Exploiting the Structure of Two Graphs via Graph Neural Networks

Victor M. Tenorio

King Juan Carlos University - Madrid (Spain)

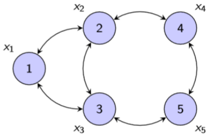
In collaboration with Antonio G. Marques

Universidad Rey Juan Carlos

July 10, 2024

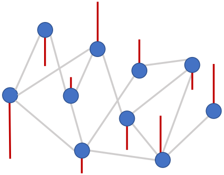
Index

- Each day huge amounts of data are generated
  - ⇒ **Irregular** structure
- Need to find new ways to
  - ⇒ Represent the data
  - ⇒ Learn from it



- Representation of irregular data
  - ⇒ Via more complex structures → **graphs**
- Learning over irregular data
  - ⇒ New machine learning algorithms
- Join both in **graph neural networks (GNNs)**

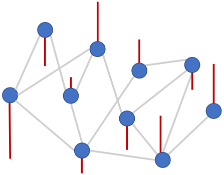
- **This work**: input and output are defined over different graphs



- A **graph** $\mathcal{G}$: N nodes and links connecting them
  - $\Rightarrow \mathcal{G} \equiv (\mathcal{V}, \mathcal{E}, \mathbf{A})$
  - $\Rightarrow \mathcal{V} = \{1, \ldots, N\}$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, $\mathbf{A} \in \mathbb{R}^{N \times N}$
- Define a signal $\mathbf{x} \in \mathbb{R}^N$ on top of the graph
  - $\Rightarrow x_i =$ Signal value at node i

Preliminaries of Graph Signal Processing

Universidad Rey Juan Carlos



- A **graph** $\mathcal{G}$: N nodes and links connecting them
  - $\Rightarrow \mathcal{G} \equiv (\mathcal{V}, \mathcal{E}, \mathbf{A})$
  - $\Rightarrow \mathcal{V} = \{1, \ldots, N\}$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, $\mathbf{A} \in \mathbb{R}^{N \times N}$
- Define a signal $\mathbf{x} \in \mathbb{R}^N$ on top of the graph
  - $\Rightarrow x_i =$ Signal value at node i
- Associated with $\mathcal{G} \rightarrow$ graph-shift operator (GSO) $\mathbf{S} \in \mathbb{R}^{N \times N}$
  - $\Rightarrow S_{ij} \neq 0$ if and only if $i = j$ or $(i, j) \in \mathcal{E}$ (local structure in $\mathcal{G}$)
- **Graph Signal Processing** $\rightarrow$ Exploit structure encoded in $\mathbf{S}$ to process $\mathbf{x}$
- First **linear processing**: graph filters, graph Fourier transform...
- Then **neural nets**: GCNNs, GRNNs, G-Tensor, G-Autoencoders

- Existing NN works dealing with graph signals [Bruna17]
  - $\Rightarrow$ Input graph signal $\mathcal{G}$, output scalar (class)
  - $\Rightarrow$ Input graph signal $\mathcal{G}$, output graph signal $\mathcal{G}$ (embeddings)

- Existing NN works dealing with graph signals [Bruna17]
  - $\Rightarrow$ Input graph signal $\mathcal{G}$, output scalar (class)
  - $\Rightarrow$ Input graph signal $\mathcal{G}$, output graph signal $\mathcal{G}$ (embeddings)
- Here, consider two signals, each defined on a different graph:
  - $\mathcal{G}_X$ with N nodes (signal $\mathbf{x} \in \mathbb{R}^N$), and graph-shift operator $\mathbf{S}_X$
  - $\mathcal{G}_Y$ with M nodes (signal $\mathbf{y} \in \mathbb{R}^M$), and graph-shift operator $\mathbf{S}_Y$

- Existing NN works dealing with graph signals [Bruna17]
  - $\Rightarrow$ Input graph signal $\mathcal{G}$, output scalar (class)
  - $\Rightarrow$ Input graph signal $\mathcal{G}$, output graph signal $\mathcal{G}$ (embeddings)

- Here, consider two signals, each defined on a different graph:
  - $\mathcal{G}_X$ with N nodes (signal $\mathbf{x} \in \mathbb{R}^N$), and graph-shift operator $\mathbf{S}_X$
  - $\mathcal{G}_Y$ with M nodes (signal $\mathbf{y} \in \mathbb{R}^M$), and graph-shift operator $\mathbf{S}_Y$

- **Goal**: Learn the **nonlinear mapping** $f_\Theta : \mathbb{R}^N \to \mathbb{R}^M$

$$\mathbf{y} = f_\Theta(\mathbf{x} \,|\mathcal{G}_X, \mathcal{G}_Y)$$

exploiting $\mathcal{G}_X$ and $\mathcal{G}_Y$ and using a Neural Network (NN) architecture

Motivation, context and goal

Universidad Rey Juan Carlos

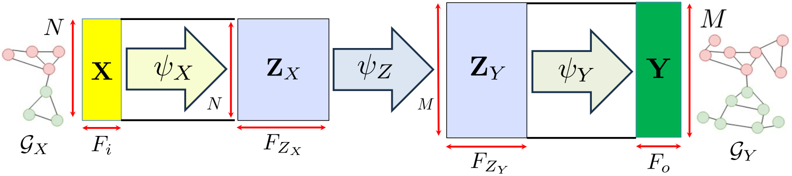
- Existing NN works dealing with graph signals [Bruna17]
  - $\Rightarrow$ Input graph signal $\mathcal{G}$, output scalar (class)
  - $\Rightarrow$ Input graph signal $\mathcal{G}$, output graph signal $\mathcal{G}$ (embeddings)
- Here, consider two signals, each defined on a different graph:
  - $\mathcal{G}_X$ with N nodes (signal $\mathbf{x} \in \mathbb{R}^N$), and graph-shift operator $\mathbf{S}_X$
  - $\mathcal{G}_Y$ with M nodes (signal $\mathbf{y} \in \mathbb{R}^M$), and graph-shift operator $\mathbf{S}_Y$
- **Goal**: Learn the **nonlinear mapping** $f_\Theta : \mathbb{R}^N \to \mathbb{R}^M$

$$\mathbf{y} = f_\Theta(\mathbf{x} \,|\mathcal{G}_X, \mathcal{G}_Y)$$

  exploiting $\mathcal{G}_X$ and $\mathcal{G}_Y$ and using a Neural Network (NN) architecture
- **Key**: Consider latent space $\mathbf{Z}$ to transform between graphs

- The underlying space **Z** implies that

$$\mathbf{x} \text{ on } \mathcal{G}_X \underbrace{\Longrightarrow}_{\psi^X_{\Theta_X}} \mathbf{Z}_X \in \mathbb{R}^{N \times F_{Z_X}}, \quad \mathbf{Z}_X \underbrace{\Longrightarrow}_{\psi^Z_{\Theta_Z}} \mathbf{Z}_Y \in \mathbb{R}^{M \times F_{Z_Y}}, \quad \mathbf{Z}_Y \underbrace{\Longrightarrow}_{\psi^Y_{\Theta_Y}} \mathbf{y} \text{ on } \mathcal{G}_Y$$

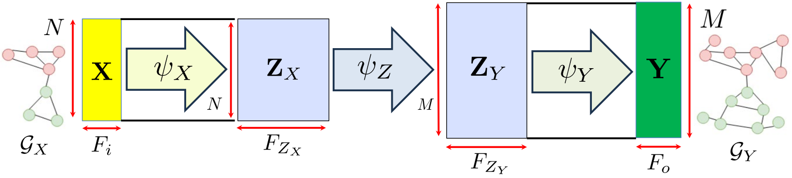


Common underlying space I

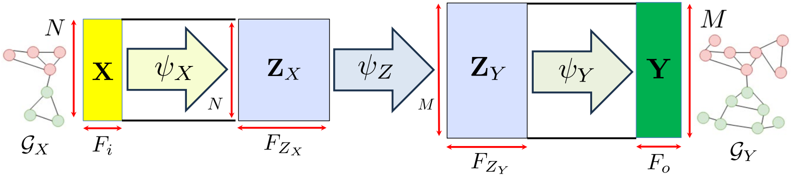
Universidad Rey Juan Carlos

- The underlying space $\mathbf{Z}$ implies that

$$\mathbf{x} \text{ on } \mathcal{G}_X \underbrace{\Longrightarrow}_{\psi^X_{\Theta_X}} \mathbf{Z}_X \in \mathbb{R}^{N \times F_{Z_X}}, \quad \mathbf{Z}_X \underbrace{\Longrightarrow}_{\psi^Z_{\Theta_Z}} \mathbf{Z}_Y \in \mathbb{R}^{M \times F_{Z_Y}}, \quad \mathbf{Z}_Y \underbrace{\Longrightarrow}_{\psi^Y_{\Theta_Y}} \mathbf{y} \text{ on } \mathcal{G}_Y$$



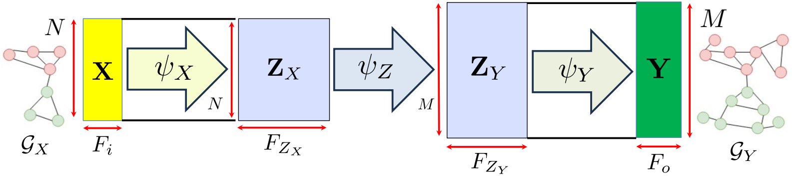
- More precisely
  - $\Rightarrow \psi^X_{\Theta_X}$ standard GNN operating over $\mathcal{G}_X$
  - $\Rightarrow \psi^Y_{\Theta_Y}$ standard GNN operating over $\mathcal{G}_Y$
  - $\Rightarrow \psi^Z_{\Theta_Z}$ transformation between domains: design covered later



- The input-output mapping is then

$$f_{\boldsymbol{\Theta}}(\mathbf{X}|\mathcal{G}_X, \mathcal{G}_Y) = \psi^{Y}_{\boldsymbol{\Theta}_Y} \circ \psi^{Z}_{\boldsymbol{\Theta}_Z} \circ \psi^{X}_{\boldsymbol{\Theta}_X} \quad ; \quad \hat{\mathbf{Y}} = \psi^{Y}_{\boldsymbol{\Theta}_Y}(\psi^{Z}_{\boldsymbol{\Theta}_Z}(\psi^{X}_{\boldsymbol{\Theta}_X}(\mathbf{X}|\mathcal{G}_X))|\mathcal{G}_Y)$$

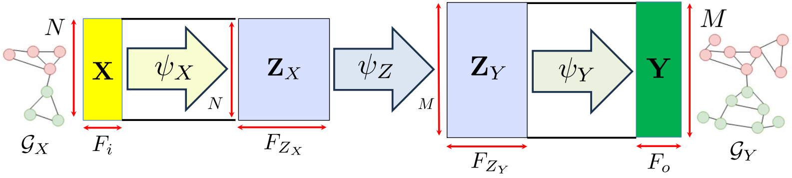
Universidad Rey Juan Carlos



- The input-output mapping is then

$$f_{\mathbf{\Theta}}(\mathbf{X}|\mathcal{G}_X,\mathcal{G}_Y) = \psi^Y_{\mathbf{\Theta}_Y} \circ \psi^Z_{\mathbf{\Theta}_Z} \circ \psi^X_{\mathbf{\Theta}_X} \;\; ; \;\; \hat{\mathbf{Y}} = \psi^Y_{\mathbf{\Theta}_Y}(\psi^Z_{\mathbf{\Theta}_Z}(\psi^X_{\mathbf{\Theta}_X}(\mathbf{X}|\mathcal{G}_X))|\mathcal{G}_Y)$$

- Parameters $\mathbf{\Theta}_X$, $\mathbf{\Theta}_Y$ and (possibly) $\mathbf{\Theta}_Z$
  - $\Rightarrow$ Learned through backpropagation
  - $\Rightarrow$ Assumption: $\psi^Z_{\mathbf{\Theta}_Z}$ is differentiable



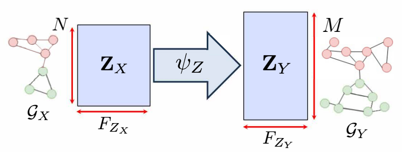
- The input-output mapping is then

$$f_{\boldsymbol{\Theta}}(\mathbf{X}|\mathcal{G}_X, \mathcal{G}_Y) = \psi^Y_{\boldsymbol{\Theta}_Y} \circ \psi^Z_{\boldsymbol{\Theta}_Z} \circ \psi^X_{\boldsymbol{\Theta}_X} \;\; ; \;\; \hat{\mathbf{Y}} = \psi^Y_{\boldsymbol{\Theta}_Y}(\psi^Z_{\boldsymbol{\Theta}_Z}(\psi^X_{\boldsymbol{\Theta}_X}(\mathbf{X}|\mathcal{G}_X))|\mathcal{G}_Y)$$

- Parameters $\boldsymbol{\Theta}_X$, $\boldsymbol{\Theta}_Y$ and (possibly) $\boldsymbol{\Theta}_Z$
  - $\Rightarrow$ Learned through backpropagation
  - $\Rightarrow$ Assumption: $\psi^Z_{\boldsymbol{\Theta}_Z}$ is differentiable
- Key in this approach: design of $\psi^Z_{\boldsymbol{\Theta}_Z}$

Design of $\psi^{Z}_{\Theta_Z}$

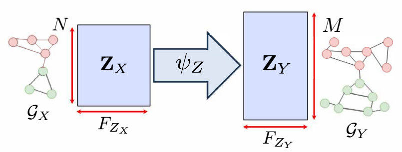
- Several design decisions $\rightarrow$ taxonomy



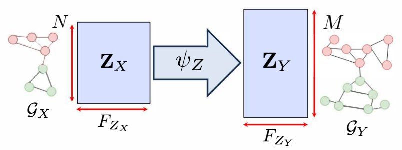
Design of $\psi^{Z}_{\Theta_Z}$

Universidad Rey Juan Carlos

- Several design decisions $\rightarrow$ taxonomy
- Domain specific vs agnostic to the task
  - $\Rightarrow$ Incorporate info about the task

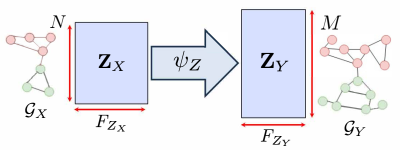


Design of $\psi^{Z}_{\mathbf{\Theta}_Z}$



- Several design decisions $\rightarrow$ taxonomy
- Domain specific vs agnostic to the task
  - $\Rightarrow$ Incorporate info about the task
  - Learnable vs fixed in advance
    - $\Rightarrow$ Propose a parametric function with parameters $\mathbf{\Theta}_Z$
    - $\Rightarrow$ $\mathbf{\Theta}_Z$ learned through gradient descent and backpropagation

Design of $\psi^{Z}_{\mathbf{\Theta}_Z}$



- Several design decisions $\rightarrow$ taxonomy
- Domain specific vs agnostic to the task
  - $\Rightarrow$ Incorporate info about the task
  - Learnable vs fixed in advance
    - $\Rightarrow$ Propose a parametric function with parameters $\mathbf{\Theta}_Z$
    - $\Rightarrow$ $\mathbf{\Theta}_Z$ learned through gradient descent and backpropagation
  - Linear vs more complex transformations
    - $\Rightarrow$ Most general case of linear transformation

$$\text{vec}(\mathbf{Z}_Y) = \mathbf{W}\text{vec}(\mathbf{Z}_X),$$

    - $\Rightarrow$ With (possibly learnable) transformation $\mathbf{W} \in \mathbb{R}^{MF_{Z_Y} \times NF_{Z_X}}$
    - $\Rightarrow$ Huge number of parameters if graphs are large $\rightarrow$ overfitting
    - $\Rightarrow$ Can incorporate structure to the transformation
    - $\Rightarrow$ More complex can include e.g. MLP

- Low Rank $\mathbf{W} = \mathbf{W}_Y \mathbf{W}_X^{\mathsf{T}}$

  $\Rightarrow$ $\mathbf{W}_Y \in \mathbb{R}^{MF_{Z_Y} \times K}$ and $\mathbf{W}_X \in \mathbb{R}^{NF_{Z_X} \times K}$

  $\Rightarrow$ Reduce params. from $MF_{Z_Y} NF_{Z_X}$ to $(MF_{Z_Y} + NF_{Z_X})K$

Linear structure in $\psi^{Z}_{\mathbf{\Theta}_Z}$

Universidad Rey Juan Carlos

- Low Rank $\mathbf{W} = \mathbf{W}_Y \mathbf{W}_X^{\mathrm{T}}$
  - $\Rightarrow \mathbf{W}_Y \in \mathbb{R}^{MF_{Z_Y} \times K}$ and $\mathbf{W}_X \in \mathbb{R}^{NF_{Z_X} \times K}$
  - $\Rightarrow$ Reduce params. from $MF_{Z_Y} NF_{Z_X}$ to $(MF_{Z_Y} + NF_{Z_X})K$
- Kronecker Structure $\mathbf{W} = \mathbf{W}_F^T \otimes \mathbf{W}_N \Rightarrow \mathbf{Z}_Y = \mathbf{W}_N \mathbf{Z}_X \mathbf{W}_F$
  - $\Rightarrow$ Using property $\mathrm{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\mathrm{vec}(\mathbf{B})$

Linear structure in $\psi^{Z}_{\mathbf{\Theta}_Z}$

- Low Rank $\mathbf{W} = \mathbf{W}_Y \mathbf{W}_X^{\mathsf{T}}$
  - $\Rightarrow$ $\mathbf{W}_Y \in \mathbb{R}^{MF_{Z_Y} \times K}$ and $\mathbf{W}_X \in \mathbb{R}^{NF_{Z_X} \times K}$
  - $\Rightarrow$ Reduce params. from $MF_{Z_Y} NF_{Z_X}$ to $(MF_{Z_Y} + NF_{Z_X})K$
- Kronecker Structure $\mathbf{W} = \mathbf{W}_F^T \otimes \mathbf{W}_N \Rightarrow \mathbf{Z}_Y = \mathbf{W}_N \mathbf{Z}_X \mathbf{W}_F$
  - $\Rightarrow$ Using property $\mathrm{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\mathrm{vec}(\mathbf{B})$
  - $\Rightarrow$ $\mathbf{W}_N \in \mathbb{R}^{M \times N}$ combines information across nodes
  - $\Rightarrow$ $\mathbf{W}_F \in \mathbb{R}^{F_{Z_X} \times F_{Z_Y}}$ combines information across features
  - $\Rightarrow$ Both $\mathbf{W}_N$ and $\mathbf{W}_F$ can be fixed or learned
  - $\Rightarrow$ If both are learned, alternating fashion

Linear structure in $\psi^{Z}_{\Theta_Z}$

Universidad Rey Juan Carlos

- Low Rank $\mathbf{W} = \mathbf{W}_Y \mathbf{W}_X^{\mathsf{T}}$
  - $\Rightarrow$ $\mathbf{W}_Y \in \mathbb{R}^{MF_{Z_Y} \times K}$ and $\mathbf{W}_X \in \mathbb{R}^{NF_{Z_X} \times K}$
  - $\Rightarrow$ Reduce params. from $MF_{Z_Y} NF_{Z_X}$ to $(MF_{Z_Y} + NF_{Z_X})K$
- Kronecker Structure $\mathbf{W} = \mathbf{W}_F^T \otimes \mathbf{W}_N \Rightarrow \mathbf{Z}_Y = \mathbf{W}_N \mathbf{Z}_X \mathbf{W}_F$
  - $\Rightarrow$ Using property $\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B})$
  - $\Rightarrow$ $\mathbf{W}_N \in \mathbb{R}^{M \times N}$ combines information across nodes
  - $\Rightarrow$ $\mathbf{W}_F \in \mathbb{R}^{F_{Z_X} \times F_{Z_Y}}$ combines information across features
  - $\Rightarrow$ Both $\mathbf{W}_N$ and $\mathbf{W}_F$ can be fixed or learned
  - $\Rightarrow$ If both are learned, alternating fashion
- Permutation matrix
  - $\Rightarrow$ Cells in $\mathbf{Z}_X$ are rearranged to form $\mathbf{Z}_Y$

Examples of domain agnostic $\psi^{Z}_{\Theta_Z}$

- Provide two examples that are
  - $\Rightarrow$ Easy to implement
  - $\Rightarrow$ Agnostic to the underlying task $\rightarrow$ applicable in any situation

Examples of domain agnostic $\psi^{Z}_{\Theta_Z}$

- Provide two examples that are
  - $\Rightarrow$ Easy to implement
  - $\Rightarrow$ Agnostic to the underlying task $\rightarrow$ applicable in any situation

- $\mathbf{Z}_Y = \mathbf{Z}_X^{\top}$
  - $\Rightarrow$ Set $F_{Z_X} = M$ in $\psi^{X}_{\Theta_X}$, and $F_{Z_Y} = N$ in $\psi^{Y}_{\Theta_Y}$
  - $\Rightarrow$ Example of $\mathbf{W}$ as a permutation matrix
  - $\Rightarrow$ Non-learnable
  - $\Rightarrow$ Equivalence features in node and graph signal

Examples of domain agnostic $\psi^{Z}_{\Theta_Z}$

- Provide two examples that are
  - $\Rightarrow$ Easy to implement
  - $\Rightarrow$ Agnostic to the underlying task $\rightarrow$ applicable in any situation
- $\mathbf{Z}_Y = \mathbf{Z}_X^\top$
  - $\Rightarrow$ Set $F_{Z_X} = M$ in $\psi^{X}_{\Theta_X}$, and $F_{Z_Y} = N$ in $\psi^{Y}_{\Theta_Y}$
  - $\Rightarrow$ Example of $\mathbf{W}$ as a permutation matrix
  - $\Rightarrow$ Non-learnable
  - $\Rightarrow$ Equivalence features in node and graph signal
- $\mathbf{Z}_Y = \mathbf{W}_N \mathbf{Z}_X$
  - $\Rightarrow$ Simple Kronecker structure with $\mathbf{W}_F = \mathbf{I}$
  - $\Rightarrow$ $F_{Z_X} = F_{Z_Y}$
  - $\Rightarrow$ $\mathbf{W}_N$ learned through backpropagation

- Standard GNNs are **permutation equivariant**

$$\psi(\mathbf{PX}, \mathbf{PSP}^\mathsf{T}) = \mathbf{P}\psi(\mathbf{X}, \mathbf{S})$$

  $\Rightarrow$ For any permutation matrix $\mathbf{P} \in \mathbb{R}^{N \times N}$
- Can we say the same about IOGNN?

  $\Rightarrow$ Input and output are not defined on the same space

  $\Rightarrow$ Cannot apply same permutation to input and output

Permutation equivariance of IOGNN

- Standard GNNs are **permutation equivariant**

$$\psi(\mathbf{PX}, \mathbf{PSP}^\top) = \mathbf{P}\psi(\mathbf{X}, \mathbf{S})$$

  $\Rightarrow$ For any permutation matrix $\mathbf{P} \in \mathbb{R}^{N \times N}$

- Can we say the same about IOGNN?

  $\Rightarrow$ Input and output are not defined on the same space

  $\Rightarrow$ Cannot apply same permutation to input and output

- We can instead say

$$f_{\mathbf{\Theta}}(\mathbf{P}_X\mathbf{X}|\mathbf{P}_X\mathbf{S}_X\mathbf{P}_X^\top, \mathbf{P}_Y\mathbf{S}_Y\mathbf{P}_Y^\top) = \mathbf{P}_Y f_{\mathbf{\Theta}}(\mathbf{X}|\mathbf{S}_X, \mathbf{S}_Y)$$

  $\Rightarrow$ Under the assumption that $\psi_{\mathbf{\Theta}_Z}^Z$ fulfills

$$\psi_{\mathbf{\Theta}_Z}^Z(\mathbf{P}_X\mathbf{Z}) = \mathbf{P}_Y\psi_{\mathbf{\Theta}_Z}^Z(\mathbf{Z})$$

- Standard GNNs are **permutation equivariant**

$$\psi(\mathbf{PX}, \mathbf{PSP}^\top) = \mathbf{P}\psi(\mathbf{X}, \mathbf{S})$$

  $\Rightarrow$ For any permutation matrix $\mathbf{P} \in \mathbb{R}^{N \times N}$
- Can we say the same about IOGNN?
  - $\Rightarrow$ Input and output are not defined on the same space
  - $\Rightarrow$ Cannot apply same permutation to input and output
- We can instead say

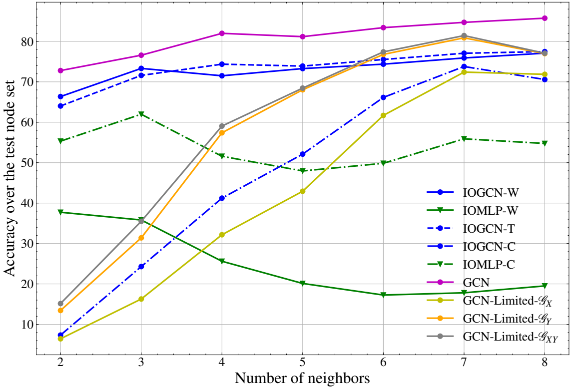
$$f_{\mathbf{\Theta}}(\mathbf{P}_X\mathbf{X}|\mathbf{P}_X\mathbf{S}_X\mathbf{P}_X^\top, \mathbf{P}_Y\mathbf{S}_Y\mathbf{P}_Y^\top) = \mathbf{P}_Y f_{\mathbf{\Theta}}(\mathbf{X}|\mathbf{S}_X, \mathbf{S}_Y)$$

  $\Rightarrow$ Under the assumption that $\psi^Z_{\mathbf{\Theta}_Z}$ fulfills

$$\psi^Z_{\mathbf{\Theta}_Z}(\mathbf{P}_X\mathbf{Z}) = \mathbf{P}_Y\psi^Z_{\mathbf{\Theta}_Z}(\mathbf{Z})$$

- As an example, consider the transformation $\mathbf{Z}_Y = \mathbf{W}_N\mathbf{Z}_X$
  - $\Rightarrow$ Rearrange weight matrix as $\mathbf{W}'_N = \mathbf{P}_Y\mathbf{W}_N\mathbf{P}_X^\top$

- From a graph $\mathcal{G}$ we sample two subgraphs $\mathcal{G}_X$ and $\mathcal{G}_Y$
  - $\Rightarrow$ $\mathcal{G}$ is the Cora graph
  - $\Rightarrow$ Mapping from node features in $\mathcal{G}_X$ to labels in $\mathcal{G}_Y$

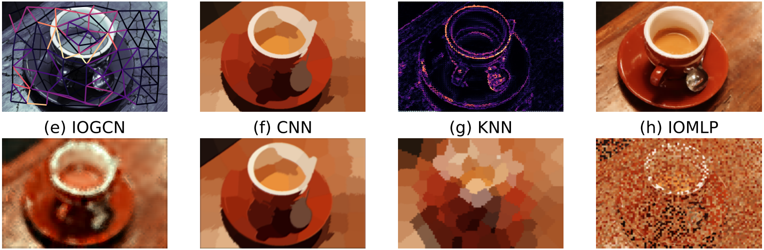


Numerical Results - Image Interpolation

- Framework suited for the interpolation task
  - $\Rightarrow$ $\mathcal{G}_X$ is a coarse/subgraph of $\mathcal{G}_Y$
  - $\Rightarrow$ Interpolation from signal in coarse $\mathcal{G}_X$ to fine $\mathcal{G}_Y$
- Image interpolation
  - $\Rightarrow$ Superpixels + Region Adjacency Graph

Universidad Rey Juan Carlos

- Framework suited for the interpolation task
  - $\Rightarrow$ $\mathcal{G}_X$ is a coarse/subgraph of $\mathcal{G}_Y$
  - $\Rightarrow$ Interpolation from signal in coarse $\mathcal{G}_X$ to fine $\mathcal{G}_Y$
- Image interpolation
  - $\Rightarrow$ Superpixels + Region Adjacency Graph

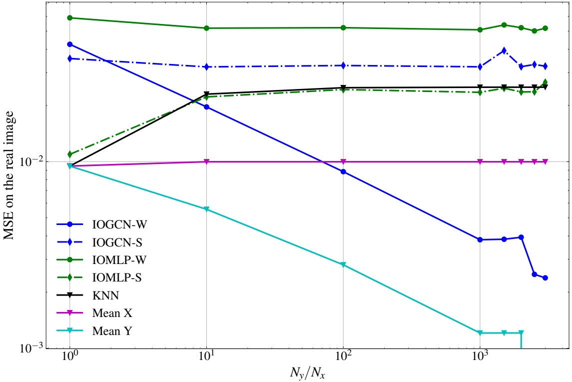


(a) RAG $\mathcal{G}_X$ (b) Mean X (c) RAG $\mathcal{G}_Y$ (d) Target/Mean Y

(e) IOGCN (f) CNN (g) KNN (h) IOMLP

Universidad Rey Juan Carlos

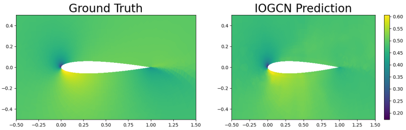
- Framework suited for the interpolation task
  - $\Rightarrow$ $\mathcal{G}_X$ is a coarse/subgraph of $\mathcal{G}_Y$
  - $\Rightarrow$ Interpolation from signal in coarse $\mathcal{G}_X$ to fine $\mathcal{G}_Y$
- Image interpolation
  - $\Rightarrow$ Superpixels + Region Adjacency Graph



Universidad Rey Juan Carlos

- Interpolation in the field of CFD
  - ⇒ Solved via Navier-Stokes PDE on meshes
  - ⇒ Fine meshes are computationally costly
- Fully supervised setting

	Interpolation	Generalization
CFD-GCN*	1,8e-02	5,4e-02
GCN*	1,4e-02	9,5e-02
IOGCN-W	**6,7e-03**	4,1e-02
IOGCN-S	1,7e-02	**3,7e-02**
IOGAT-W	8,7e-03	6,6e-02
IOGAT-S	8,3e-03	6,2e-02
IOMLP-W	8,4e-03	4,0e-02
GAT	1,1e-02	1,1e-01



- Canonical Correlation Analysis (CCA)
  - ⇒ Given two views of data $\mathbf{X} \in \mathbb{R}^{N \times F_X}$ and $\mathbf{Y} \in \mathbb{R}^{N \times F_Y}$
  - ⇒ Compute transformations $\mathbf{U} \in \mathbb{R}^{F_X \times F_Z}$ and $\mathbf{V} \in \mathbb{R}^{F_Y \times F_Z}$
  - ⇒ Seek maximal correlation in transformed space

$$\max_{\mathbf{U},\mathbf{V}} \operatorname{tr}(\mathbf{U}^\top \mathbf{\Sigma}_{XY} \mathbf{V})$$
$$\text{s. to: } \mathbf{U}^\top \mathbf{\Sigma}_{XX} \mathbf{U} = \mathbf{V}^\top \mathbf{\Sigma}_{YY} \mathbf{V} = \mathbf{I},$$

- Canonical Correlation Analysis (CCA)
  - ⇒ Given two views of data $\mathbf{X} \in \mathbb{R}^{N \times F_X}$ and $\mathbf{Y} \in \mathbb{R}^{N \times F_Y}$
  - ⇒ Compute transformations $\mathbf{U} \in \mathbb{R}^{F_X \times F_Z}$ and $\mathbf{V} \in \mathbb{R}^{F_Y \times F_Z}$
  - ⇒ Seek maximal correlation in transformed space

$$\max_{\mathbf{U},\mathbf{V}} \operatorname{tr}(\mathbf{U}^\top \boldsymbol{\Sigma}_{XY} \mathbf{V})$$
$$\text{s. to: } \mathbf{U}^\top \boldsymbol{\Sigma}_{XX} \mathbf{U} = \mathbf{V}^\top \boldsymbol{\Sigma}_{YY} \mathbf{V} = \mathbf{I},$$

- Deep setting: Deep CCA

$$\max_{\boldsymbol{\Theta}_X, \boldsymbol{\Theta}_Y} \operatorname{tr}\left(f_{\boldsymbol{\Theta}_X}(\mathbf{X})^\top f_{\boldsymbol{\Theta}_Y}(\mathbf{Y})\right)$$
$$\text{s. to: } f_{\boldsymbol{\Theta}_X}(\mathbf{X})^\top f_{\boldsymbol{\Theta}_X}(\mathbf{X}) = f_{\boldsymbol{\Theta}_Y}(\mathbf{Y})^\top f_{\boldsymbol{\Theta}_Y}(\mathbf{Y}) = \mathbf{I}$$

- Canonical Correlation Analysis (CCA)
  - ⇒ Given two views of data $\mathbf{X} \in \mathbb{R}^{N \times F_X}$ and $\mathbf{Y} \in \mathbb{R}^{N \times F_Y}$
  - ⇒ Compute transformations $\mathbf{U} \in \mathbb{R}^{F_X \times F_Z}$ and $\mathbf{V} \in \mathbb{R}^{F_Y \times F_Z}$
  - ⇒ Seek maximal correlation in transformed space

$$\max_{\mathbf{U},\mathbf{V}} \operatorname{tr}(\mathbf{U}^\top \boldsymbol{\Sigma}_{XY} \mathbf{V})$$
$$\text{s. to: } \mathbf{U}^\top \boldsymbol{\Sigma}_{XX} \mathbf{U} = \mathbf{V}^\top \boldsymbol{\Sigma}_{YY} \mathbf{V} = \mathbf{I},$$

- Deep setting: Deep CCA

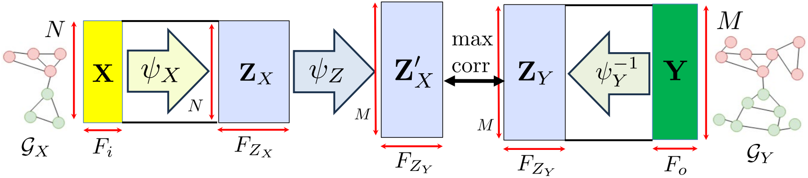
$$\max_{\boldsymbol{\Theta}_X,\boldsymbol{\Theta}_Y} \operatorname{tr}\left(f_{\boldsymbol{\Theta}_X}(\mathbf{X})^\top f_{\boldsymbol{\Theta}_Y}(\mathbf{Y})\right)$$
$$\text{s. to: } f_{\boldsymbol{\Theta}_X}(\mathbf{X})^\top f_{\boldsymbol{\Theta}_X}(\mathbf{X}) = f_{\boldsymbol{\Theta}_Y}(\mathbf{Y})^\top f_{\boldsymbol{\Theta}_Y}(\mathbf{Y}) = \mathbf{I}$$

  - ⇒ Slight different reformulation (CCA-SSG)

$$\min_{\boldsymbol{\Theta}_X,\boldsymbol{\Theta}_Y} \|f_{\boldsymbol{\Theta}_X}(\mathbf{X}) - f_{\boldsymbol{\Theta}_Y}(\mathbf{Y})\|_F^2 + \lambda\left(\mathcal{L}_{SDL}(f_{\boldsymbol{\Theta}_X}(\mathbf{X})) + \mathcal{L}_{SDL}(f_{\boldsymbol{\Theta}_Y}(\mathbf{Y}))\right)$$

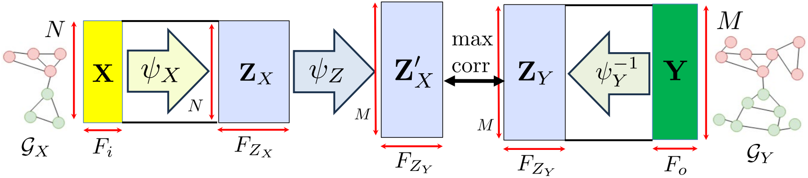
Universidad Rey Juan Carlos

- We can apply our architecture to the CCA setting
  - ⇒ Now we know both **X** and **Y**
  - ⇒ Goal: find alternative representations $\mathbf{Z}_X$ and $\mathbf{Z}_Y$



Connections with CCA and SSL

- We can apply our architecture to the CCA setting
  - $\Rightarrow$ Now we know both $\mathbf{X}$ and $\mathbf{Y}$
  - $\Rightarrow$ Goal: find alternative representations $\mathbf{Z}_X$ and $\mathbf{Z}_Y$



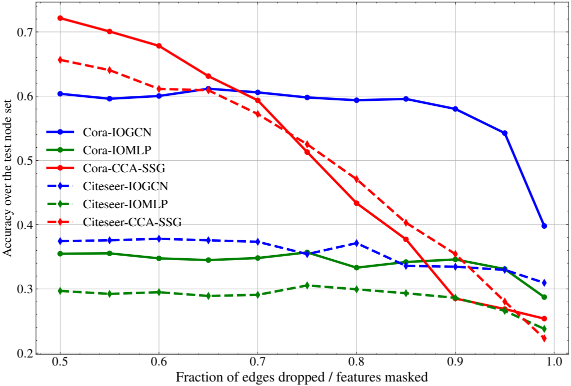
- Aim to solve

$$\max_{\boldsymbol{\Theta}_X,\boldsymbol{\Theta}_Z,\boldsymbol{\Theta}_Y} \ \mathrm{tr}(\psi^Z_{\boldsymbol{\Theta}_Z}(\psi^X_{\boldsymbol{\Theta}_X}(\mathbf{X}|\mathcal{G}_X))^\mathsf{T}\psi^{Y\,-1}_{\boldsymbol{\Theta}_Y}(\mathbf{Y}|\mathcal{G}_Y))$$

$$\text{s. to: } \psi^Z_{\boldsymbol{\Theta}_Z}(\psi^X_{\boldsymbol{\Theta}_X}(\mathbf{X}|\mathcal{G}_X))^\mathsf{T}\psi^Z_{\boldsymbol{\Theta}_Z}(\psi^X_{\boldsymbol{\Theta}_X}(\mathbf{X}|\mathcal{G}_X)) = \psi^{Y\,-1}_{\boldsymbol{\Theta}_Y}(\mathbf{Y}|\mathcal{G}_Y)^\mathsf{T}\psi^{Y\,-1}_{\boldsymbol{\Theta}_Y}(\mathbf{Y}|\mathcal{G}_Y) = \mathbf{I}$$

Numerical Results - Self-Supervised Learning

- Problem setting
  - ⇒ $\mathcal{G}$ is a common graph with two views
  - ⇒ $\mathcal{G}_X$ edges dropped, features masked
  - ⇒ $\mathcal{G}_Y$ subgraph with perfect information
- Perform node classification via transformed views
  - ⇒ SSL setting



Closing remarks

Universidad Rey Juan Carlos

- Novel NN architecture to learn mapping from $(\mathbf{x}, \mathcal{G}_X)$ to $(\mathbf{y}, \mathcal{G}_Y)$
- **Key idea**: latent common space and two graph-aware NN
  - $\Rightarrow$ Step 1) graph-aware NN from $(\mathbf{x}, \mathcal{G}_X)$ to latent space $\mathbf{Z}_X$
  - $\Rightarrow$ Step 2) transformation between $\mathbf{Z}_X$ in $\mathcal{G}_X$ to $\mathbf{Z}_Y$ in $\mathcal{G}_Y$
  - $\Rightarrow$ Step 3) graph-aware NN from latent space $\mathbf{Z}_Y$ to $(\mathbf{y}, \mathcal{G}_Y)$
  - $\Rightarrow$ Parameters jointly learned (backpropag using input-output pairs)

Closing remarks

Universidad Rey Juan Carlos

- Novel NN architecture to learn mapping from $(\mathbf{x}, \mathcal{G}_X)$ to $(\mathbf{y}, \mathcal{G}_Y)$
- **Key idea**: latent common space and two graph-aware NN
  - $\Rightarrow$ Step 1) graph-aware NN from $(\mathbf{x}, \mathcal{G}_X)$ to latent space $\mathbf{Z}_X$
  - $\Rightarrow$ Step 2) transformation between $\mathbf{Z}_X$ in $\mathcal{G}_X$ to $\mathbf{Z}_Y$ in $\mathcal{G}_Y$
  - $\Rightarrow$ Step 3) graph-aware NN from latent space $\mathbf{Z}_Y$ to $(\mathbf{y}, \mathcal{G}_Y)$
  - $\Rightarrow$ Parameters jointly learned (backpropag using input-output pairs)
- Taxonomy of functions for transformation ψ_Z
  - $\Rightarrow$ Several design decisions
  - $\Rightarrow$ Flexible design to accommodate different scenarios

Closing remarks

- Novel NN architecture to learn mapping from $(\mathbf{x}, \mathcal{G}_X)$ to $(\mathbf{y}, \mathcal{G}_Y)$
- **Key idea**: latent common space and two graph-aware NN
  - $\Rightarrow$ Step 1) graph-aware NN from $(\mathbf{x}, \mathcal{G}_X)$ to latent space $\mathbf{Z}_X$
  - $\Rightarrow$ Step 2) transformation between $\mathbf{Z}_X$ in $\mathcal{G}_X$ to $\mathbf{Z}_Y$ in $\mathcal{G}_Y$
  - $\Rightarrow$ Step 3) graph-aware NN from latent space $\mathbf{Z}_Y$ to $(\mathbf{y}, \mathcal{G}_Y)$
  - $\Rightarrow$ Parameters jointly learned (backpropag using input-output pairs)
- Taxonomy of functions for transformation ψ_Z
  - $\Rightarrow$ Several design decisions
  - $\Rightarrow$ Flexible design to accommodate different scenarios
- Analogies with CCA and SSL
  - $\Rightarrow$ Able to learn alternative informative representations
  - $\Rightarrow$ Used for downstream tasks

Thank you

Questions at: **victor.tenorio@urjc.es**