

Pontificia Universidad Católica de Chile  
Escuela de Ingeniería  
Departamento de Ingeniería de Transporte y Logística



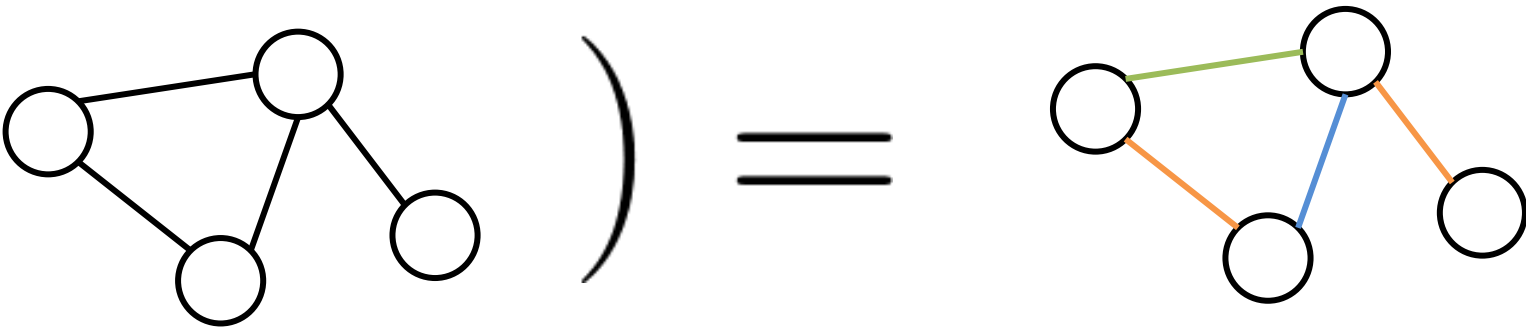
# Sistemas Urbanos Inteligentes

Redes convolucionales para grafos

Hans Löbel

Dpto. Ingeniería de Transporte y Logística  
Dpto. Ciencia de la Computación

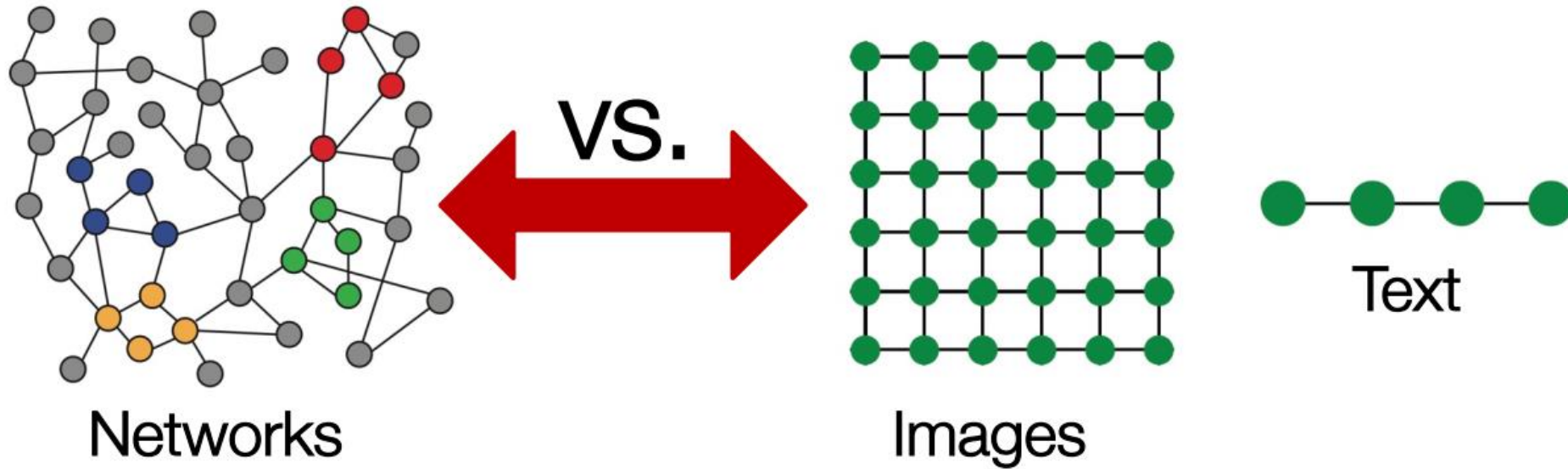
No olvidemos lo que buscamos

$$f \left( \text{graph} \right) = \text{graph}$$


The diagram illustrates a function  $f$  applied to a graph. The input graph on the left consists of four nodes and four edges forming a triangle with an additional node connected to one of the triangle's vertices. The output graph on the right is identical in structure but has colored edges: a green edge, a blue edge, and two orange edges.

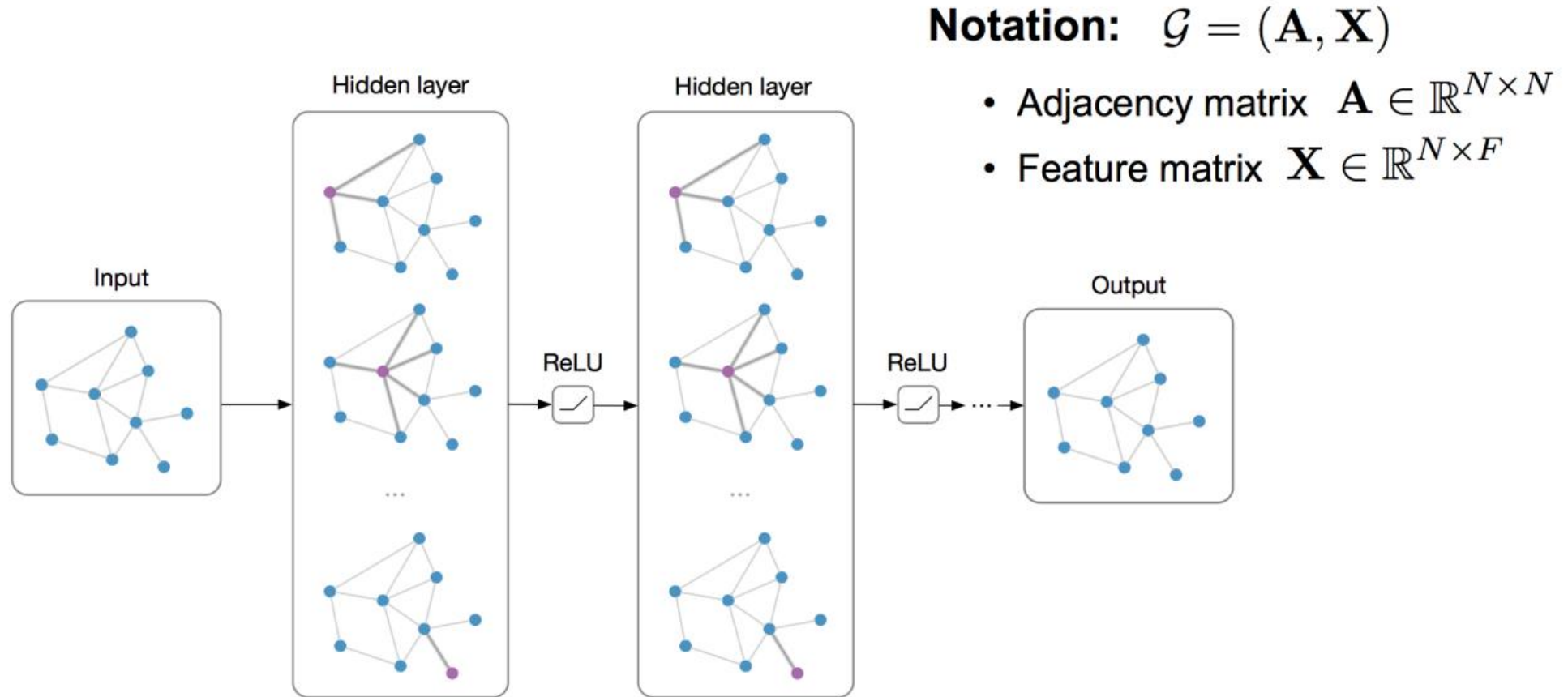
Si queremos utilizar redes neuronales para parametrizar  $f$ ,  
necesitamos que esta sea **differentiable**, **componible** y **escalable**.

Ni por qué es difícil



Los grafos son estructuras complejas: tamaño arbitrario, no existe el concepto de orden ni de punto de referencia, dinámicos, etc.

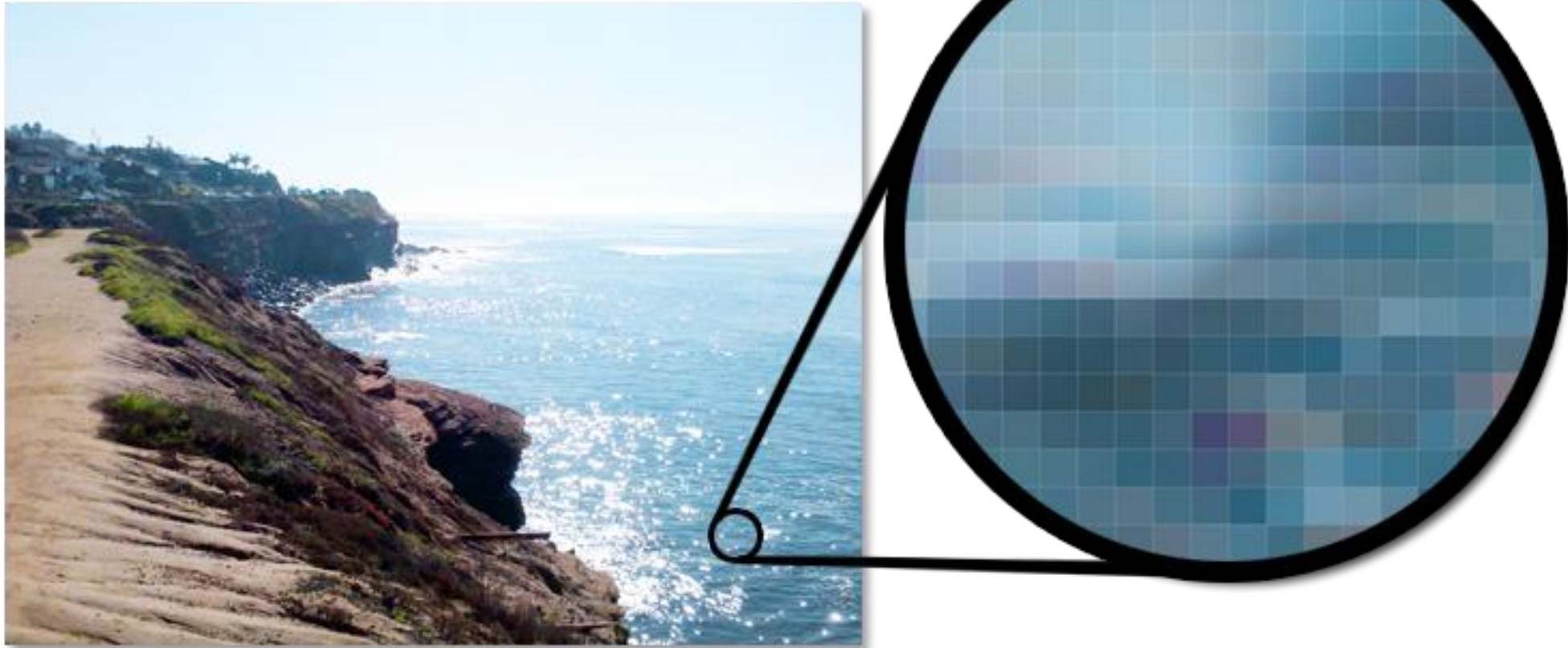
## Hacia dónde vamos



**Idea principal:** pasar mensajes entre nodos y combinarlos

**Otra perspectiva más ML:** pasar mensajes entre nodos para refinar la representación

Recordemos la idea detrás de las convoluciones



Si descomponemos la convolución, tenemos una parte que actúa de forma local y otra sobre el vecindario

$$f \left( \begin{array}{|c|c|c|c|} \hline \square & \square & \square & \square \\ \hline \square & \textcolor{blue}{\square} & \textcolor{blue}{\square} & \textcolor{blue}{\square} \\ \hline \square & \textcolor{blue}{\square} & \textcolor{red}{\square} & \textcolor{blue}{\square} \\ \hline \square & \textcolor{blue}{\square} & \textcolor{blue}{\square} & \textcolor{blue}{\square} \\ \hline \end{array} \right) = \mathbf{a}^T \textcolor{red}{\square} + \sum_{j \in \begin{array}{|c|c|} \hline \textcolor{blue}{\square} & \textcolor{blue}{\square} \\ \hline \textcolor{blue}{\square} & \textcolor{blue}{\square} \\ \hline \textcolor{blue}{\square} & \textcolor{blue}{\square} \\ \hline \end{array}} \mathbf{c}_j^T \textcolor{blue}{\square}$$

Operación local  
(pixel central)

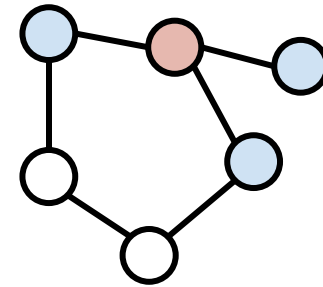
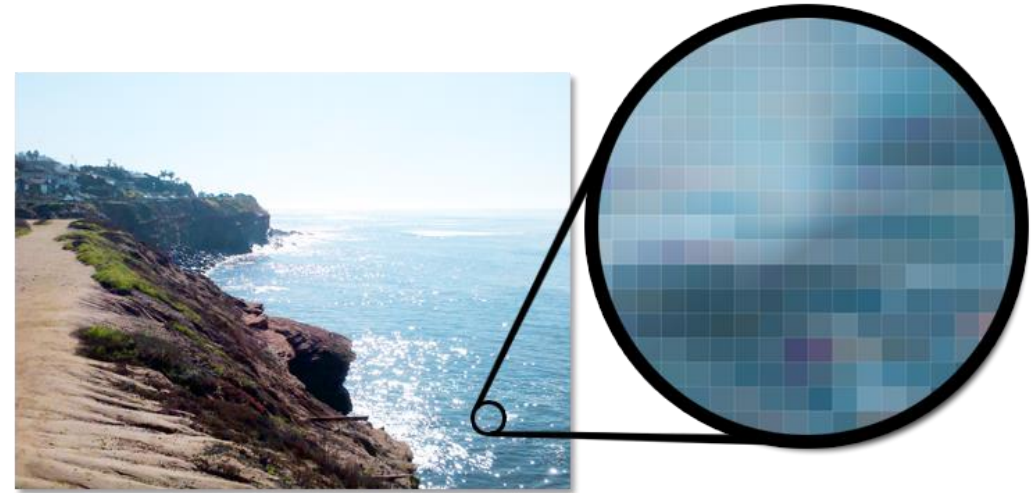
Agregación

Operación sobre  
vecinos (resto de  
los pixeles)

# ¿En qué se diferencian las imágenes de los grafos?

La clave es fijarse en la estructura local

- ✓ Tiene sentido la localidad (vecindario)
- ✗ Tamaño del vecinario es fijo
- ✗ Orden del vecindario es fijo



## Convoluciones en grafos (versión simplificada)

$$f\left(\begin{array}{c} \text{blue} \quad \text{red} \quad \text{blue} \\ | \quad / \quad \backslash \\ \text{white} \quad \text{blue} \\ | \quad \backslash \\ \text{white} \end{array}\right) = W_0^T \text{red} + \sum_{j \in \begin{array}{c} \text{blue} \quad \text{white} \quad \text{blue} \\ | \quad / \quad \backslash \\ \text{blue} \end{array}} W_1^T \text{blue}$$

Operación local

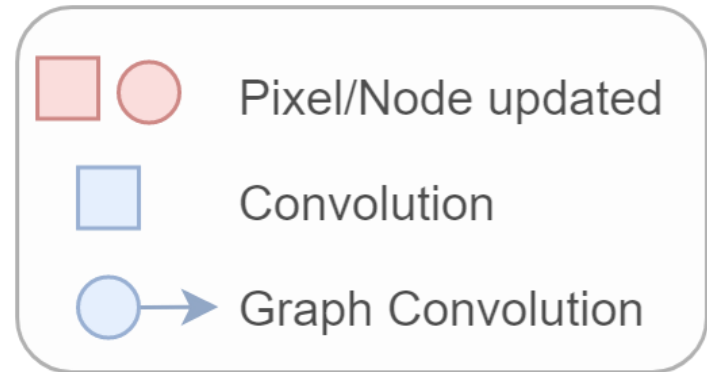
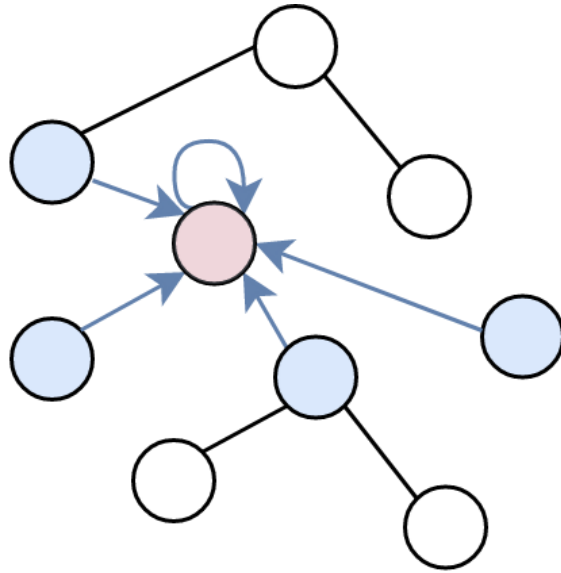
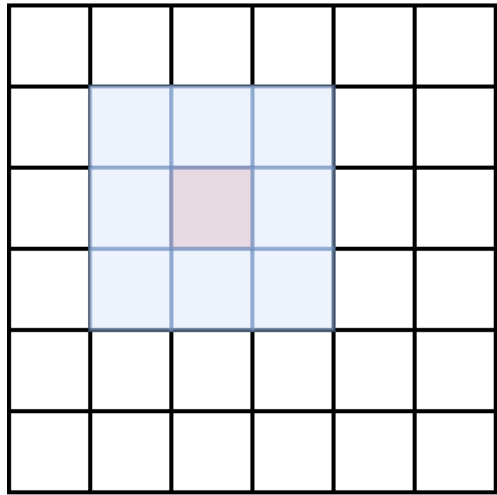
Agregación

Operación sobre  
vecinos (dados por  
matriz de adyacencia)

Es importante notar que acá solo podemos aprender una transformación que no depende del orden/posición del vecino

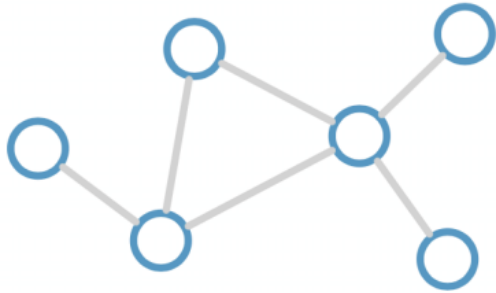


## Convoluciones en grafos (versión simplificada)

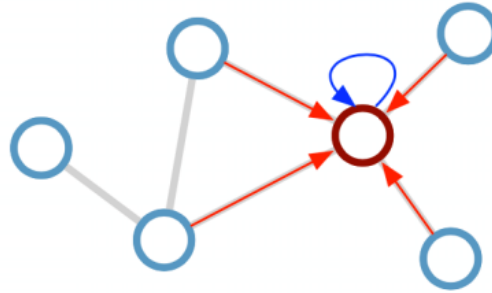


# Convoluciones en grafos (versión paso de mensajes)

Consider this undirected graph:



Calculate update for node in red:



## Desirable properties:

- Weight sharing over all locations
- Invariance to permutations
- Linear complexity  $O(E)$
- Applicable both in transductive and inductive settings

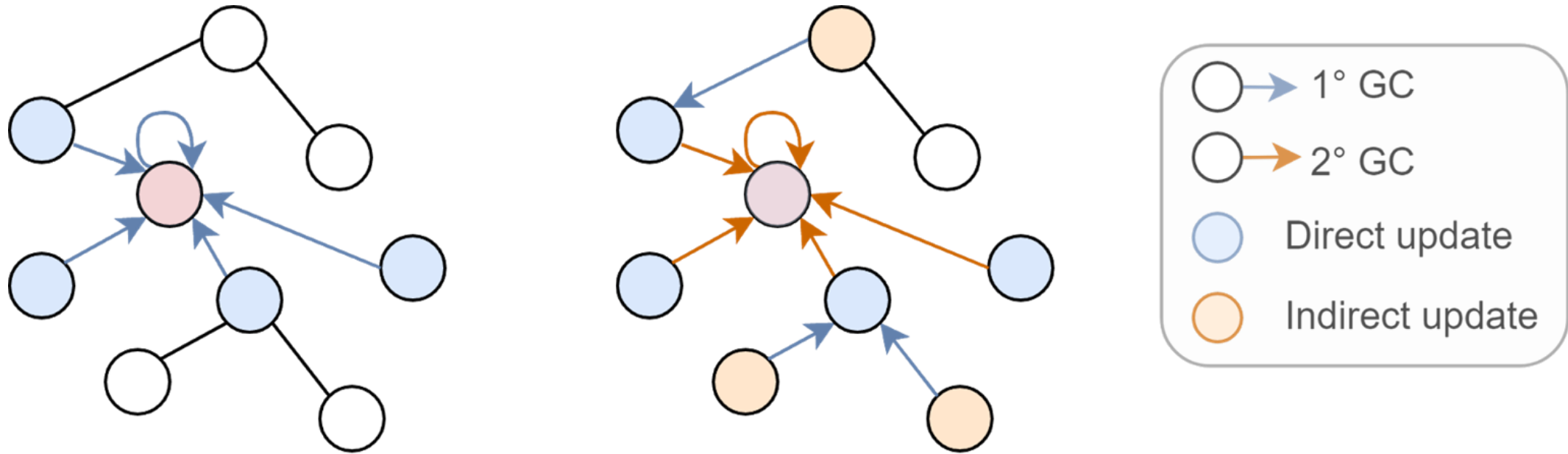
**Update rule:**

$$\mathbf{h}_i^{(l+1)} = \sigma \left( \mathbf{h}_i^{(l)} \mathbf{W}_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{h}_j^{(l)} \mathbf{W}_1^{(l)} \right)$$

**Scalability: subsample messages** [Hamilton et al., NIPS 2017]

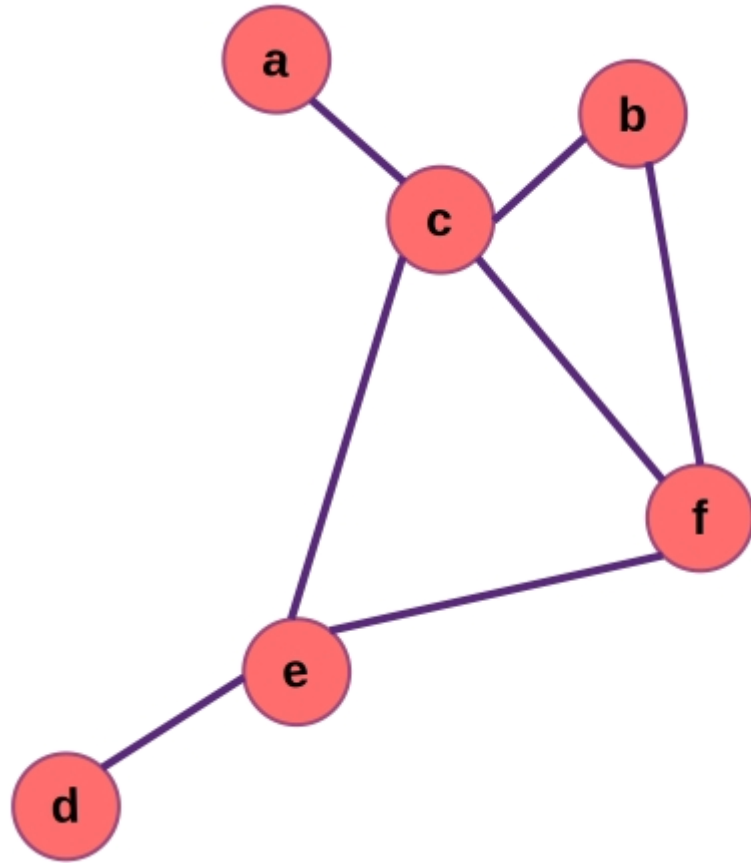
$\mathcal{N}_i$  : neighbor indices       $c_{ij}$  : norm. constant

¿Qué pasa si tenemos/queremos más capas?



Capas sucesivas amplían la “visión” del vecindario de cada nodo

## Convoluciones en grafos (versión matricial)



	a	b	c	d	e	f
a	0	0	0	0	0	0
b	0	0	1	0	0	1
c	1	1	0	0	1	1
d	0	0	0	0	1	0
e	0	0	1	1	0	1
f	0	1	1	0	1	0

## Convoluciones en grafos (versión matricial)

$$\sigma\left(\begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \times \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}\right) = \begin{array}{|c|} \hline Y \\ \hline \end{array}$$

*x* *w*

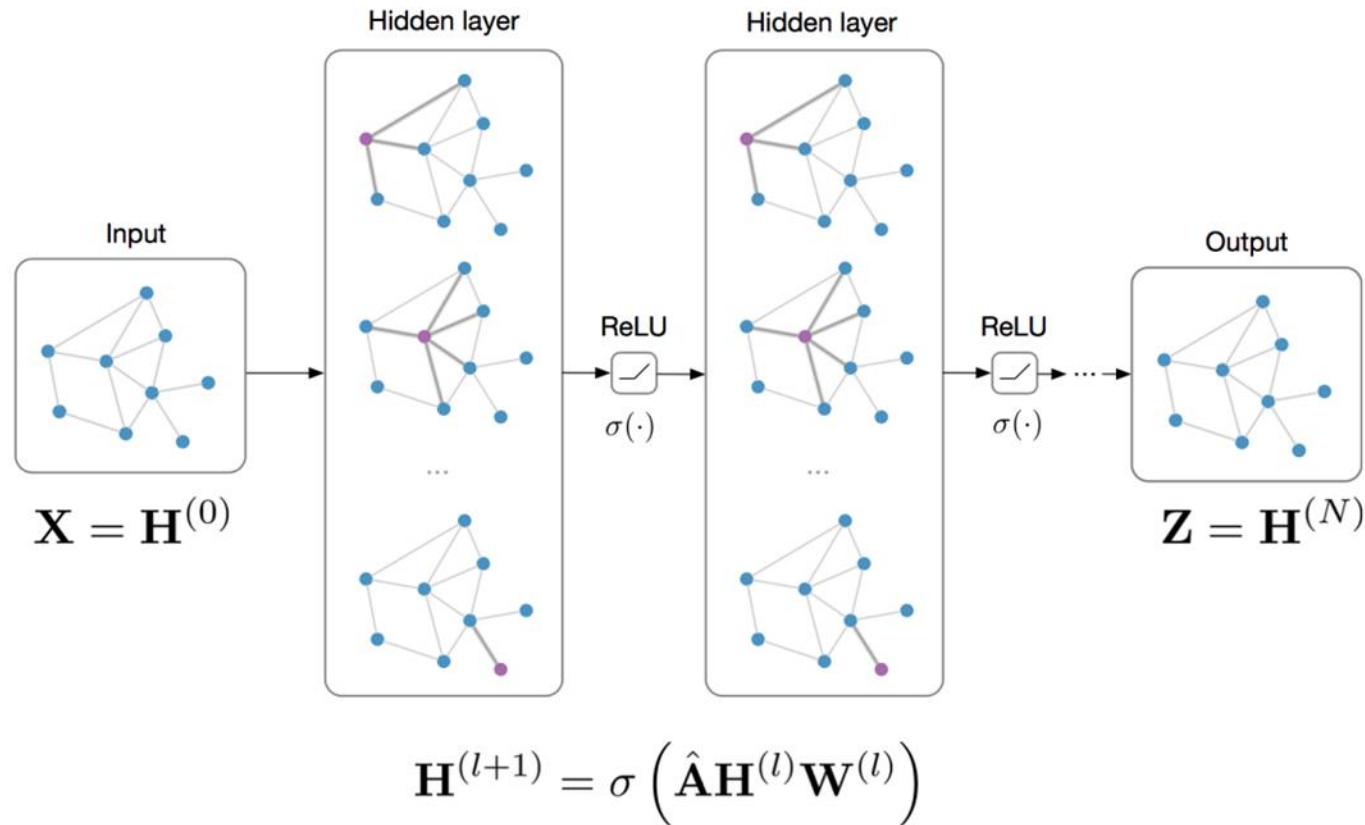
## Convoluciones en grafos (versión matricial)

$$\sigma \left( \begin{array}{c} \text{Matrix A} \\ A \end{array} \times \begin{array}{c} \text{Matrix X} \\ X \end{array} \times \begin{array}{c} \text{Matrix W} \\ W \end{array} \right) = \begin{array}{c} \text{Matrix Y} \\ Y \end{array}$$

The diagram illustrates a matrix-based graph convolution operation. It shows the composition of three matrices:  $A$  (green),  $X$  (red), and  $W$  (purple), which are multiplied together and then passed through an activation function  $\sigma$  to produce the output matrix  $Y$  (purple).

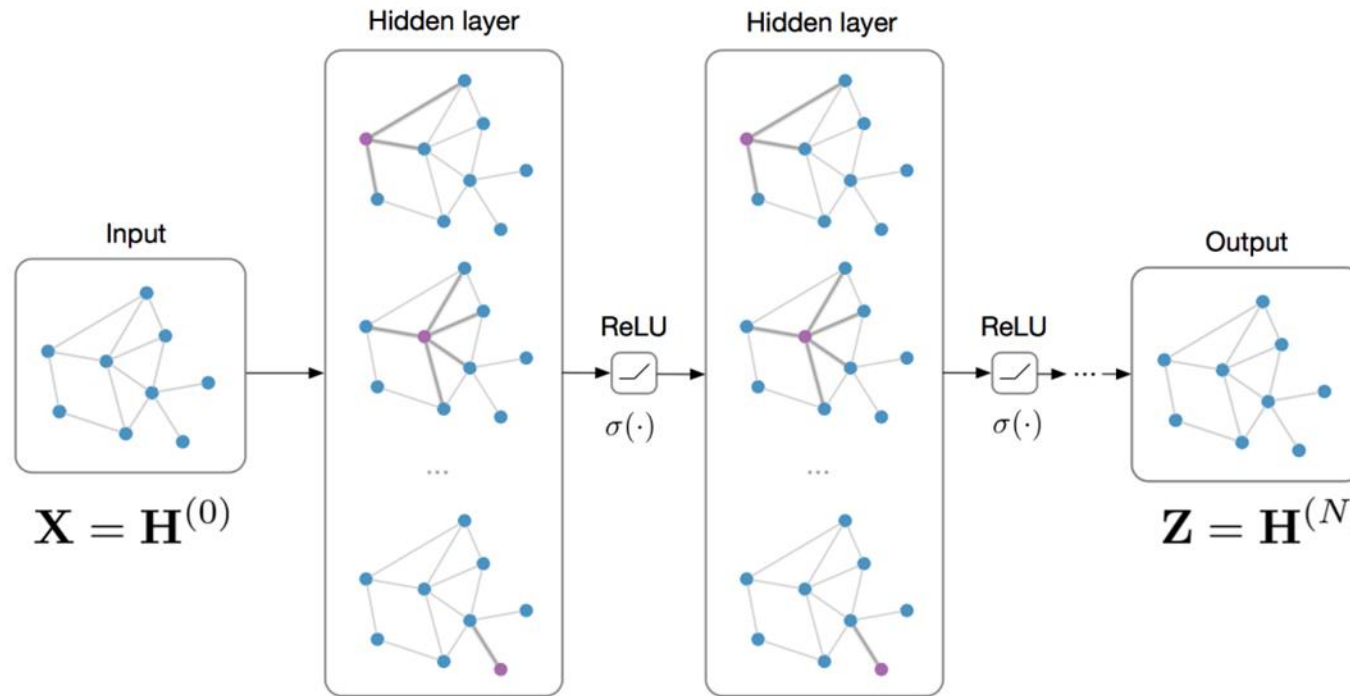
# Convoluciones en grafos (versión matricial)

**Input:** Feature matrix  $\mathbf{X} \in \mathbb{R}^{N \times E}$ , preprocessed adjacency matrix  $\hat{\mathbf{A}}$



# Convoluciones en grafos (versión matricial)

**Input:** Feature matrix  $\mathbf{X} \in \mathbb{R}^{N \times E}$ , preprocessed adjacency matrix  $\hat{\mathbf{A}}$



$$\mathbf{H}^{(l+1)} = \sigma \left( \hat{\mathbf{A}} \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right)$$

**Node classification:**

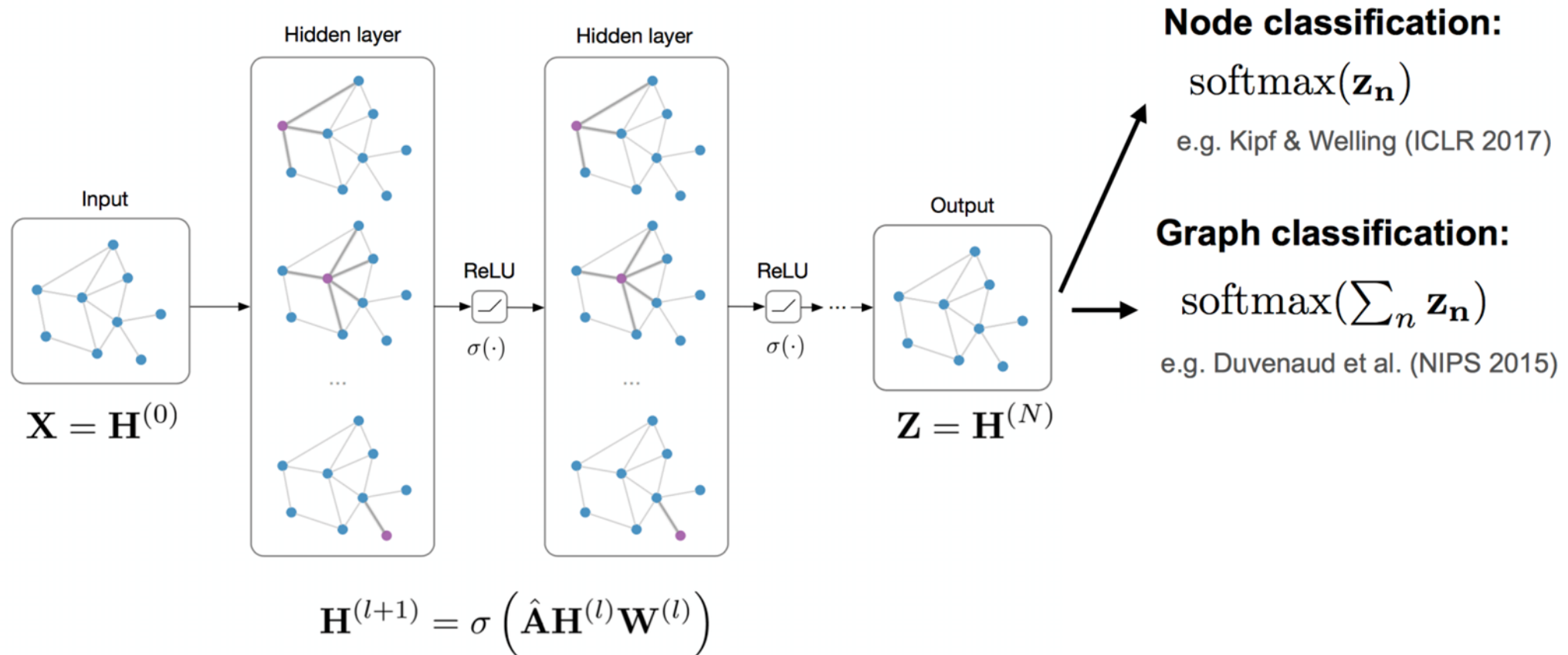
$$\text{softmax}(\mathbf{z}_{\mathbf{n}})$$

e.g. Kipf & Welling (ICLR 2017)



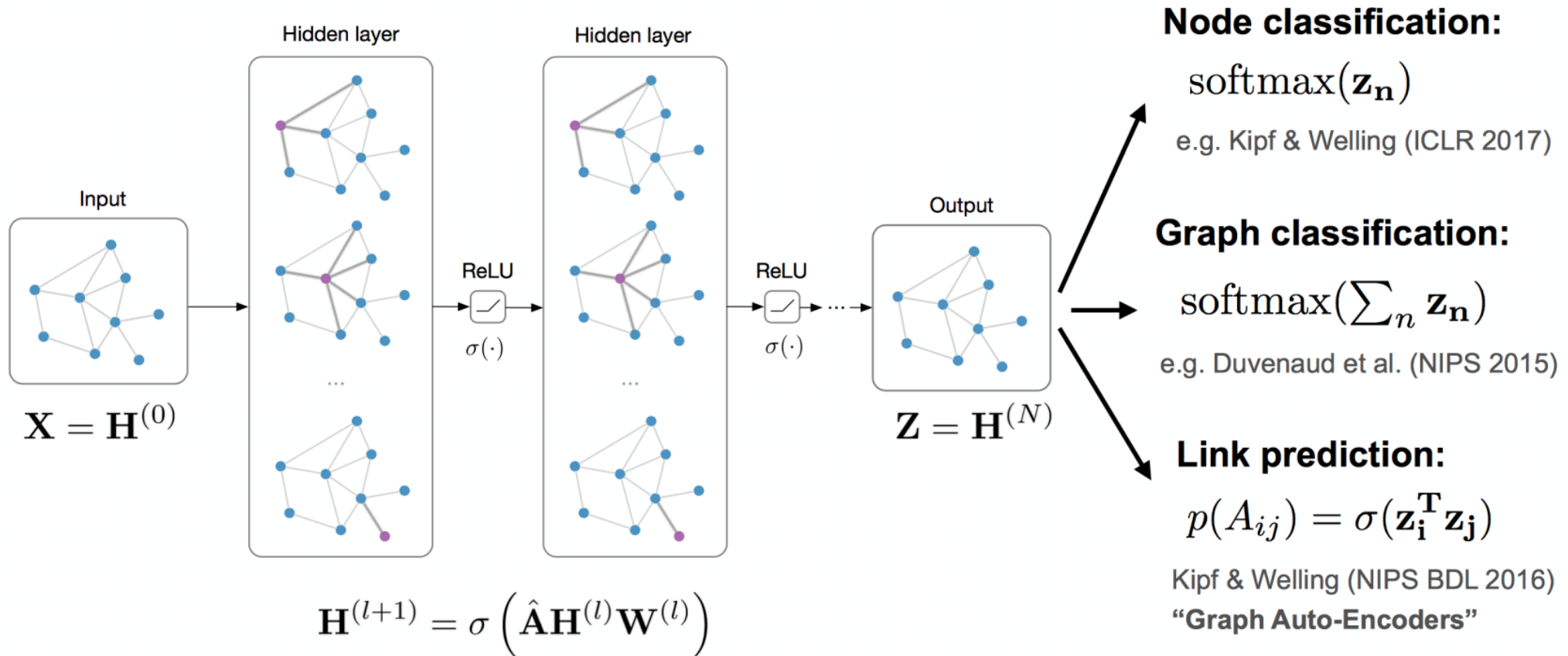
# Convoluciones en grafos (versión matricial)

**Input:** Feature matrix  $\mathbf{X} \in \mathbb{R}^{N \times E}$ , preprocessed adjacency matrix  $\hat{\mathbf{A}}$



# Convoluciones en grafos (versión matricial)

**Input:** Feature matrix  $\mathbf{X} \in \mathbb{R}^{N \times E}$ , preprocessed adjacency matrix  $\hat{\mathbf{A}}$



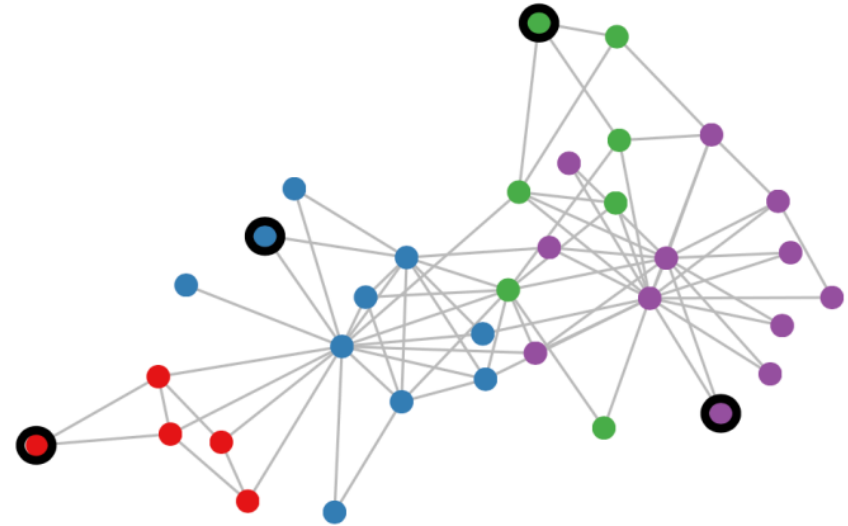
## También podemos hacer clasificación semi-supervisada

### Setting:

Some nodes are labeled (black circle)  
All other nodes are unlabeled

### Task:

Predict node label of unlabeled nodes



Evaluate loss on labeled nodes only:

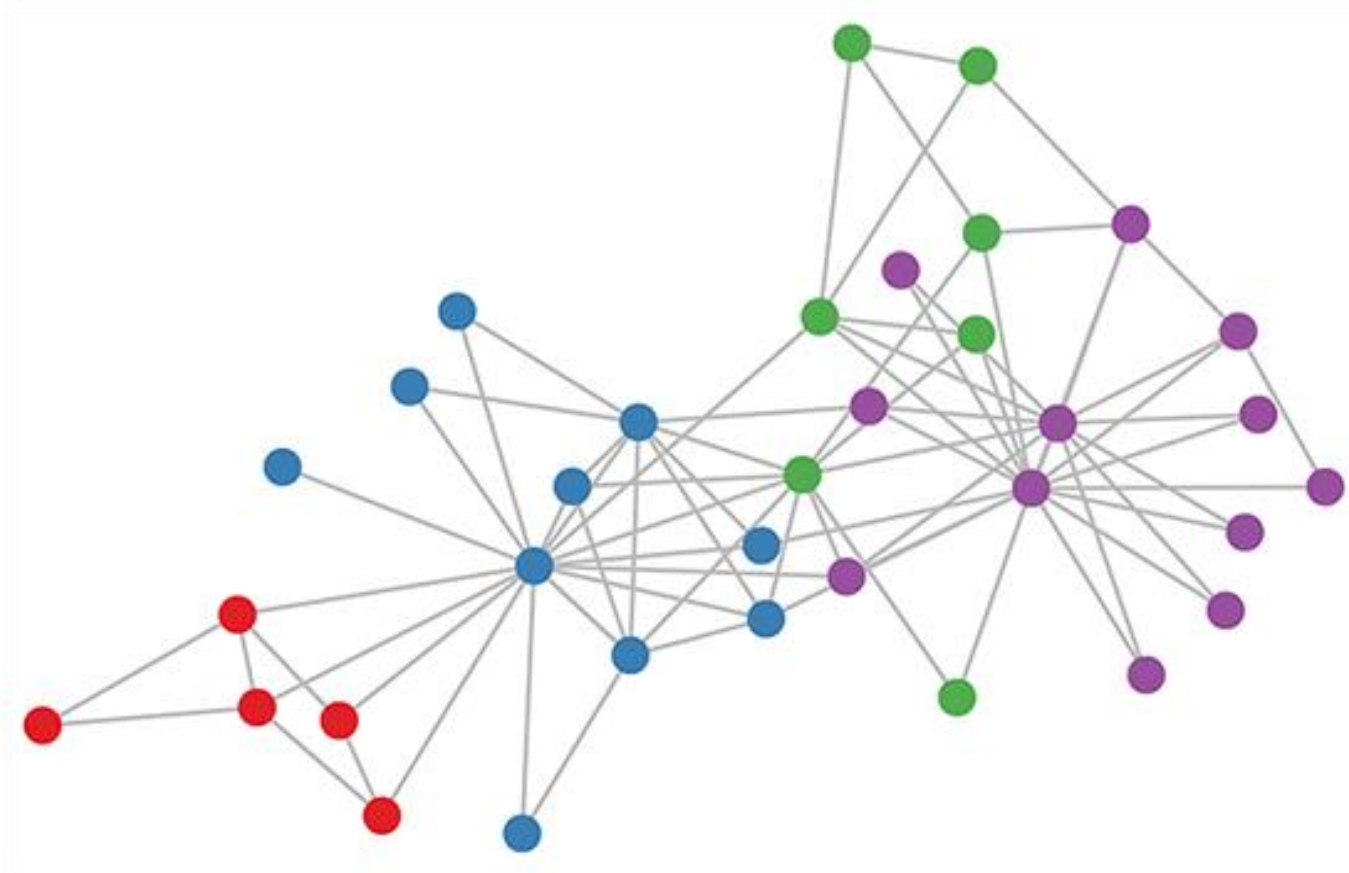
$$\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{f=1}^F Y_{lf} \ln Z_{lf}$$

$\mathcal{Y}_L$  set of labeled node indices

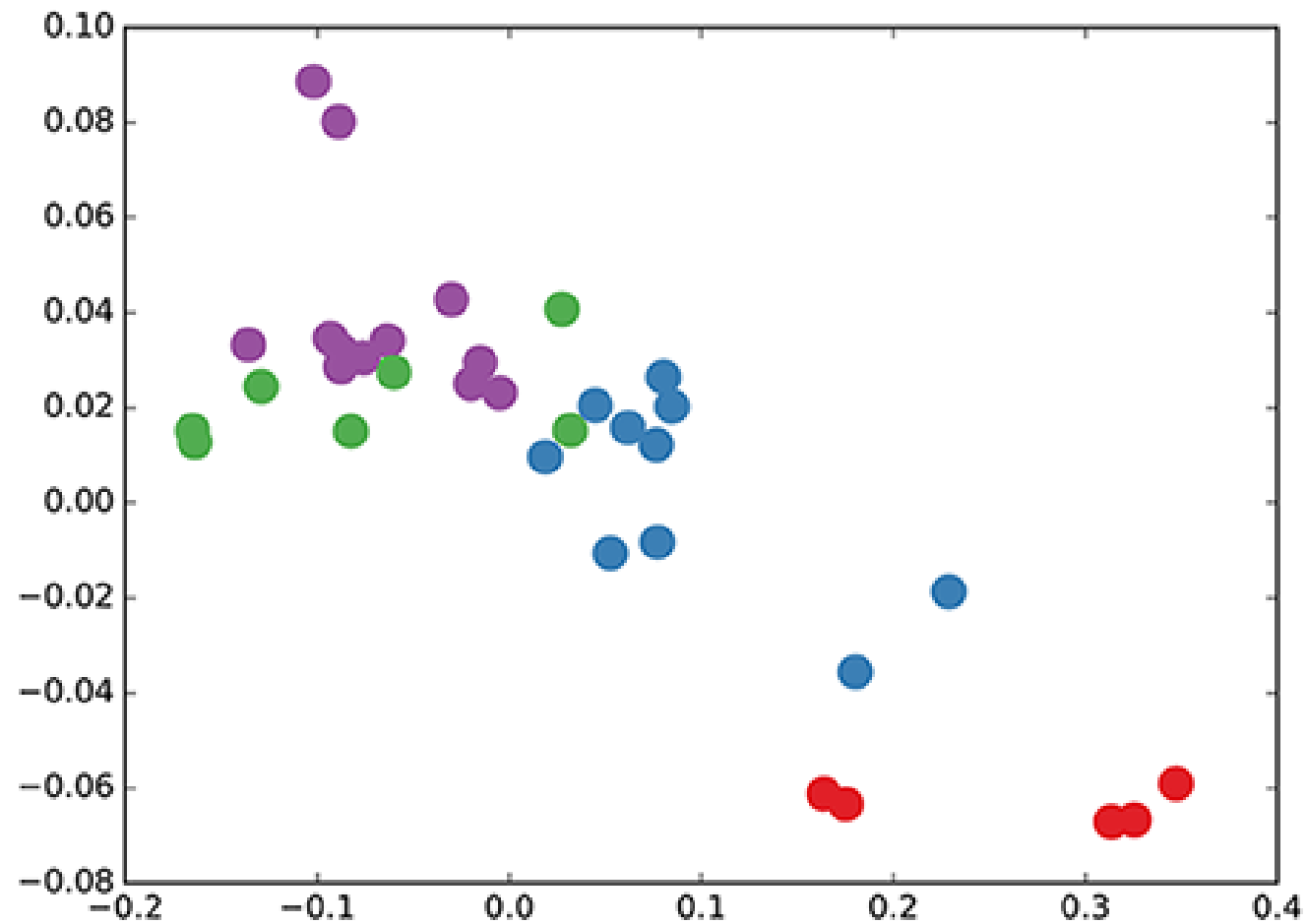
$\mathbf{Y}$  label matrix

$\mathbf{Z}$  GCN output (after softmax)

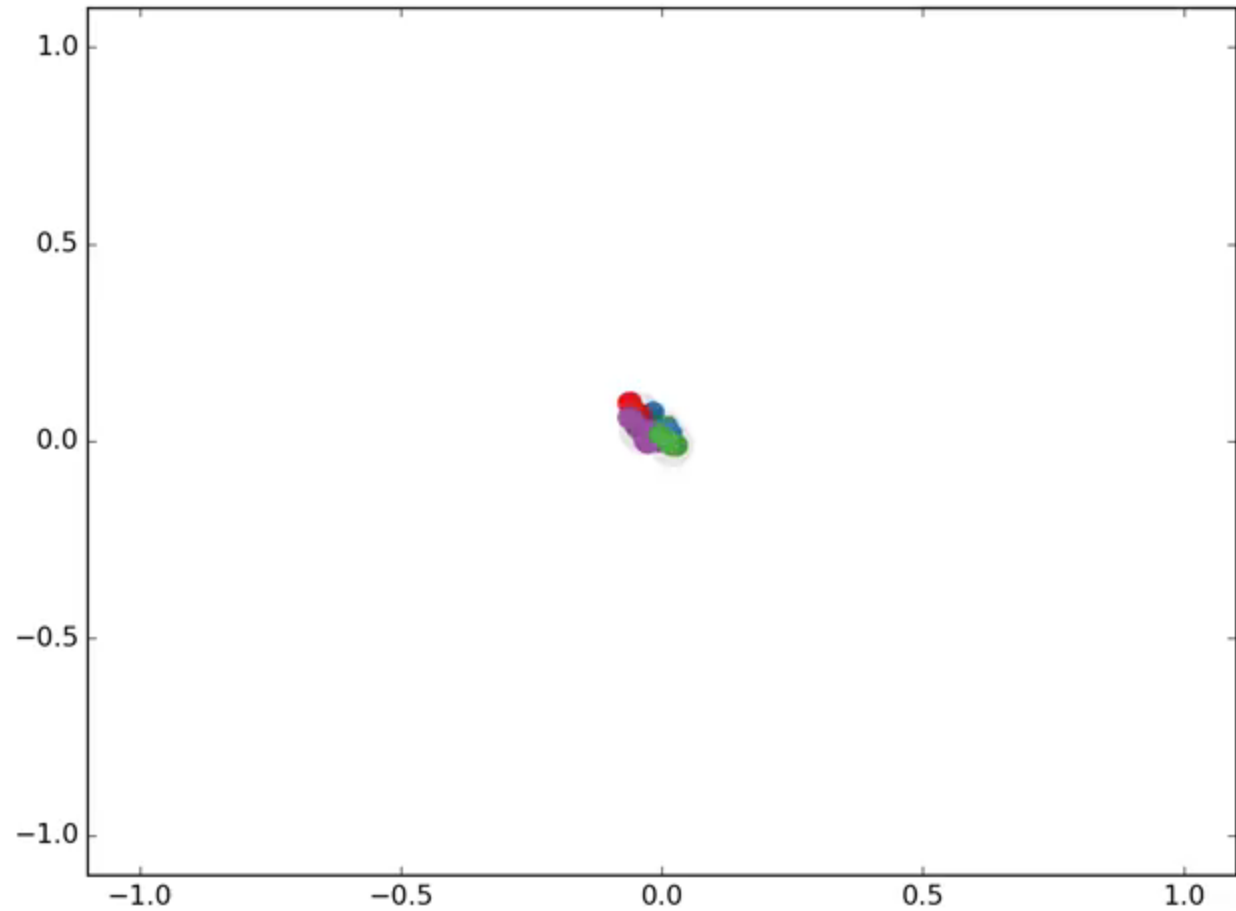
Veamos como funciona esto



Veamos como funciona esto



Veamos como funciona esto



Pontificia Universidad Católica de Chile  
Escuela de Ingeniería  
Departamento de Ingeniería de Transporte y Logística



# Sistemas Urbanos Inteligentes

Redes convolucionales para grafos

Hans Löbel

Dpto. Ingeniería de Transporte y Logística  
Dpto. Ciencia de la Computación