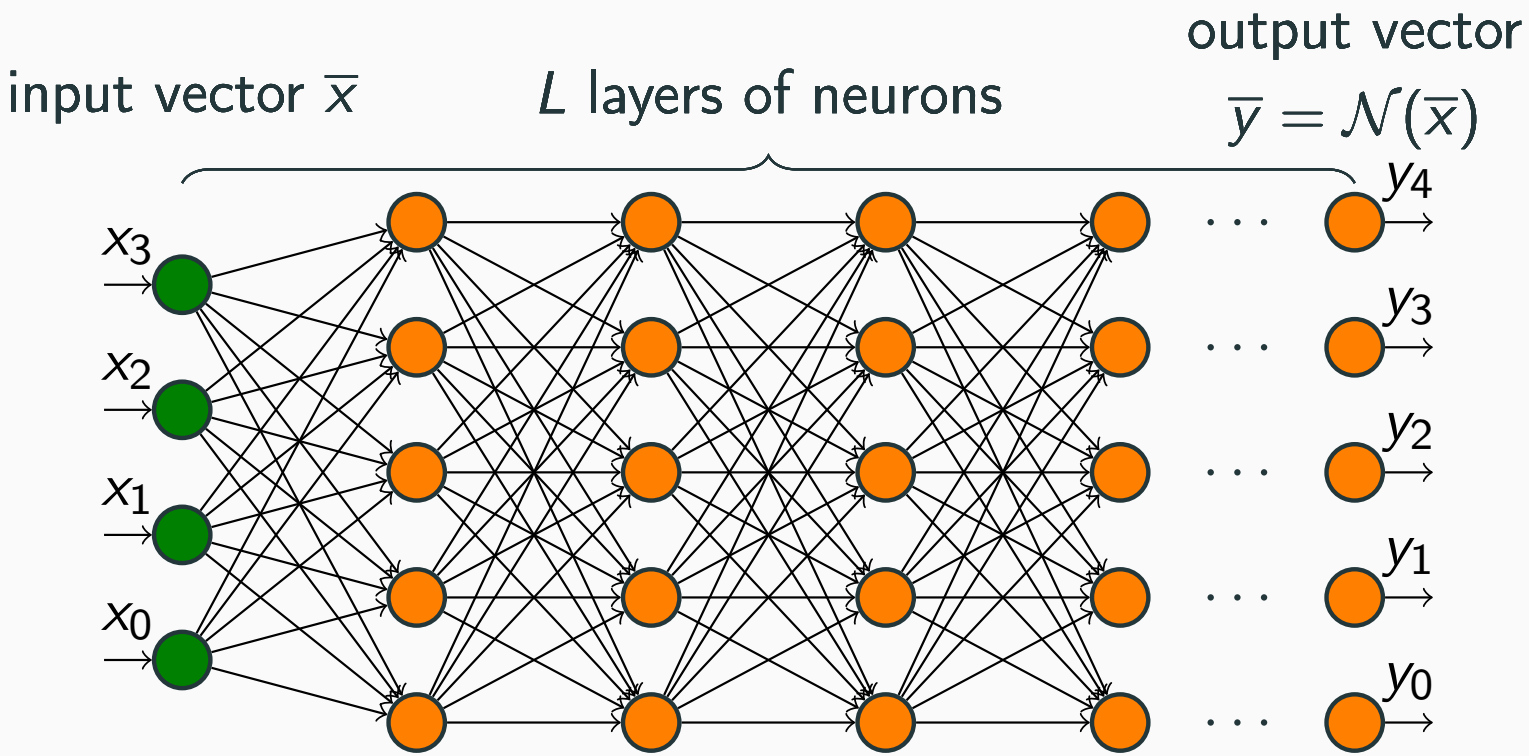
Neural Networks (NNs)



A fully connected neural network $\mathcal{N}$.

- Weight $w_{n' \to n}$ between two consecutive neurons

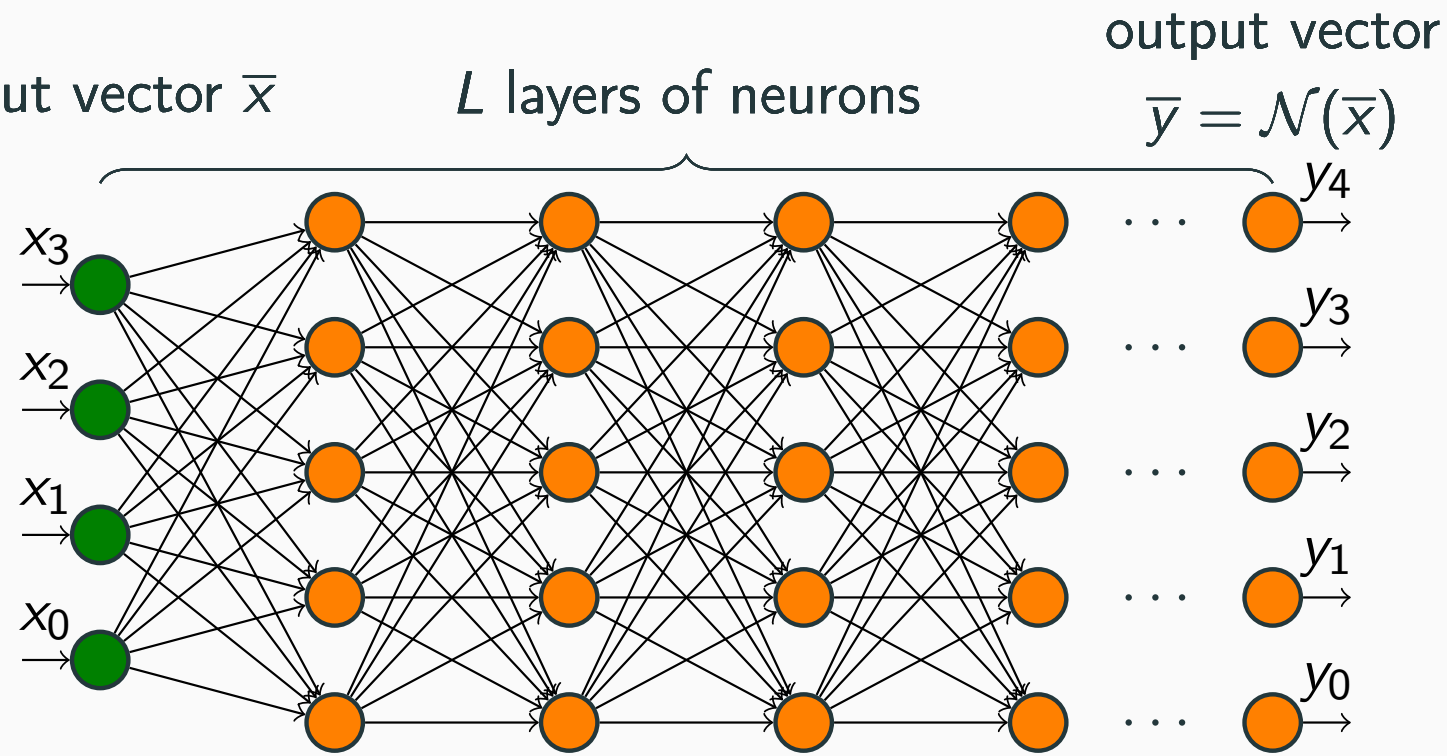
Neural Networks (NNs)



A fully connected neural network $\mathcal{N}$.

- Weight $w_{n' \to n}$ between two consecutive neurons
- Compute left to right $\lambda(n) \stackrel{\text{def}}{=} f(\sum w_{n' \to n} \times \lambda(n'))$

Neural Networks (NNs)



A fully connected neural network $\mathcal{N}$.

- Weight $w_{n' \to n}$ between two consecutive neurons
- Compute left to right $\lambda(n) \stackrel{\text{def}}{=} f(\sum w_{n' \to n} \times \lambda(n'))$
- Goal: find the weights that "solve" your problem (classification, clustering, regression, etc.)

Finding the weights

- Goal: find the weights that "solve" your problem

$\rightarrow$ minimize $\mathrm{Dist}(\mathcal{N}(\overline{x}), g(\overline{x}))$, where g is what you want to learn

Finding the weights

- Goal: find the weights that "solve" your problem

$\rightarrow$ minimize $\mathrm{Dist}(\mathcal{N}(\overline{x}), g(\overline{x}))$, where g is what you want to learn

$\rightarrow$ use backpropagation algorithm

Finding the weights

- Goal: find the weights that "solve" your problem

$\rightarrow$ minimize $\mathrm{Dist}(\mathcal{N}(\overline{x}), g(\overline{x}))$, where g is what you want to learn

$\rightarrow$ use backpropagation algorithm

- **Problem**: when a layer has many neurons, there are a lot of weights...

Finding the weights

- Goal: find the weights that "solve" your problem

$\rightarrow$ minimize $\mathrm{Dist}(\mathcal{N}(\overline{x}), g(\overline{x}))$, where g is what you want to learn

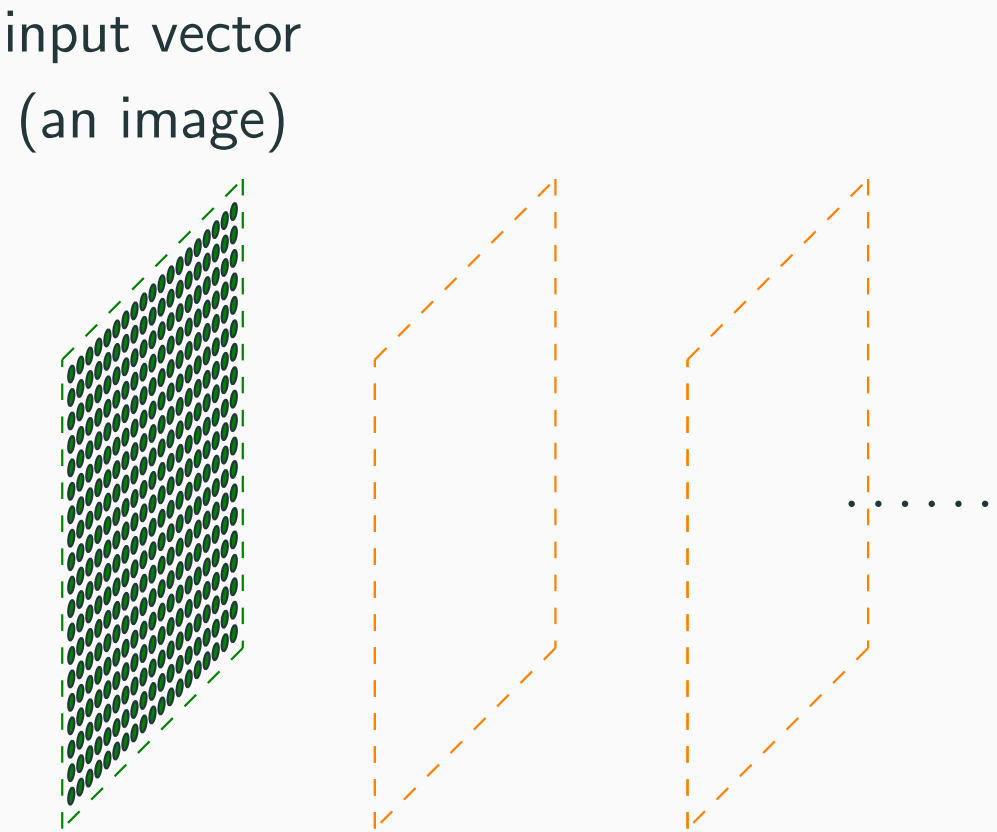
$\rightarrow$ use backpropagation algorithm

- **Problem**: when a layer has many neurons, there are a lot of weights...

$\rightarrow$ example: input is a 250×250 pixels image, and we want to build a fully connected NN with 500 neurons per layer

$\rightarrow$ between each consecutive layers we have $250 \times 250 \times 500 = 31,250,000$ weights

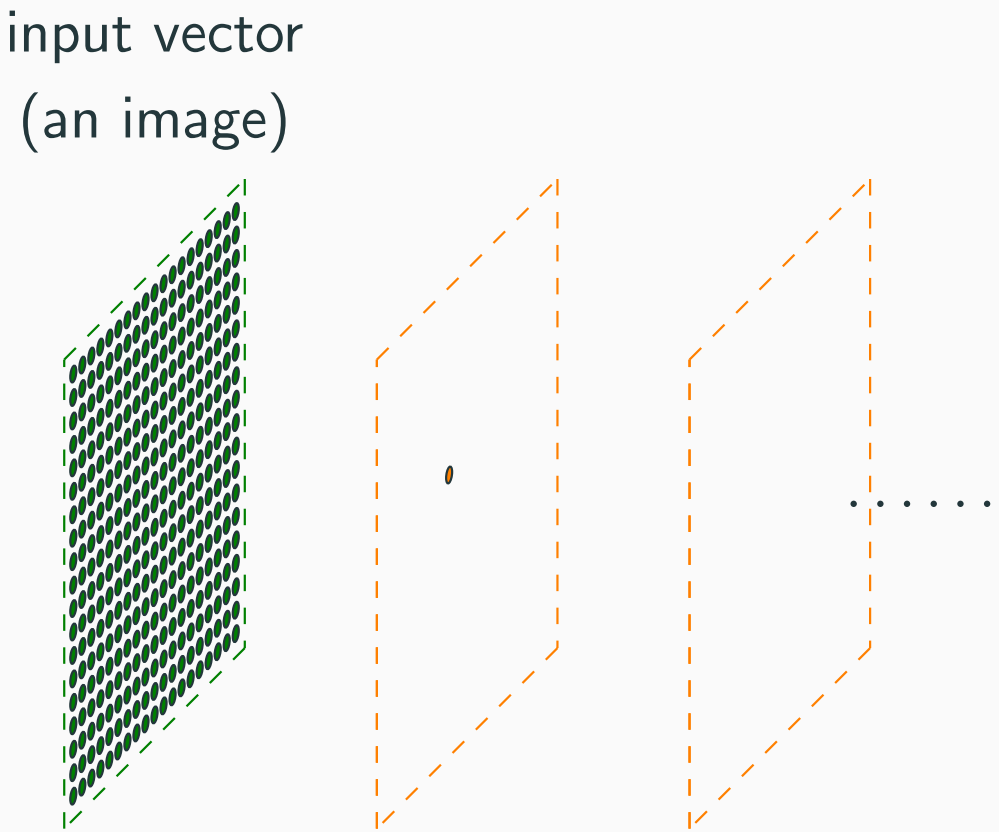
Convolutional Neural Networks



A convolutional neural network.

- **Idea**: use the structure of the data (here, a grid)

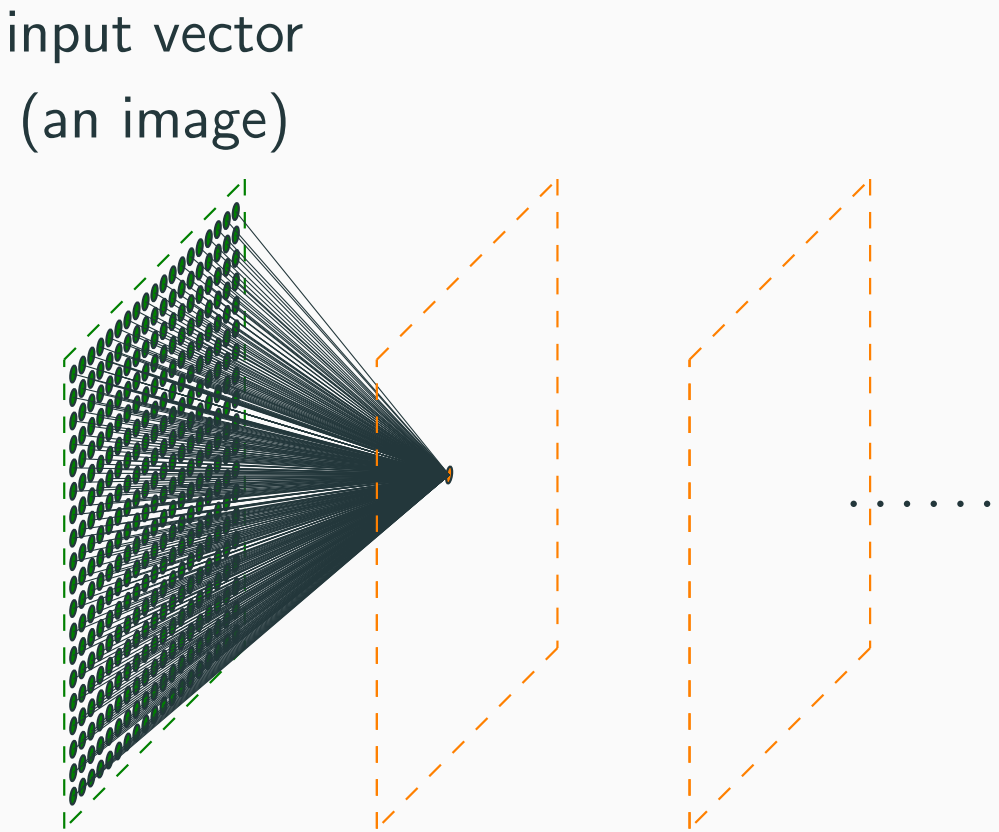
Convolutional Neural Networks



A convolutional neural network.

- **Idea**: use the structure of the data (here, a grid)

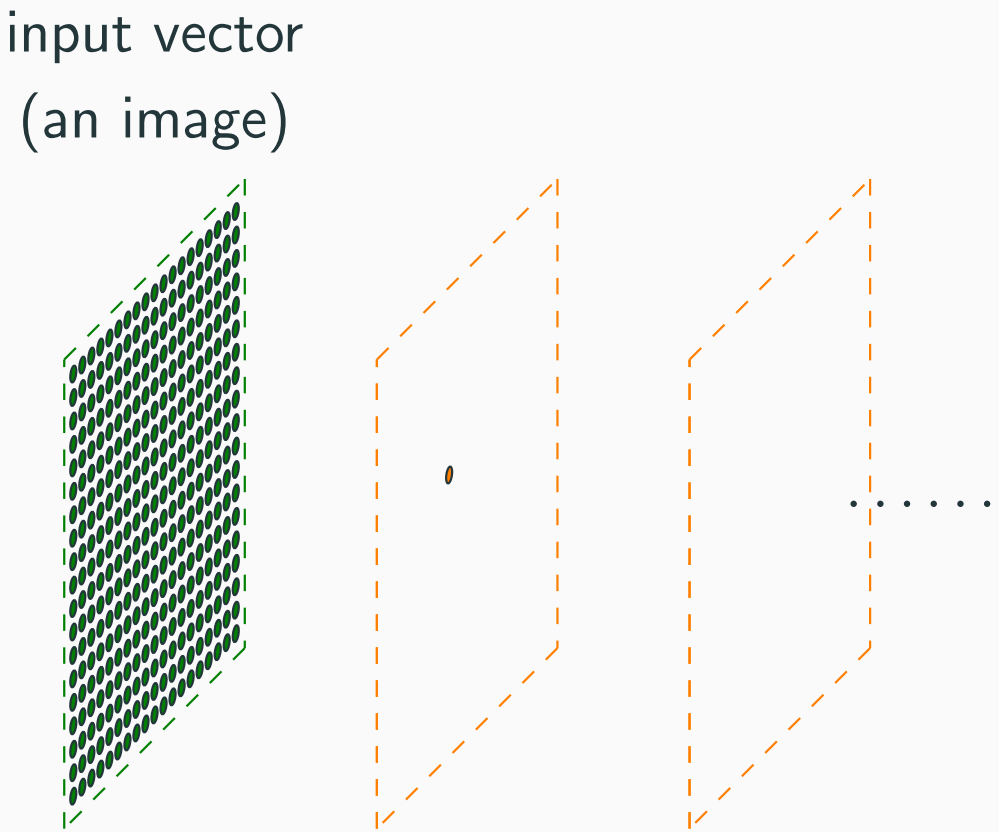
Convolutional Neural Networks



A convolutional neural network.

- **Idea**: use the structure of the data (here, a grid)

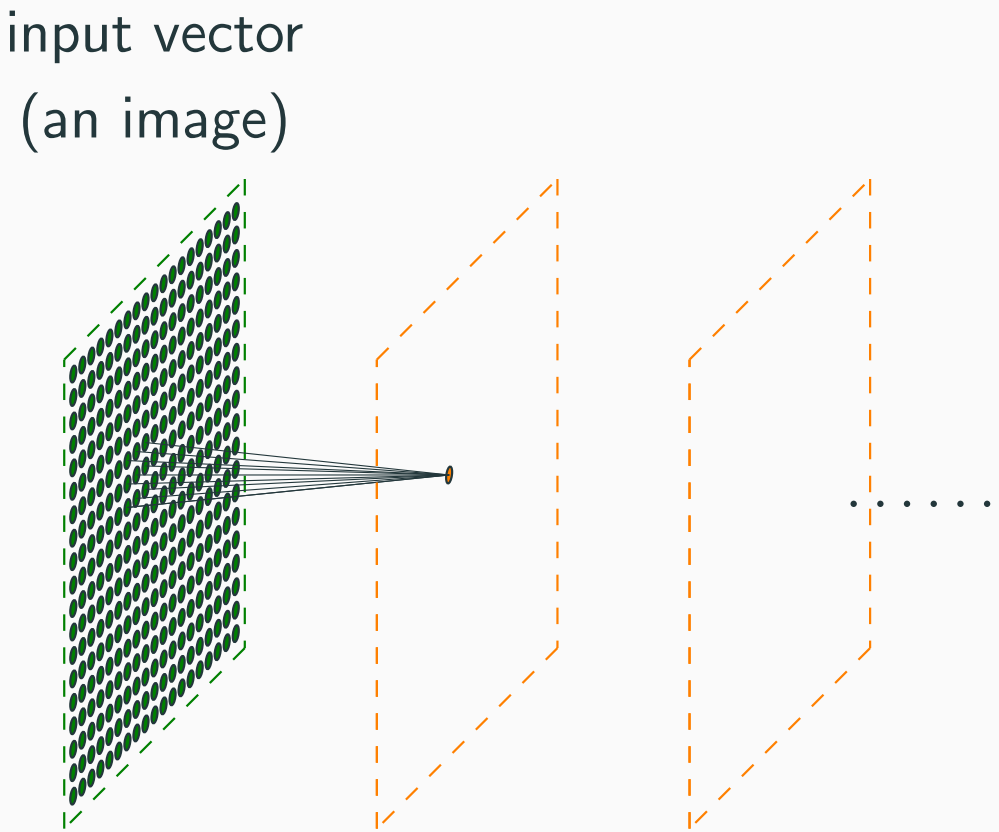
Convolutional Neural Networks



A convolutional neural network.

- **Idea**: use the structure of the data (here, a grid)

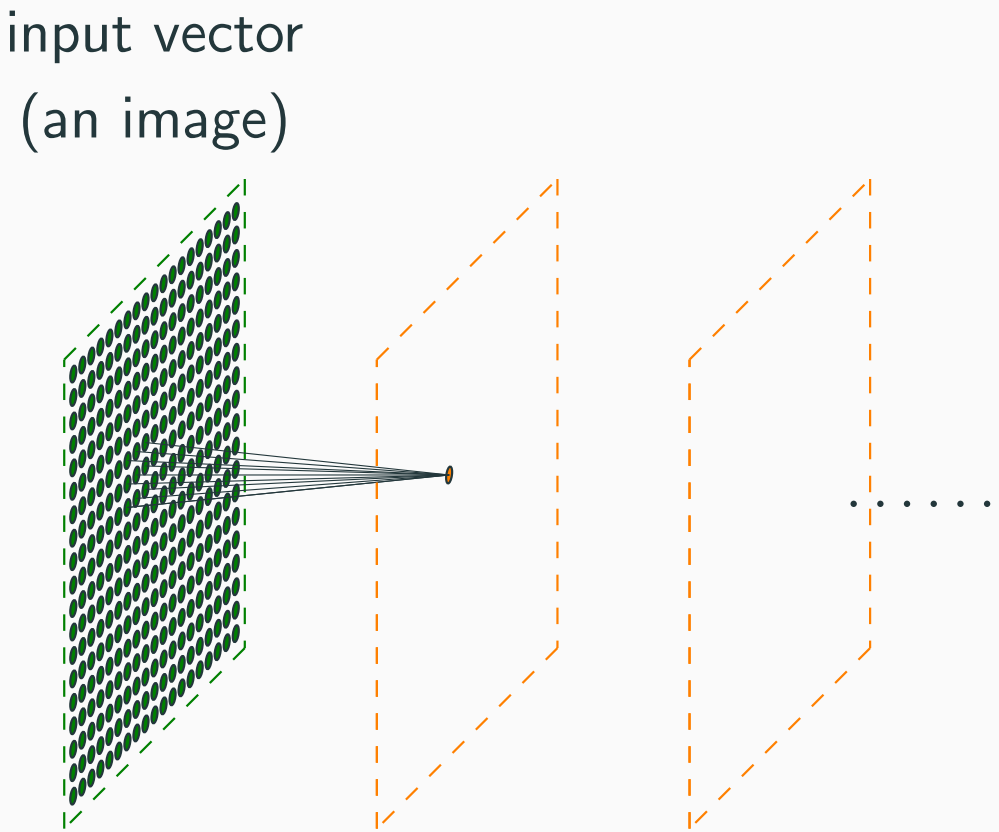
Convolutional Neural Networks



A convolutional neural network.

- **Idea**: use the structure of the data (here, a grid)

Convolutional Neural Networks

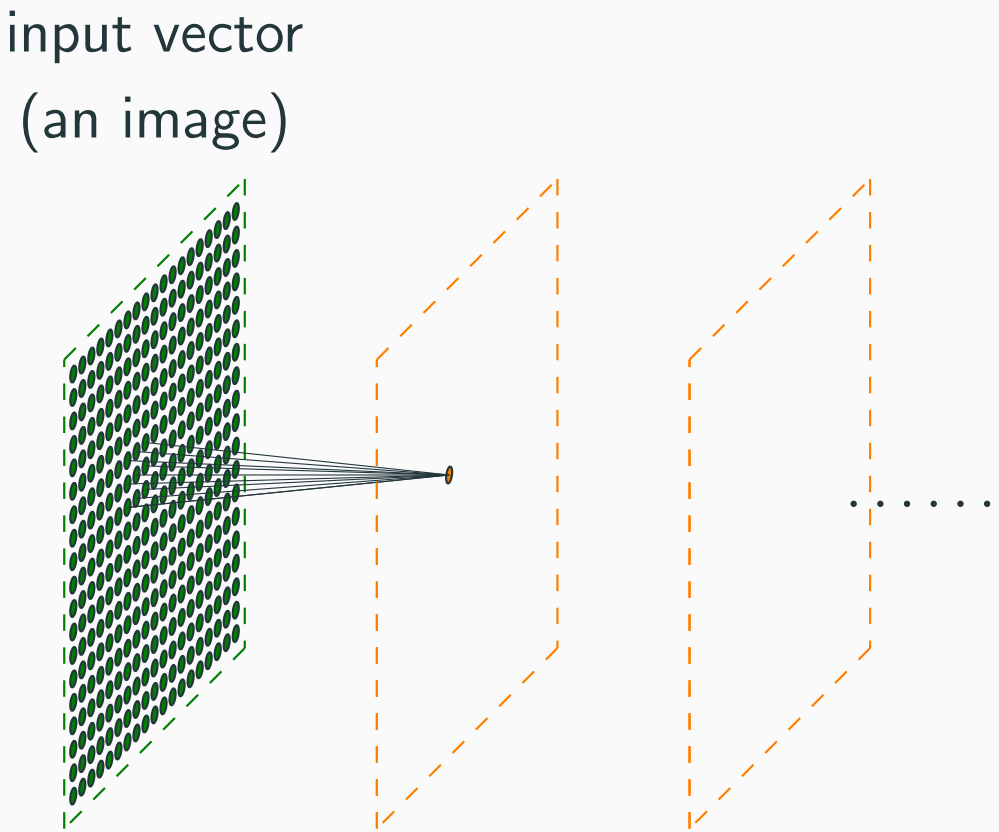


A convolutional neural network.

- **Idea**: use the structure of the data (here, a grid)

$\rightarrow$ fewer weights to learn (e.g, $500 * 9 = 4,500$ for the first layer)

Convolutional Neural Networks



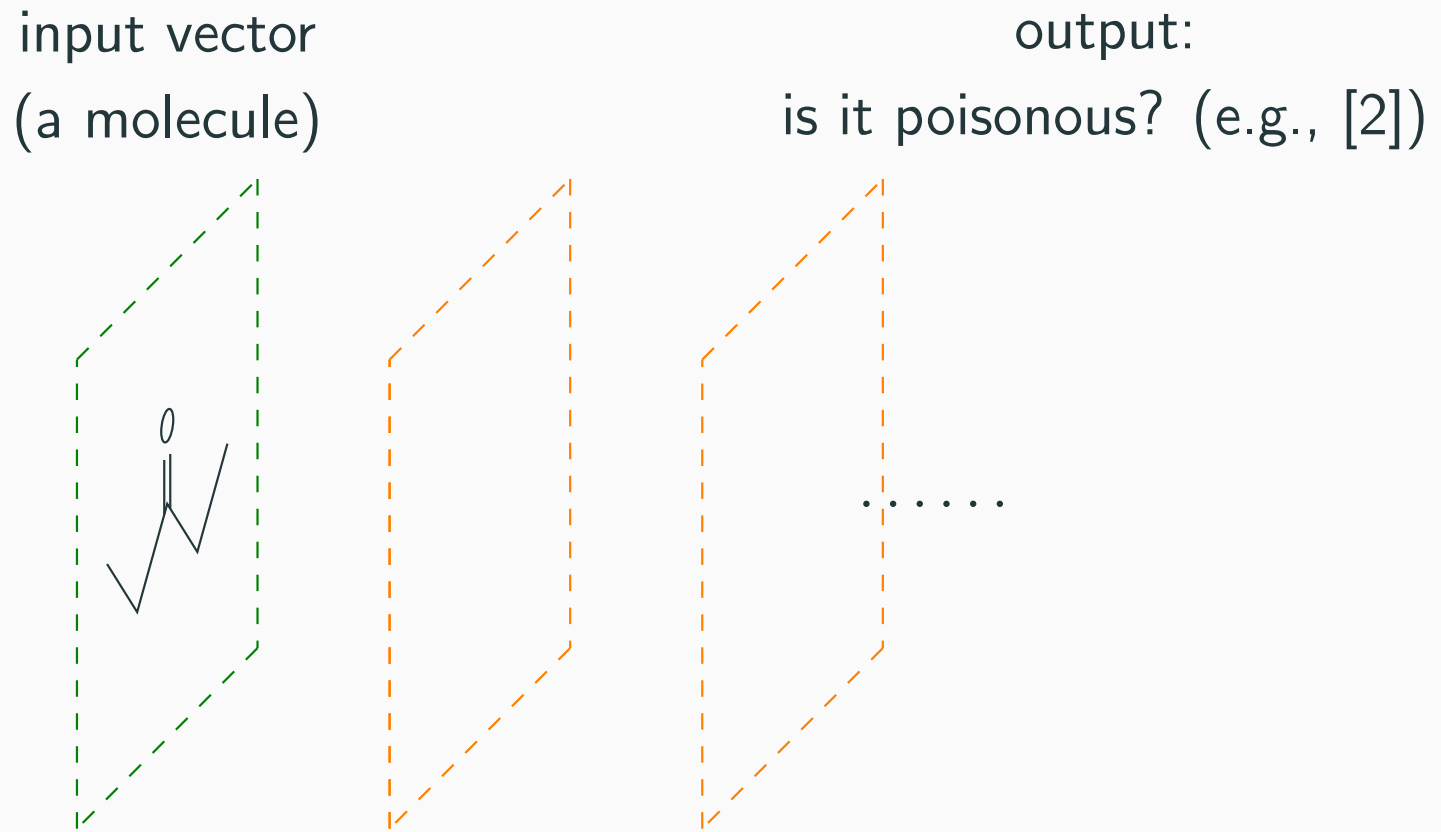
A convolutional neural network.

- **Idea**: use the structure of the data (here, a grid)

$\rightarrow$ fewer weights to learn (e.g, $500 * 9 = 4,500$ for the first layer)

$\rightarrow$ other advantages: recognize patterns that are local

Graph Neural Networks (GNNs)

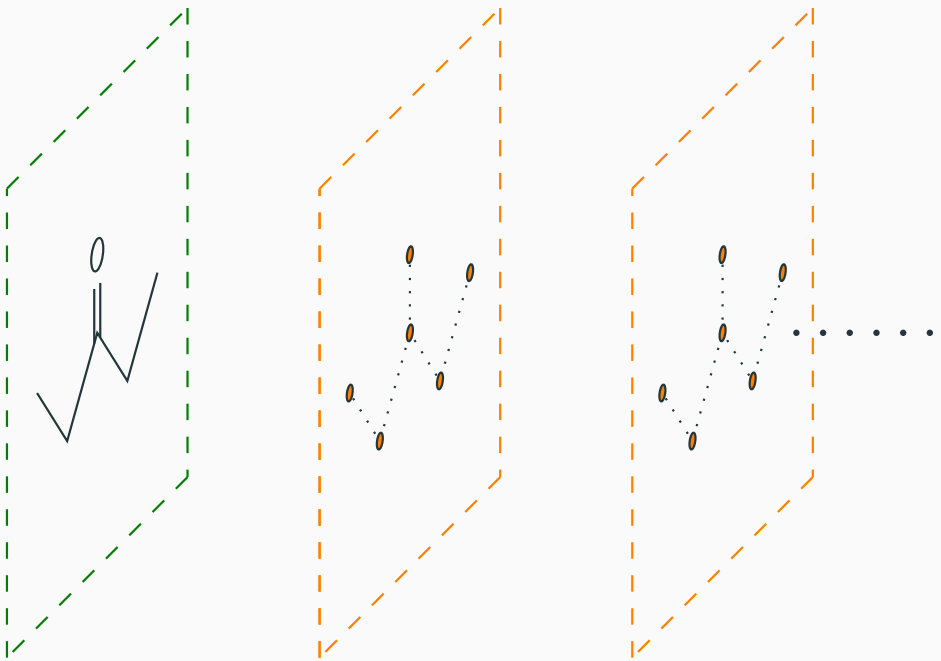


A graph neural network.

- **Idea**: use the structure of the data (here, a grid)

→ GNNs generalize this idea to allow *any* graph as input (in particular, not constrained with fixed-size inputs anymore)

Graph Neural Networks (GNNs)



A graph neural network.

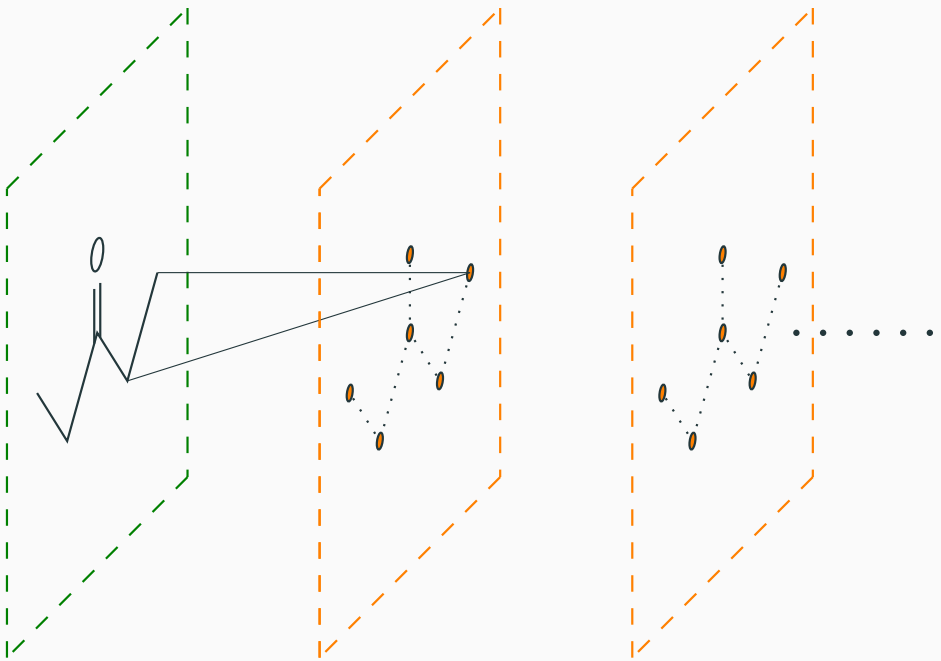
- **Idea**: use the structure of the data (here, a grid)

→ GNNs generalize this idea to allow *any* graph as input (in particular, not constrained with fixed-size inputs anymore)

Graph Neural Networks (GNNs)

input vector
(a molecule)

output:
is it poisonous? (e.g., [2])



A graph neural network.

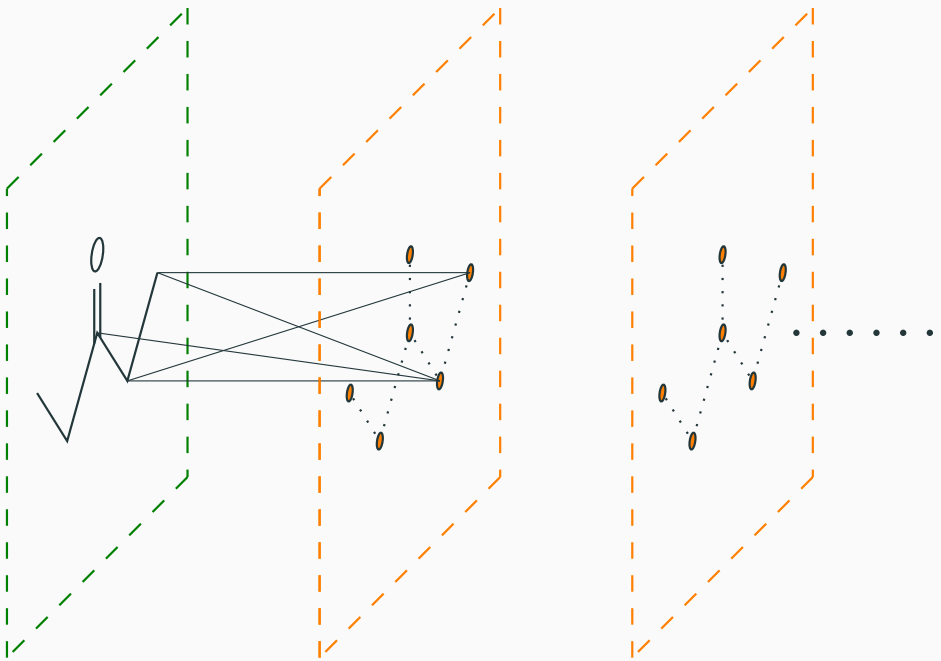
- **Idea**: use the structure of the data (here, a grid)

→ GNNs generalize this idea to allow *any* graph as input (in particular, not constrained with fixed-size inputs anymore)

Graph Neural Networks (GNNs)

input vector
(a molecule)

output:
is it poisonous? (e.g., [2])



A graph neural network.

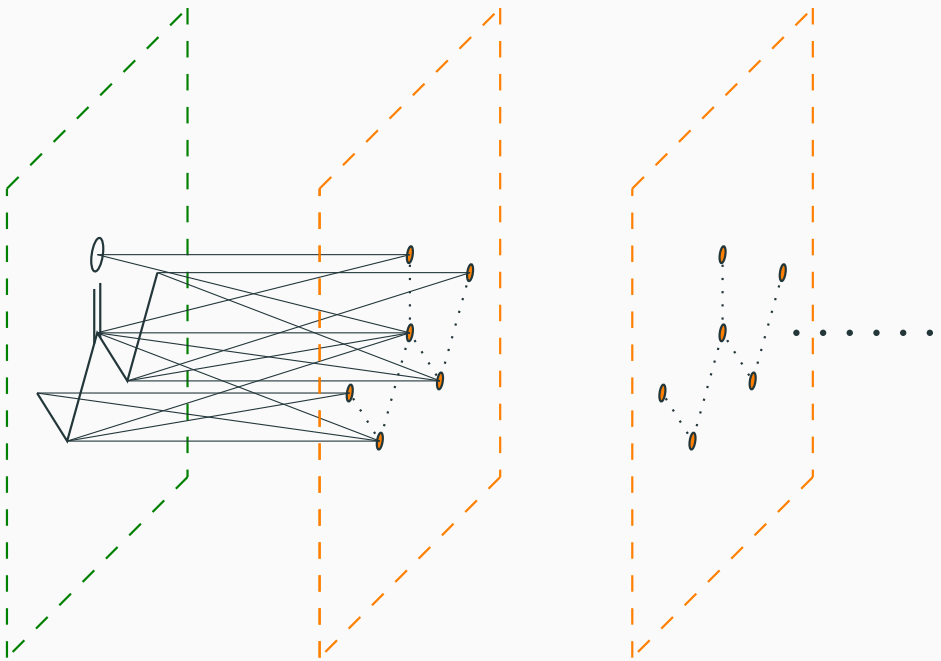
- **Idea**: use the structure of the data (here, a grid)

→ GNNs generalize this idea to allow *any* graph as input (in particular, not constrained with fixed-size inputs anymore)

Graph Neural Networks (GNNs)

input vector
(a molecule)

output:
is it poisonous? (e.g., [2])



A graph neural network.

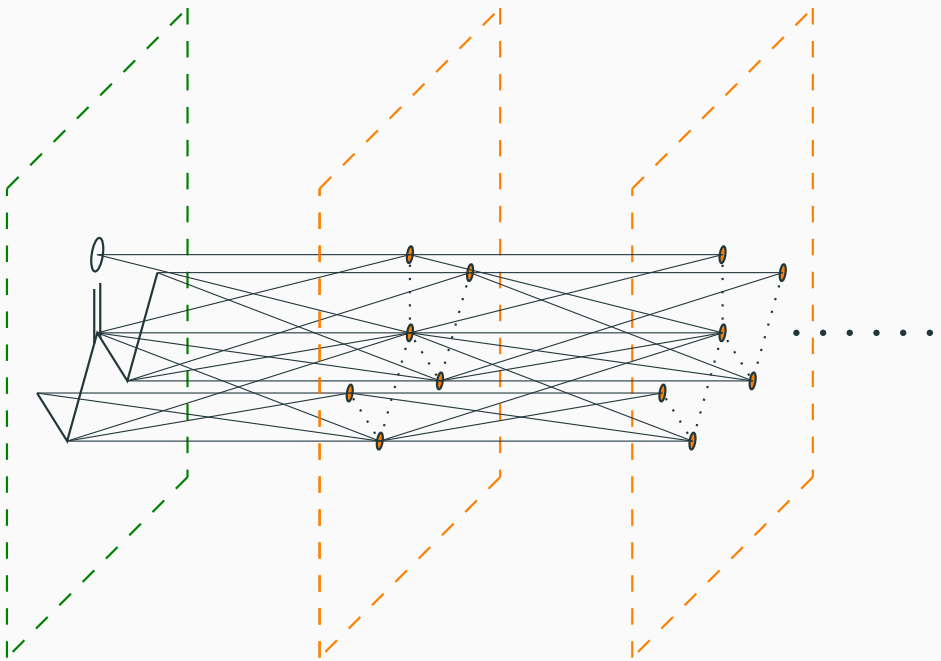
- **Idea**: use the structure of the data (here, a grid)

→ GNNs generalize this idea to allow *any* graph as input (in particular, not constrained with fixed-size inputs anymore)

Graph Neural Networks (GNNs)

input vector
(a molecule)

output:
is it poisonous? (e.g., [2])



A graph neural network.

- **Idea**: use the structure of the data (here, a grid)

→ GNNs generalize this idea to allow *any* graph as input (in particular, not constrained with fixed-size inputs anymore)

Question: what can we do with graph neural networks? (from a *theoretical* perspective)

GNNs: formalisation

- $\mathcal{N}_G(u) \stackrel{\text{def}}{=}$ closed neighborhood of node u in **labeled** graph $G = (V, E, \lambda)$, where $\lambda : V \to \mathbb{R}^d$

GNNs: formalisation

- $\mathcal{N}_G(u) \stackrel{\text{def}}{=}$ closed neighborhood of node u in **labeled** graph $G = (V, E, \lambda)$, where $\lambda : V \to \mathbb{R}^d$
- Run of a GNN $\mathcal{A}$ with L **layers** on G: iteratively compute $\mathcal{A}^i_G(u) \in \mathbb{R}^d$ for $0 \leq i \leq L$ as follows:

GNNs: formalisation

- $\mathcal{N}_G(u) \stackrel{\text{def}}{=}$ closed neighborhood of node u in **labeled** graph $G = (V, E, \lambda)$, where $\lambda : V \to \mathbb{R}^d$
- Run of a GNN $\mathcal{A}$ with L **layers** on G: iteratively compute $\mathcal{A}^i_G(u) \in \mathbb{R}^d$ for $0 \leq i \leq L$ as follows:
  - $\mathcal{A}^0_G(u) \stackrel{\text{def}}{=} \lambda(u)$

GNNs: formalisation

- $\mathcal{N}_G(u) \stackrel{\text{def}}{=}$ closed neighborhood of node u in **labeled** graph $G = (V, E, \lambda)$, where $\lambda : V \to \mathbb{R}^d$
- Run of a GNN $\mathcal{A}$ with L **layers** on G: iteratively compute $\mathcal{A}^i_G(u) \in \mathbb{R}^d$ for $0 \leq i \leq L$ as follows:
  - $\mathcal{A}^0_G(u) \stackrel{\text{def}}{=} \lambda(u)$
  - $\mathcal{A}^{i+1}_G(u) \stackrel{\text{def}}{=} \text{AGGREGATE}(\{\{\mathcal{A}^i_G(v) \mid v \in \mathcal{N}_G(u)\}\})$

GNNs: formalisation

- $\mathcal{N}_G(u) \stackrel{\text{def}}{=}$ closed neighborhood of node u in **labeled** graph $G = (V, E, \lambda)$, where $\lambda : V \to \mathbb{R}^d$
- Run of a GNN $\mathcal{A}$ with L **layers** on G: iteratively compute $\mathcal{A}^i_G(u) \in \mathbb{R}^d$ for $0 \leq i \leq L$ as follows:
  - $\mathcal{A}^0_G(u) \stackrel{\text{def}}{=} \lambda(u)$
  - $\mathcal{A}^{i+1}_G(u) \stackrel{\text{def}}{=} \text{AGGREGATE}(\{\{\mathcal{A}^i_G(v) \mid v \in \mathcal{N}_G(u)\}\})$
- Then, optionally, read the value of all output neurons, and output something
  - $\text{READOUT}(\{\{\mathcal{A}^L_G(u) \mid u \in V\}\})$

GNNs: formalisation

- $\mathcal{N}_G(u) \stackrel{\text{def}}{=}$ closed neighborhood of node u in **labeled** graph $G = (V, E, \lambda)$, where $\lambda : V \to \mathbb{R}^d$
- Run of a GNN $\mathcal{A}$ with L **layers** on G: iteratively compute $\mathcal{A}^i_G(u) \in \mathbb{R}^d$ for $0 \leq i \leq L$ as follows:
  - $\mathcal{A}^0_G(u) \stackrel{\text{def}}{=} \lambda(u)$
  - $\mathcal{A}^{i+1}_G(u) \stackrel{\text{def}}{=} \text{AGGREGATE}(\{\{\mathcal{A}^i_G(v) \mid v \in \mathcal{N}_G(u)\}\})$
- Then, optionally, read the value of all output neurons, and output something
  - $\text{READOUT}(\{\{\mathcal{A}^L_G(u) \mid u \in V\}\})$

Where AGGREGATE and READOUT are what define $\mathcal{A}$

GNNs: what can they do? Related work (1/2)

- $\mathcal{A}^0_G(u) \stackrel{\text{def}}{=} \lambda(u)$
- $\mathcal{A}^{i+1}_G(u) \stackrel{\text{def}}{=} \text{AGGREGATE}(\{\{\mathcal{A}^i_G(v) \mid v \in \mathcal{N}_G(u)\}\})$
- $\text{READOUT}(\{\{\mathcal{A}^L_G(u) \mid u \in V\}\})$

GNNs: what can they do? Related work (1/2)

- $\mathcal{A}^0_G(u) \stackrel{\text{def}}{=} \lambda(u)$
- $\mathcal{A}^{i+1}_G(u) \stackrel{\text{def}}{=} \text{AGGREGATE}(\{\{\mathcal{A}^i_G(v) \mid v \in \mathcal{N}_G(u)\}\})$
- $\text{READOUT}(\{\{\mathcal{A}^L_G(u) \mid u \in V\}\})$

- [3, 4] say: for classification of nodes in a given graph, or for classification of graphs, GNNs are always **less discriminative** than the *Weisfeiler-Lehman (WL) isomorphism test*

GNNs: what can they do? Related work (1/2)

- $\mathcal{A}^0_G(u) \stackrel{\text{def}}{=} \lambda(u)$
- $\mathcal{A}^{i+1}_G(u) \stackrel{\text{def}}{=} \text{AGGREGATE}(\{\{\mathcal{A}^i_G(v) \mid v \in \mathcal{N}_G(u)\}\})$
- $\text{READOUT}(\{\{\mathcal{A}^L_G(u) \mid u \in V\}\})$

- [3, 4] say: for classification of nodes in a given graph, or for classification of graphs, GNNs are always **less discriminative** than the *Weisfeiler-Lehman (WL) isomorphism test*

$\rightarrow$ (because this test works exactly like GNNs do, with an AGGREGATE function that is injective)

GNNs: what can they do? Related work (1/2)

- $\mathcal{A}^0_G(u) \stackrel{\text{def}}{=} \lambda(u)$
- $\mathcal{A}^{i+1}_G(u) \stackrel{\text{def}}{=} \text{AGGREGATE}(\{\{\mathcal{A}^i_G(v) \mid v \in \mathcal{N}_G(u)\}\})$
- $\text{READOUT}(\{\{\mathcal{A}^L_G(u) \mid u \in V\}\})$

- [3, 4] say: for classification of nodes in a given graph, or for classification of graphs, GNNs are always **less discriminative** than the *Weisfeiler-Lehman (WL) isomorphism test*

$\rightarrow$ (because this test works exactly like GNNs do, with an AGGREGATE function that is injective)

- [3, 4]: moreover, there is a way to construct a GNN $\mathcal{A}_{\text{WL}}$ such that $\mathcal{A}$ is as discriminative as WL

GNNs: what can they do? Related work (2/2)

- There is a link between the WL test and FO with 2 variables and counting (FOC_2)

$\rightarrow$ example: $(\exists^{\geq 10} x).(\exists^{\geq 5} y).E(x,y) \vee (\exists^{\geq 2} x).E(y,x) \wedge C(x)$

GNNs: what can they do? Related work (2/2)

- There is a link between the WL test and FO with 2 variables and counting (FOC_2)

$\rightarrow$ example: $(\exists^{\geq 10} x).(\exists^{\geq 5} y).E(x,y) \vee (\exists^{\geq 2} x).E(y,x) \wedge C(x)$

- [1]: we have $\mathrm{WL}^i(G_1) = \mathrm{WL}^i(G_2)$ if and only if G_1 and G_2 agree on all FOC_2 sentences of quantifier depth $\leq i$

However, we think that the links between GNNs and logics are not very well understood

Things we are working on

- Given a unary formula $\varphi(x)$, say that φ is expressible by a GNN with no $\mathrm{READOUT}$ if there is a GNN $\mathcal{A}_\varphi$ such that for any input graph G, $\mathcal{A}_\varphi$ labels each node u of G with 0 or 1, depending if $G, u \models \varphi(x)$

Things we are working on

- Given a unary formula $\varphi(x)$, say that φ is expressible by a GNN with no $\mathrm{READOUT}$ if there is a GNN $\mathcal{A}_\varphi$ such that for any input graph G, $\mathcal{A}_\varphi$ labels each node u of G with 0 or 1, depending if $G, u \models \varphi(x)$
- **Question**: which unary formulas are expressible by a GNN without $\mathrm{READOUT}$?

Things we are working on

- Given a unary formula $\varphi(x)$, say that φ is expressible by a GNN with no $\mathrm{READOUT}$ if there is a GNN $\mathcal{A}_\varphi$ such that for any input graph G, $\mathcal{A}_\varphi$ labels each node u of G with 0 or 1, depending if $G, u \models \varphi(x)$
- **Question**: which unary formulas are expressible by a GNN without $\mathrm{READOUT}$?

$\rightarrow$ $(\exists y, z).E(x, y) \wedge E(y, z) \wedge E(z, x)$?

Things we are working on

- Given a unary formula $\varphi(x)$, say that φ is expressible by a GNN with no $\mathrm{READOUT}$ if there is a GNN $\mathcal{A}_\varphi$ such that for any input graph G, $\mathcal{A}_\varphi$ labels each node u of G with 0 or 1, depending if $G, u \models \varphi(x)$
- **Question**: which unary formulas are expressible by a GNN without $\mathrm{READOUT}$?

$\rightarrow$ $(\exists y, z).E(x, y) \wedge E(y, z) \wedge E(z, x)$?NO

Things we are working on

- Given a unary formula $\varphi(x)$, say that φ is expressible by a GNN with no $\mathrm{READOUT}$ if there is a GNN $\mathcal{A}_\varphi$ such that for any input graph G, $\mathcal{A}_\varphi$ labels each node u of G with 0 or 1, depending if $G, u \models \varphi(x)$
- **Question**: which unary formulas are expressible by a GNN without $\mathrm{READOUT}$?

$\rightarrow$ $(\exists y, z).E(x, y) \wedge E(y, z) \wedge E(z, x)$?NO

$\rightarrow$ $C(x) \wedge (\exists^{\geq 100} y).\mathrm{True}$?

Things we are working on

- Given a unary formula $\varphi(x)$, say that φ is expressible by a GNN with no $\mathrm{READOUT}$ if there is a GNN $\mathcal{A}_\varphi$ such that for any input graph G, $\mathcal{A}_\varphi$ labels each node u of G with 0 or 1, depending if $G, u \models \varphi(x)$
- **Question**: which unary formulas are expressible by a GNN without $\mathrm{READOUT}$?

$\rightarrow$ $(\exists y, z).E(x, y) \wedge E(y, z) \wedge E(z, x)$?NO

$\rightarrow$ $C(x) \wedge (\exists^{\geq 100} y).\mathrm{True}$?NO

Things we are working on

- Given a unary formula $\varphi(x)$, say that φ is expressible by a GNN with no $\mathrm{READOUT}$ if there is a GNN $\mathcal{A}_\varphi$ such that for any input graph G, $\mathcal{A}_\varphi$ labels each node u of G with 0 or 1, depending if $G, u \models \varphi(x)$
- **Question**: which unary formulas are expressible by a GNN without $\mathrm{READOUT}$?

$\rightarrow$ $(\exists y, z).E(x, y) \wedge E(y, z) \wedge E(z, x)$?NO

$\rightarrow$ $C(x) \wedge (\exists^{\geq 100} y).\mathrm{True}$?NO

$\rightarrow$ $(\exists n \in \mathbb{N}).(\exists^{!n} y).E(x, y) \wedge (\exists^{!2n} x).E(y, x) \wedge C(x)$?

Things we are working on

- Given a unary formula $\varphi(x)$, say that φ is expressible by a GNN with no $\mathrm{READOUT}$ if there is a GNN $\mathcal{A}_\varphi$ such that for any input graph G, $\mathcal{A}_\varphi$ labels each node u of G with 0 or 1, depending if $G, u \models \varphi(x)$
- **Question**: which unary formulas are expressible by a GNN without $\mathrm{READOUT}$?

$\rightarrow$ $(\exists y, z).E(x,y) \wedge E(y,z) \wedge E(z,x)$?NO

$\rightarrow$ $C(x) \wedge (\exists^{\geq 100} y).\mathrm{True}$?NO

$\rightarrow$ $(\exists n \in \mathbb{N}).(\exists^{!n} y).E(x,y) \wedge (\exists^{!2n} x).E(y,x) \wedge C(x)$?YES

Things we are working on

- Given a unary formula $\varphi(x)$, say that φ is expressible by a GNN with no $\mathrm{READOUT}$ if there is a GNN $\mathcal{A}_\varphi$ such that for any input graph G, $\mathcal{A}_\varphi$ labels each node u of G with 0 or 1, depending if $G, u \models \varphi(x)$
- **Question**: which unary formulas are expressible by a GNN without $\mathrm{READOUT}$?

$\rightarrow$ $(\exists y, z).E(x, y) \wedge E(y, z) \wedge E(z, x)$?NO

$\rightarrow$ $C(x) \wedge (\exists^{\geq 100} y).\mathrm{True}$?NO

$\rightarrow$ $(\exists n \in \mathbb{N}).(\exists^{!n} y).E(x, y) \wedge (\exists^{!2n} x).E(y, x) \wedge C(x)$?YES

- Same question for sentences, with $\mathrm{READOUT}$?

Things we are working on

- Given a unary formula $\varphi(x)$, say that φ is expressible by a GNN with no $\mathrm{READOUT}$ if there is a GNN $\mathcal{A}_\varphi$ such that for any input graph G, $\mathcal{A}_\varphi$ labels each node u of G with 0 or 1, depending if $G, u \models \varphi(x)$
- **Question**: which unary formulas are expressible by a GNN without $\mathrm{READOUT}$?

$\rightarrow$ $(\exists y, z).E(x,y) \wedge E(y,z) \wedge E(z,x)$?NO

$\rightarrow$ $C(x) \wedge (\exists^{\geq 100} y).\mathrm{True}$?NO

$\rightarrow$ $(\exists n \in \mathbb{N}).(\exists^{!n} y).E(x,y) \wedge (\exists^{!2n} x).E(y,x) \wedge C(x)$?YES

- Same question for sentences, with $\mathrm{READOUT}$?
- Given a GNN, can we find an equivalent unary formula? (explainability, regularisation)

Things we are working on

- Given a unary formula $\varphi(x)$, say that φ is expressible by a GNN with no $\mathrm{READOUT}$ if there is a GNN $\mathcal{A}_\varphi$ such that for any input graph G, $\mathcal{A}_\varphi$ labels each node u of G with 0 or 1, depending if $G, u \models \varphi(x)$
- **Question**: which unary formulas are expressible by a GNN without $\mathrm{READOUT}$?

$\rightarrow$ $(\exists y, z).E(x,y) \wedge E(y,z) \wedge E(z,x)$?NO

$\rightarrow$ $C(x) \wedge (\exists^{\geq 100} y).\mathrm{True}$?NO

$\rightarrow$ $(\exists n \in \mathbb{N}).(\exists^{!n} y).E(x,y) \wedge (\exists^{!2n} x).E(y,x) \wedge C(x)$?YES

- Same question for sentences, with $\mathrm{READOUT}$?
- Given a GNN, can we find an equivalent unary formula? (explainability, regularisation)

Thanks for your attention!

Bibliography I

Jin-Yi Cai, Martin Fürer, and Neil Immerman.
An optimal lower bound on the number of variables for graph identification.
Combinatorica, 12(4):389–410, 1992.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams.
Convolutional networks on graphs for learning molecular fingerprints.
In *Advances in neural information processing systems*, pages 2224–2232, 2015.

Bibliography II


Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. **Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks.**
arXiv preprint arXiv:1810.02244, 2018.


Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. **How Powerful are Graph Neural Networks?**
arXiv preprint arXiv:1810.00826, 2018.