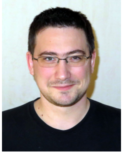
Any2Graph: End-To-End Supervised Graph Prediction With An Optimal Transport Loss

Paul KRZAKALA, LTCI (Télécom Paris) & CMAP (Polytechnique)

&


J. Yang


R. Flamary

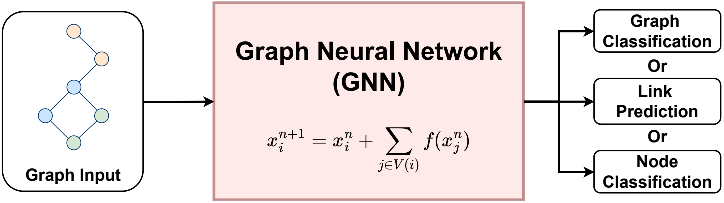

F. d'Alché-Buc


C. Laclau


M. Labeau

Supervised Graph Prediction

Graphs as output



A (very) naive approach

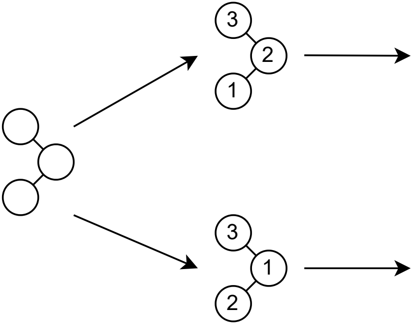
Goal: from input $x \in \mathcal{X}$ learn to predict graph $g \in \mathrm{g}$.

Naive approach: Represent graph g by adjacency matrix $A \in [0,1]^{m \times m}$

Minimize:

$$\min_{\theta} \frac{1}{n} \sum_{k=1}^{n} ||f_\theta(x_k) - A_k||_2^2$$

With some neural net.

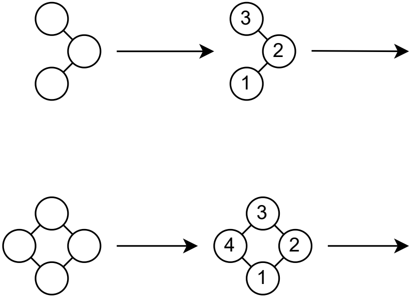


$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Our framework needs to be graph isomorphism invariant!

Graph Prediction: challenges

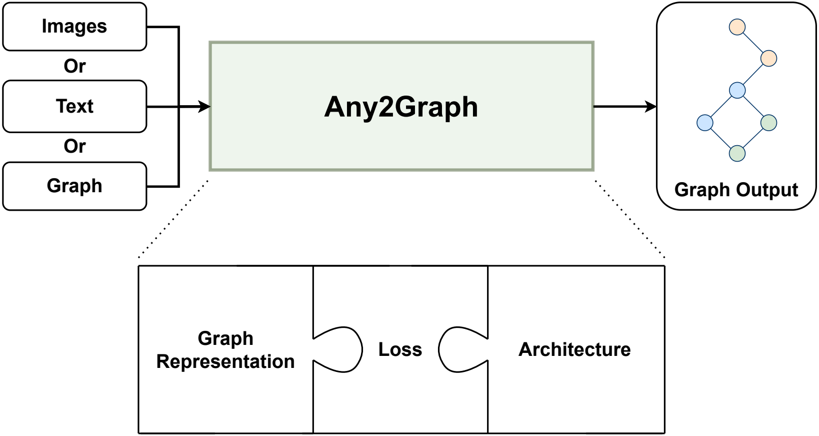


$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Our framework needs to deal with graphs of arbitrary sizes!

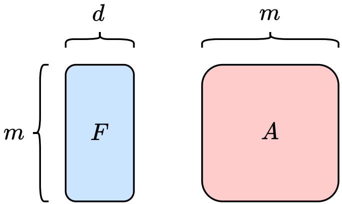
Our framework



Graph Representation

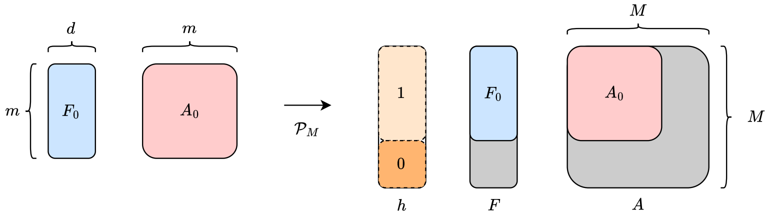
Starting point

Classical representation of **graph of size** m with **features of dimension** d:



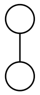
Padding

We pad all graph to have same size M:



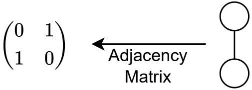
Pipeline

Example for $M = 3$:

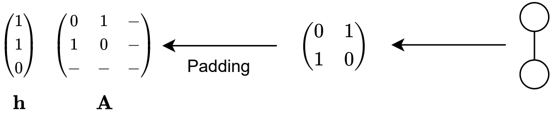


Pipeline

Example for $M = 3$:



Example for $M = 3$:



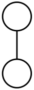
Pipeline

Example for $M = 3$:

Input

$\mathbf{x}$

$$\begin{pmatrix}1\\1\\0\end{pmatrix} \quad \begin{pmatrix}0 & 1 & -\\ 1 & 0 & -\\ - & - & -\end{pmatrix} \longleftarrow \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \longleftarrow$$



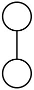
$\mathbf{h}$ $\quad$ $\mathbf{A}$

Pipeline

Example for $M = 3$:

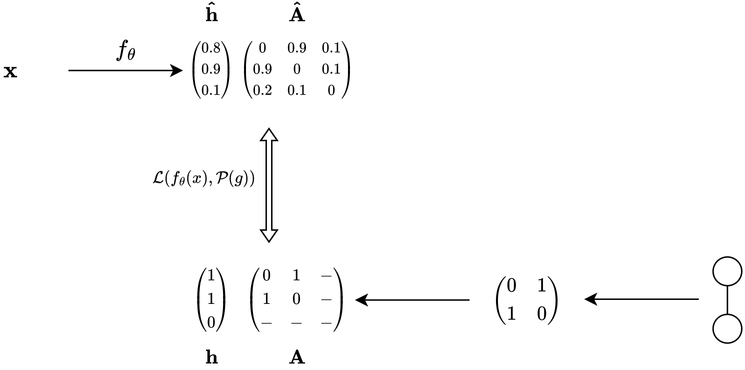
$$\mathbf{x} \xrightarrow{f_\theta} \underbrace{\begin{pmatrix} 0.8 \\ 0.9 \\ 0.1 \end{pmatrix}}_{\hat{\mathbf{h}}} \quad \underbrace{\begin{pmatrix} 0 & 0.9 & 0.1 \\ 0.9 & 0 & 0.1 \\ 0.2 & 0.1 & 0 \end{pmatrix}}_{\hat{\mathbf{A}}}$$

$$\underbrace{\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}}_{\mathbf{h}} \quad \underbrace{\begin{pmatrix} 0 & 1 & - \\ 1 & 0 & - \\ - & - & - \end{pmatrix}}_{\mathbf{A}} \longleftarrow \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \longleftarrow$$



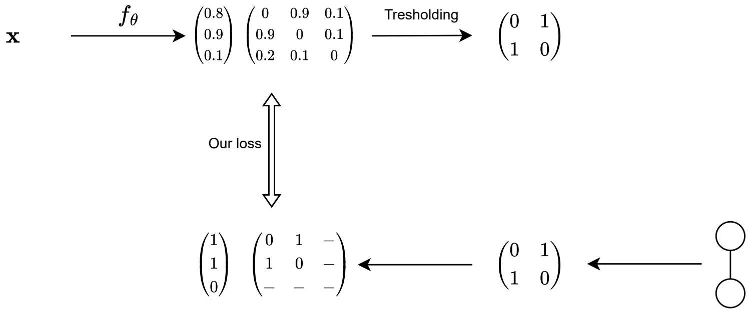
Pipeline

Example for $M = 3$:



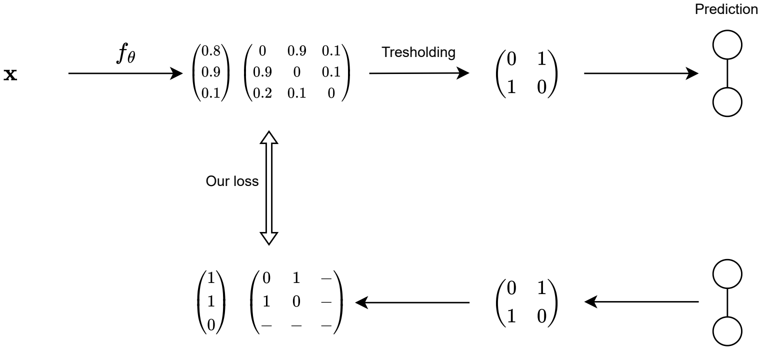
Pipeline

Example for $M = 3$:



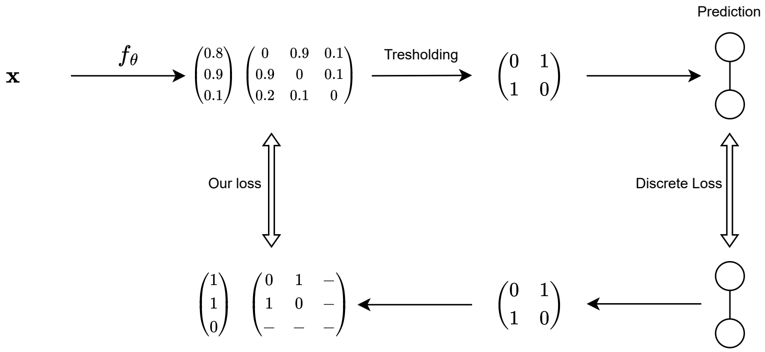
Pipeline

Example for $M = 3$:



Pipeline

Example for $M = 3$:



PMFGW Loss

Designing a loss:

We need a loss $\mathcal{L}(\hat{y}, y)$ to compare predicted triplet $\hat{y} = (\hat{\mathsf{h}}, \hat{\mathsf{F}}, \hat{\mathsf{A}})$ and target triplet $y = (\mathsf{h}, \mathsf{F}, \mathsf{A})$.

Requirements:

- Differentiable
- Permutation Invariant
- Efficient computation

- Mémoli introduce Gromov-Wasserstein (GW) distance to compare mm-spaces [1]

$$\min_{\pi \in \Pi(\mu_{\mathcal{X}}, \mu_{\mathcal{Y}})} \int_{\mathcal{X} \times \mathcal{Y}} \int_{\mathcal{X} \times \mathcal{Y}} \ell(d_{\mathcal{X}}(x, x'), d_{\mathcal{Y}}(y, y')) \, \mathrm{d}\pi(x, y) \, \mathrm{d}\pi(x', y')$$

where $\Pi(\mu_{\mathcal{X}}, \mu_{\mathcal{Y}})$ is the set of transport plan

$$\Pi(\mu_{\mathcal{X}}, \mu_{\mathcal{Y}}) = \{\pi \in \mathcal{P}(\mathcal{X} \times \mathcal{Y}) \mid \pi_{\mathcal{X}} = \mu_{\mathcal{X}}, \pi_{\mathcal{Y}} = \mu_{\mathcal{Y}}\}$$

OT at the rescue! A brief history.

- Mémoli introduce Gromov-Wasserstein (GW) distance to compare mm-spaces [1]
- Peyré et al. applied GW to compare graphs. [2]

$$\min_{\mathbf{T}\in\pi_{n,m}} \sum_{i,j=1}^{n} \sum_{k,l=1}^{m} T_{i,k} T_{j,l} \ell(\hat{A}_{i,j}, A_{k,l})$$

where $\pi_{n,m}$ is the set of discrete transport plan

$$\pi_{n,m} = \{\mathbf{T} \in [0,1]^{n\times m} \mid \sum_i T_{i,j} = \sum_j T_{i,j} = 1\}$$

OT at the rescue! A brief history.

- Mémoli introduce Gromov-Wasserstein (GW) distance to compare mm-spaces [1]
- Peyré et al. applied GW to compare graphs. [2]
- Vayer et al. introduce FGW to compare labeled graphs [3]

$$\min_{\mathbf{T}\in\pi_{n,m}} \sum_{i=1}^{n}\sum_{k=1}^{m} T_{i,k}\ell_F(\hat{F}_i, F_k) + \sum_{i,j=1}^{n}\sum_{k,l=1}^{m} T_{i,k}T_{j,l}\ell_A(\hat{A}_{i,j}, A_{k,l})$$

where $\pi_{n,m}$ is the set of discrete transport plan

$$\pi_{n,m} = \{\mathbf{T} \in [0,1]^{n\times m} \mid \sum_i T_{i,j} = \sum_j T_{i,j} = 1\}$$

OT at the rescue! A brief history.

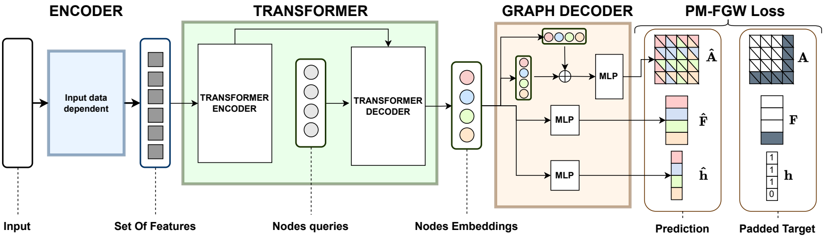
- Mémoli introduce Gromov-Wasserstein (GW) distance to compare mm-spaces [1]
- Peyré et al. applied GW to compare graphs. [2]
- Vayer et al. introduce FGW to compare labeled graphs [3]
- Our work introduce the PMFGW to compare predicted triplet $(\hat{\mathrm{h}}, \hat{\mathrm{F}}, \hat{\mathrm{A}})$ and padded target $(\mathrm{h}, \mathrm{F}, \mathrm{A})$

$$\min_{\mathrm{T} \in \pi_M} \sum_{i,k=1}^{M} T_{i,k} \ell_h(\hat{h}_i, h_k) + \sum_{i,k=1}^{M} T_{i,k} \ell_F(\hat{F}_i, F_k) h_i + \sum_{i,j,k,l=1}^{M} T_{i,k} T_{j,l} \ell_A(\hat{A}_{i,j}, A_{k,l}) h_i h_j$$

$$\pi_M = \{\mathrm{T} \in [0,1]^{M \times M} \mid \sum_i T_{i,j} = \sum_j T_{i,j} = 1\}$$

Architecture

Architecture



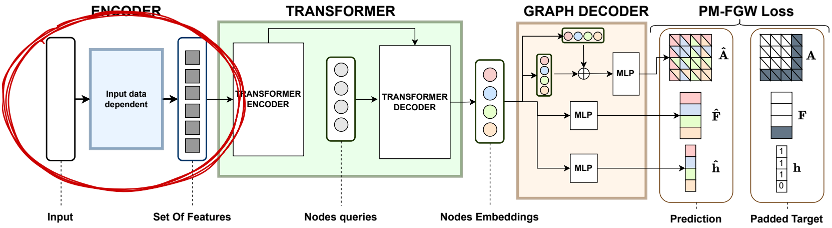
- The **encoder** extract a set of features $x \to (\mathrm{V}_1, ..., \mathrm{V}_k) \in \mathbb{R}^{k \times d}$
- The **transformer** translate them into M nodes embedding $(\mathrm{Z}_1, ..., \mathrm{Z}_M) \to \in \mathbb{R}^{M \times d}$
- The **decoder** produce the graph following

$$\hat{h}_i = \sigma(\mathrm{MLP}_m(\mathbf{z}_i)) \qquad \forall i \in \{1, \ldots, M\}$$

$$\hat{F}_i = \mathrm{MLP}_f(\mathbf{z}_i) \qquad \forall i \in \{1, \ldots, M\}$$

$$\hat{A}_{i,j} = \sigma(\mathrm{MLP}_s(\mathbf{z}_i + \mathbf{z}_j)) \qquad \forall i, j \in \{1, \ldots, M\}^2$$

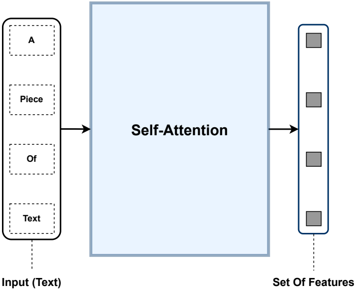
Architecture

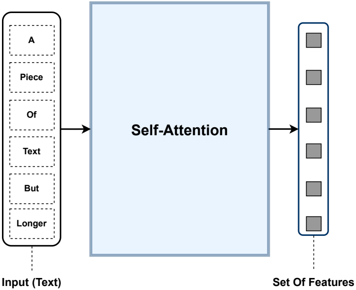


Philosophy of the encoder

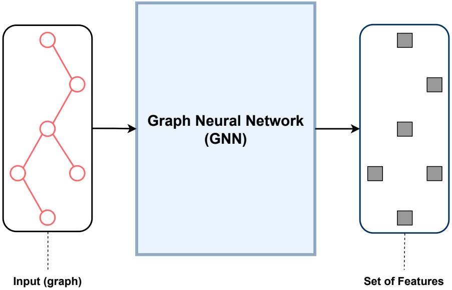
- Architecture adapts to input modality
- Can leverage pretrained models
- Must extract a **list of features**. Avoid vector bottleneck.

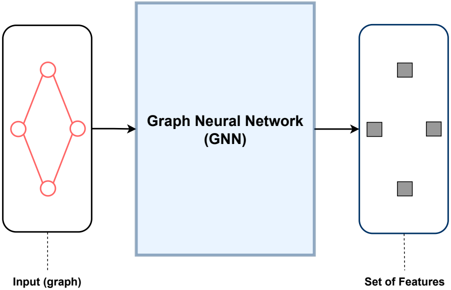
Encoder (text input)



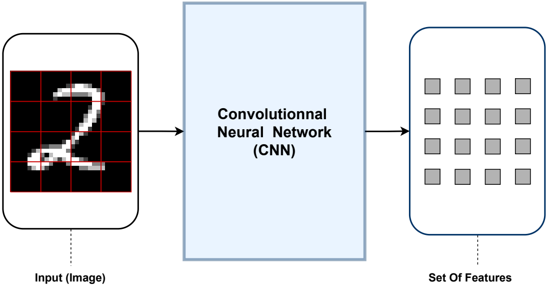


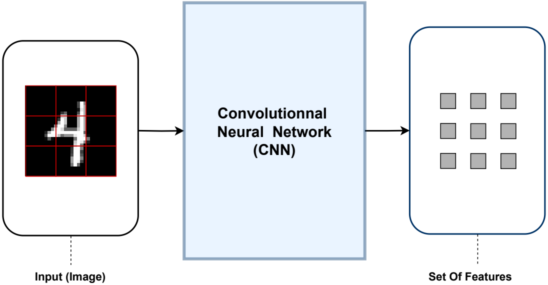
Encoder (graph input)





Encoder (image input)





Applying the framework

Prediction performances

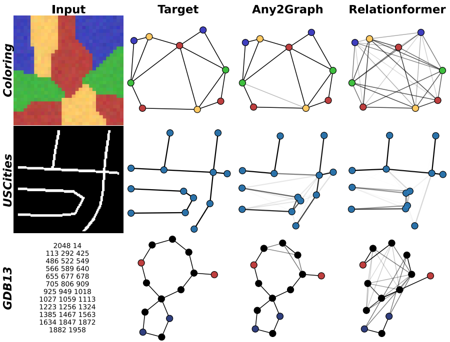
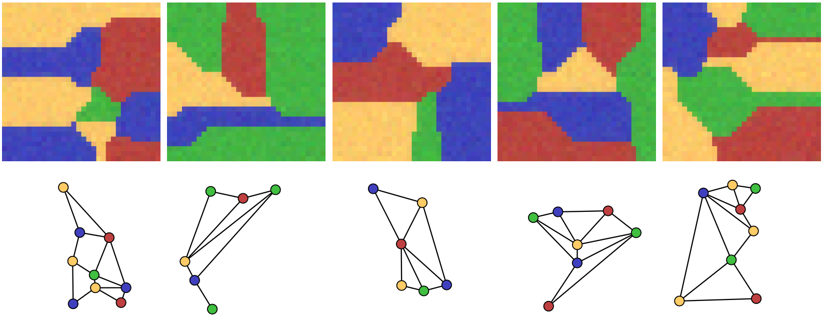


Figure 4: Qualitative comparison of Any2Graph (ours) and Relationformer.

Datasets	Model	Edit Distance ↓
Coloring	FGWBary-NN*	6.73
	Relationformer	5.47
	Any2Graph (Ours)	**0.20**
Toulouse	FGWBary-NN*	8.11
	Relationformer	**0.13**
	Any2Graph (Ours)	**0.13**
USCities	Relationformer	2.09
	Any2Graph (Ours)	**1.86**
QM9	FGWBary-ILE*	2.84
	Relationformer	3.80
	Any2Graph (Ours)	**2.13**
GDB13	Relationformer	8.83
	Any2Graph (Ours)	**3.63**

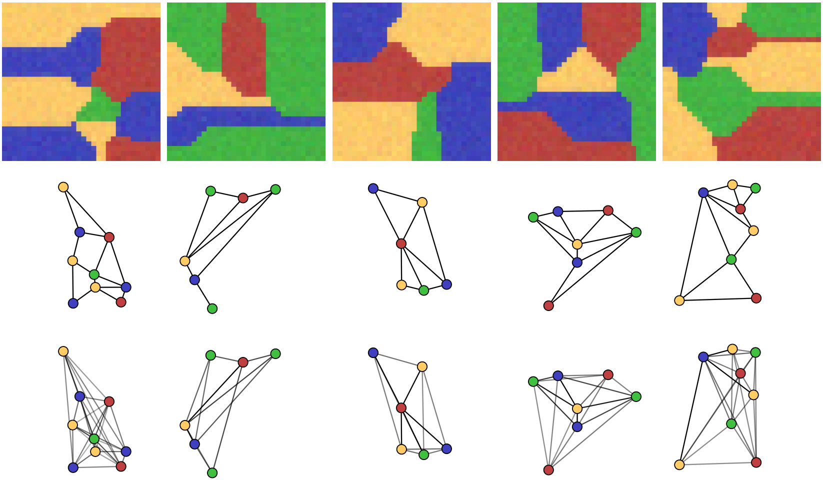
Table 1: Prediction performances measured with (test) edit distance.

Training Dynamics



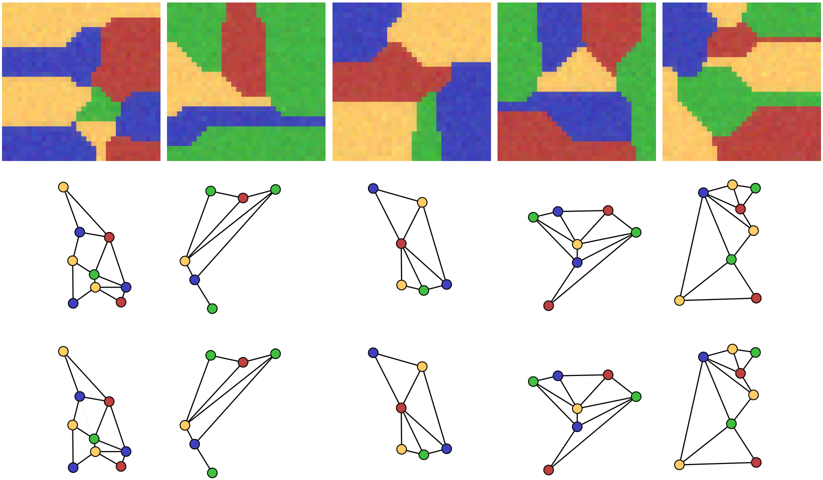
$\rightarrow 100K$ samples

... after 5 epochs



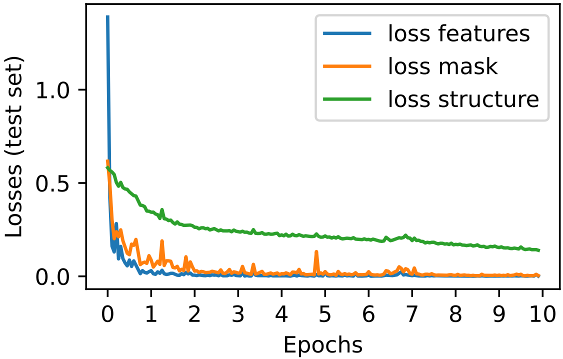
Number of nodes ✓ Nodes features ✓

... after 100 epochs



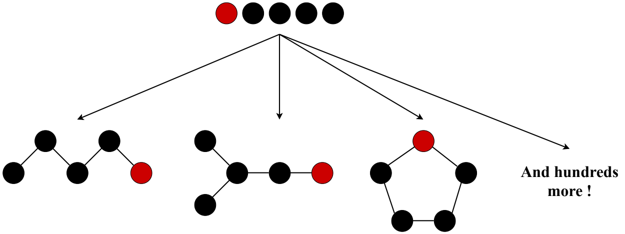
Number of nodes ✓ Nodes features ✓ Structure ✓

Decomposing the loss



A more challenging case

For some datasets (e.g. molecules) the prediction of nodes poorly guides the prediction of the structure:

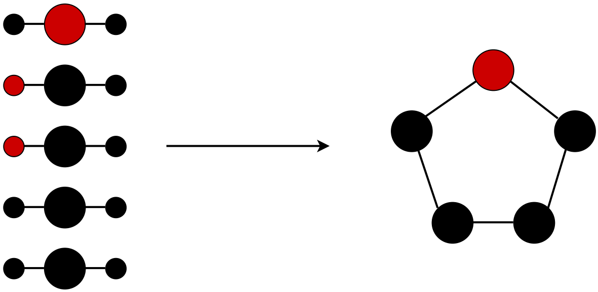


And the good dynamic does not occur.

Feature Diffusion trick

We ask the model to predict the features of a nodes + the features of its neighbors, formally:

$$F \mapsto [F, AF]$$



Effect of Feature Diffusion

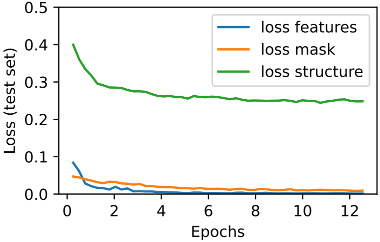


Figure 5: Without Feature Diffusion.

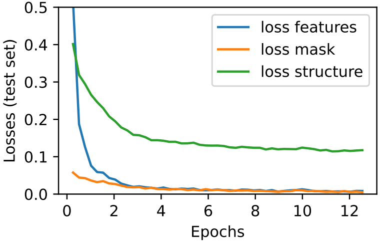


Figure 6: With Feature Diffusion.

Feature diffusion also helps the OT solver converge faster!

Thank you for your `softmax`$(QK^T)V$!

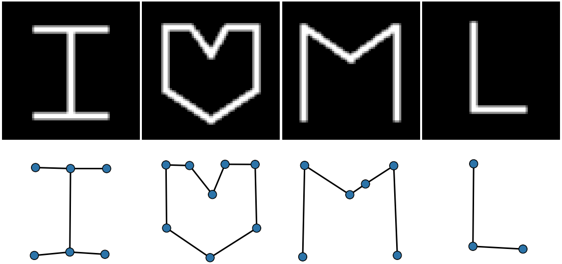


Figure 7: Any2Graph performing an Img2Graph task.

M is the maximum number of nodes the model can use.

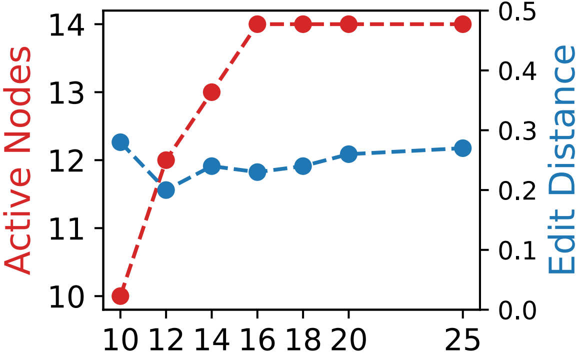


Figure 8: Effect of M.

Effect the hyperparameters: α

$\boldsymbol{\alpha} = [\alpha_h, \alpha_F, \alpha_A]$ are the weights balancing the terms of the loss.

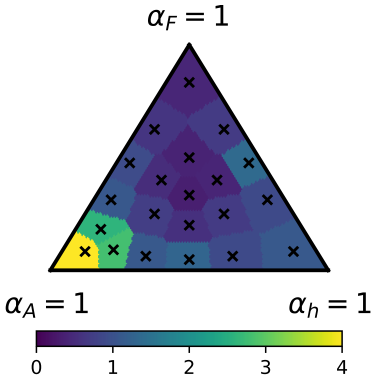


Figure 9: Effect of $\boldsymbol{\alpha}$ on the performances (grid search on the simplex).

Effect the hyperparameters: α

Setting α_A too high prevents the good dynamic!

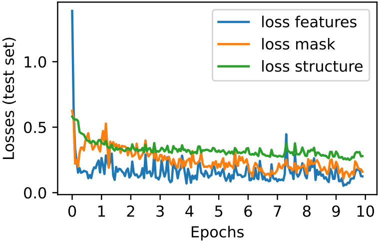


Figure 10: $\alpha = [10, 1, 1]$.

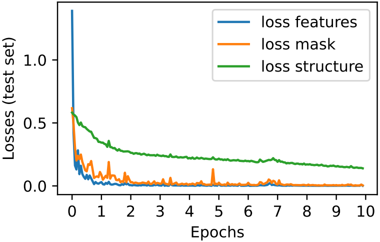


Figure 11: $\alpha = [1, 1, 1]$.

A toy example

A target graph , $g = (\mathsf{F}, \mathsf{A})$ where

$$\mathsf{F} = \begin{pmatrix} \mathsf{f}_1 \\ \mathsf{f}_2 \end{pmatrix} ; \mathsf{A} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

For $a = h = 1$ the prediction is perfect $\mathcal{L}(\hat{y}_{1,1}, \mathcal{P}_3(g)) = 0$

$$\hat{\mathsf{h}} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} ; \hat{\mathsf{F}} = \begin{pmatrix} \mathsf{f}_1 \\ \mathsf{f}_2 \\ \mathsf{f}_2 \end{pmatrix} ; \hat{\mathsf{A}} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

A toy example

A target graph , $g = (\mathsf{F}, \mathsf{A})$ where

$$\mathsf{F} = \begin{pmatrix} \mathsf{f}_1 \\ \mathsf{f}_2 \end{pmatrix} ; \mathsf{A} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

For $a = h = 0$ the prediction is perfect $\mathcal{L}(\hat{y}_{0,0}, \mathcal{P}(g)) = 0$

$$\hat{\mathsf{h}} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} ; \hat{\mathsf{F}} = \begin{pmatrix} \mathsf{f}_1 \\ \mathsf{f}_2 \\ \mathsf{f}_2 \end{pmatrix} ; \hat{\mathsf{A}} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

We can plot the loss landscape

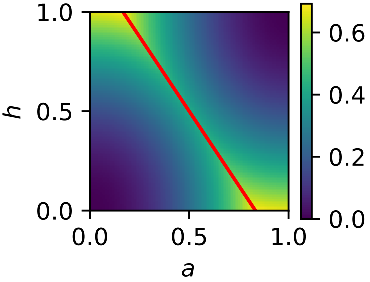


Figure 12: $\ell(a,h) = \mathcal{L}(\hat{y}_{a,h}, \mathcal{P}(g))$

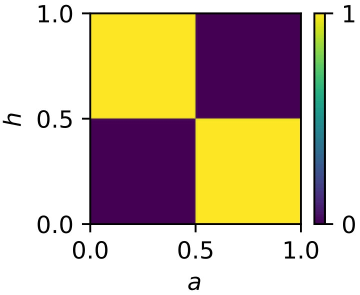


Figure 13: $\ell(a,h) = \mathrm{ED}(\mathcal{P}^{-1}(\hat{y}_{a,h}), g)$

Computing the loss

Recall the expression of the loss:

$$\min_{T\in\pi_M} \sum_{i,k=1}^{M} T_{i,k}\mathcal{L}_h(\hat{h}_i, h_k) + \sum_{i,k=1}^{M} T_{i,k}\mathcal{L}_F(\hat{F}_i, F_k)h_k + \sum_{i,j,k,l=1}^{M} T_{i,k}T_{j,l}\mathcal{L}_A(\hat{A}_{i,j}, A_{k,l})h_k h_l$$

The inner optimization problem writes

$$\min_{\mathsf{T}\in\pi_M} \langle \mathsf{T}, \mathsf{U}\rangle + \langle \mathsf{T}, \mathsf{L}\otimes \mathsf{T}\rangle$$

For $\mathsf{U}_{i,k} = \ell_h(\hat{h}_i, h_k) + \ell_F(\hat{f}_i, f_k)h_k$
and $(\mathsf{L}\otimes \mathsf{T})_{i,k} = \sum_{j,l} T_{j,l}\ell_A(\hat{A}_{i,j}, A_{k,l})h_k h_l$

Computing the loss

Conditionnal Gradient solver:

$$\mathsf{T}^{(k+1)} = \min_{\mathsf{T} \in \pi_M} \langle \mathsf{T}, \mathsf{C}^{(k)} \rangle$$

- Each step is a standard OT problem!
- With cost $\mathsf{C}^{(k)} = \mathsf{U} + \mathsf{L} \otimes \mathsf{T}^{(k)}$
- We provide a factorisation for fast computation of $\mathsf{L} \otimes \mathsf{T}^{(k)}$