Any2Graph: Deep end-to-end supervised graph prediction with an optimal transport loss

Rémi Flamary - CMAP, École Polytechnique, Institut Polytechnique de Paris

June 20 2024

Learning and Optimization in Luminy (LOL 24)


P. Krzakala


J. Yang


F. d'Alché-Buc


C. Laclau

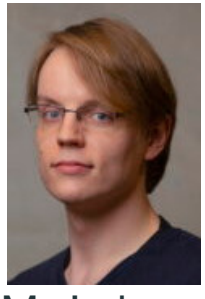
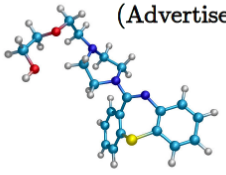

M. Labeau

Table of content

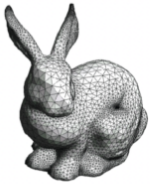
Supervised Graph Prediction

Graphs are everywhere

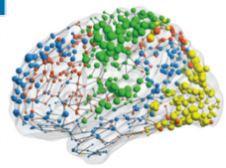
Social networks
(Advertisement)



Drug/Material
molecules
(Chemistry)

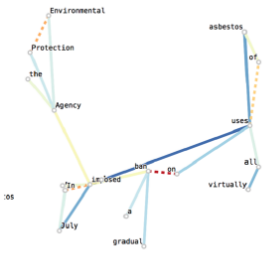


3D Meshes
(Computer Graphics)

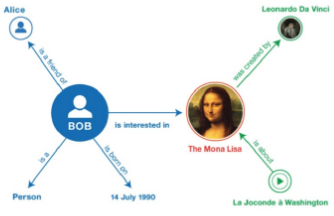


Brain
connectivity
(Neuroscience)

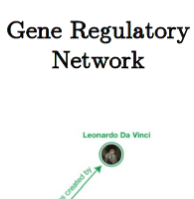
Transportation
networks



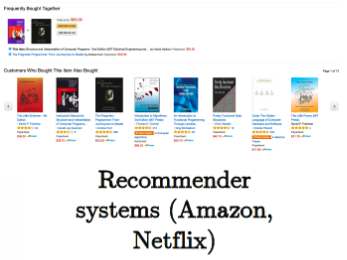
Words relationships
(NLP)



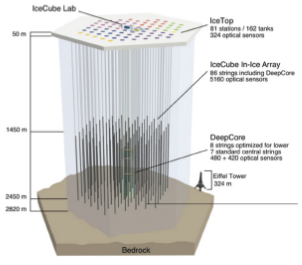
Knowledge graph
(Causality)



Gene Regulatory
Network

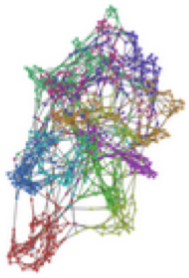


Recommender
systems (Amazon,
Netflix)



Neutrino
detection (High-
energy Physics)

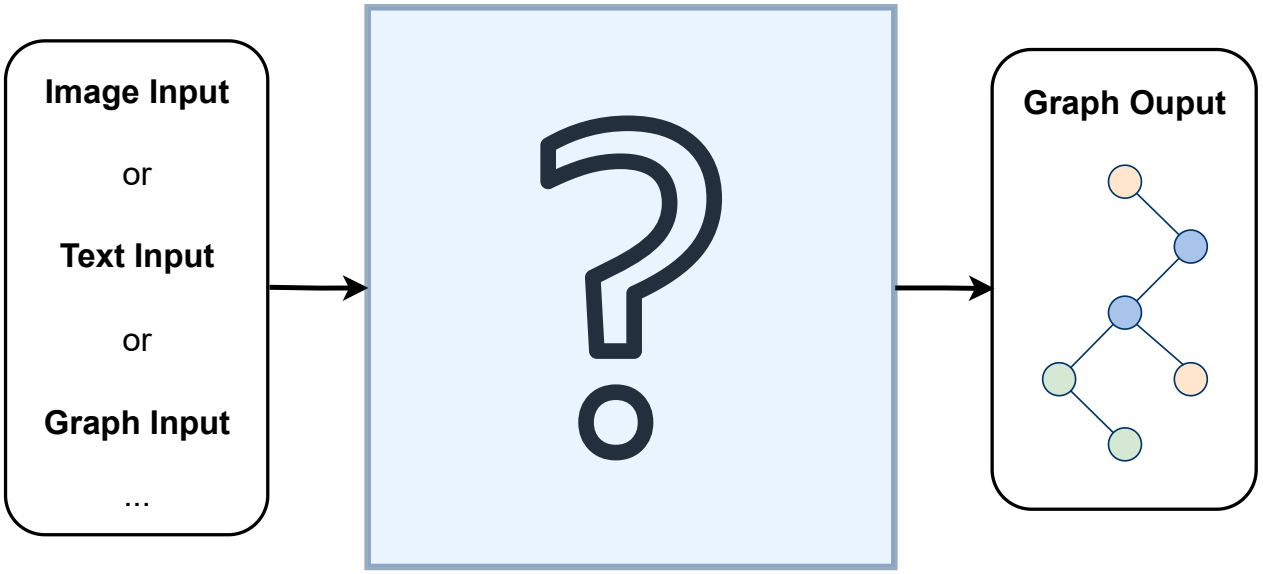
=



Graphs/
Networks

- Classical approaches : spectral and Fourier based analysis and processing.
- More recently : deep learning on graphs (GNN).
- This talk is about **supervised graph prediction**.

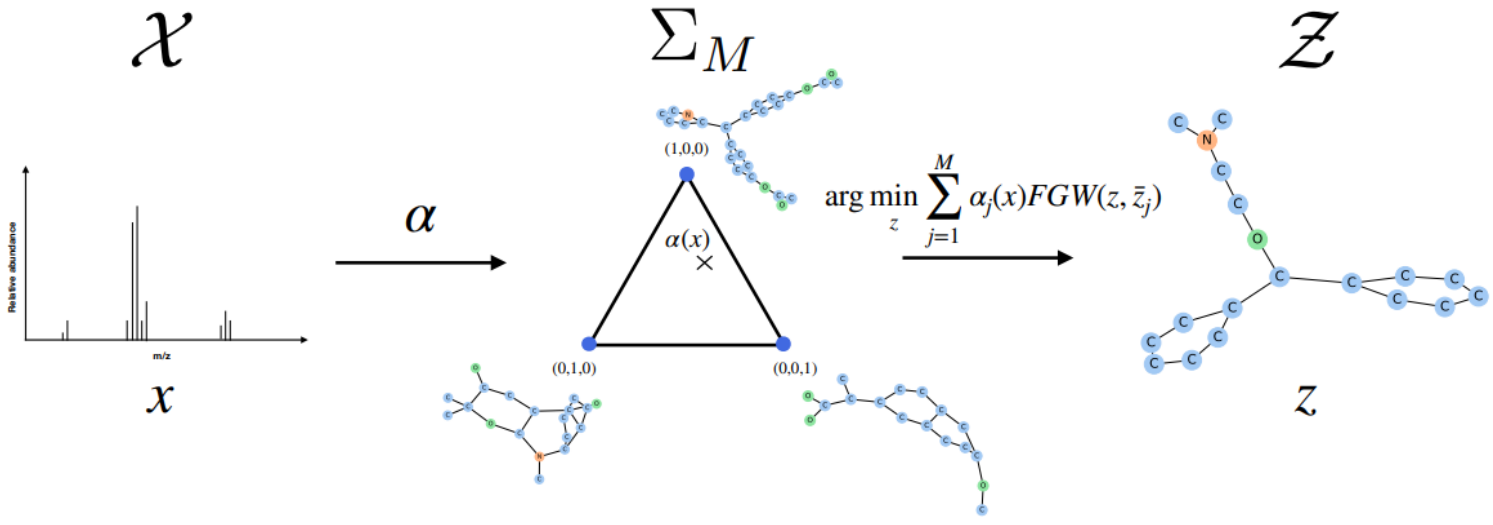
Supervised Graph prediction



Supervised graph prediction

- Special case of structured prediction.
- Objective : learn a function f predicting a graph g from an input x.
- Applications of SGP:
  - knowledge graph extraction [Melnyk et al., 2022]
  - Natural language processing [Dozat and Manning, 2017]
  - Molecule identification in chemistry [Brouard et al., 2016]
- Most methods are data specific and/or slow at inference.

How to predict a graph? (1)



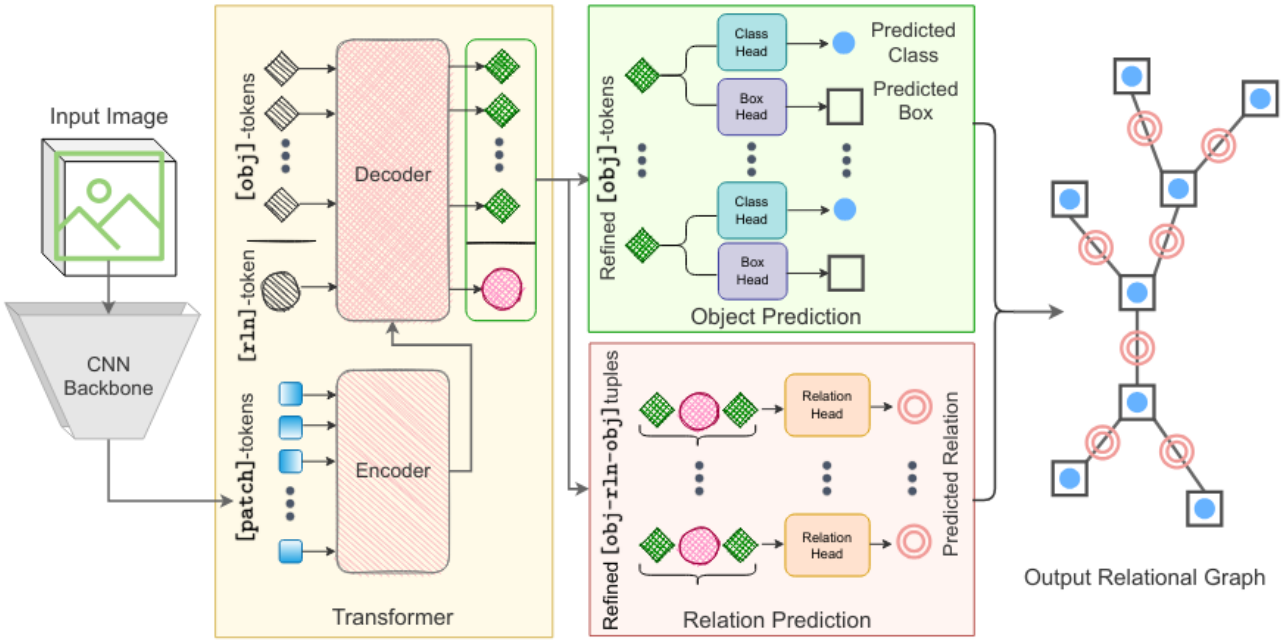
Surrogate based methods [Brouard et al., 2016, El Ahmad et al., 2024]

- Represent graph as a vector in a high dimensional space (RKHS).
- Learn a mapping from input to this space.
- Decode the vector to a graph (e.g. search among finite candidates).

Graph prediction with OT barycenters [Brogat-Motte et al., 2022]

- Decoding done with a conditional FGW.
- Can learn parametric (NN) and non-parametric (kernel) models.
- Slow at training and prediction due to barycenter computation.

How to predict a graph? (2)

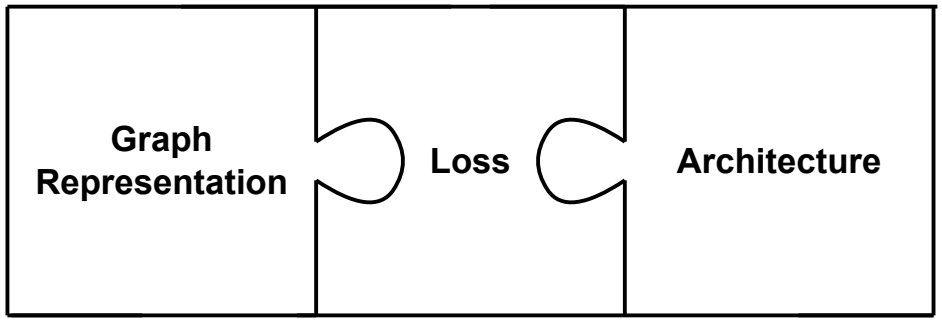


Relationformer [Shit et al., 2022]

- Predict a graph of max size M and activation scores for nodes to keep.
- Encoder-Decoder Transformer to predict node embeddings.
- Loss solves linear assignment problem (Hungarian) and uses assignment in quadratic loss between graphs of same size (padding the target).
- Fast prediction (thresholding) of graphs but focused on Image2Graph.

Deep end-to-end supervised graph prediction

Any2Graph framework



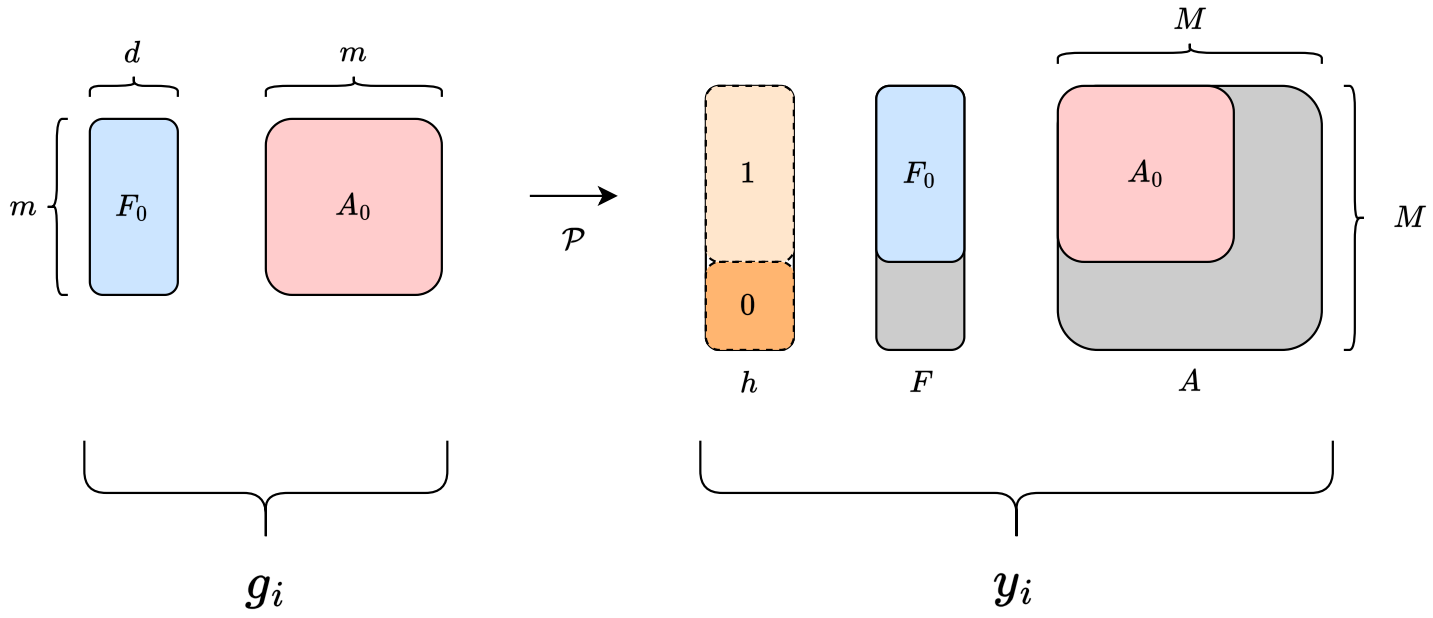
Principle [Krzakala et al., 2024]

- End-to-end supervised graph prediction with a deep learning framework.
- Learning optimization problem:

$$\min_{\theta} \quad \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f_\theta(x_i), \mathcal{P}(g_i)). \tag{1}$$

- $\{x_i, g_i\}$ are the input/output training data and $\mathcal{P}$ is a padding operator.
- f_θ is a transformer neural network with fixed max number of nodes M.
- f_θ also predicts is a padding vector $\hat{\boldsymbol{h}}$ (selection of subset of nodes).
- $\mathcal{L}$ is an optimal transport based loss for permutation invariant prediction.

Forcing the target graph size with padding

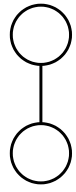


Padding Operator

$$\mathcal{P}(g) = \left(\begin{pmatrix} \mathbf{1}_m \\ \mathbf{0}_{M-m} \end{pmatrix}, \begin{pmatrix} \mathbf{F}_m \\ \mathbf{0}_{M-m} \end{pmatrix}, \begin{pmatrix} \mathbf{A}_m & \mathbf{0}_{M-m} \\ \mathbf{0}_{M-m}^T & \mathbf{0}_{M-m,M-m} \end{pmatrix} \right)$$

- $g = (\mathbf{F}_m, \mathbf{A}_m) \in \mathcal{G}_m$ is a labeled graph of size $m \leq M$.
- $\mathbf{A}_m$ is the adjacency matrix and $\mathbf{F}_m$ is the node feature matrix.
- $\mathcal{P}(g) = (\mathbf{h}, \mathbf{F}, \mathbf{A})$ where $\mathbf{h} \in 0, 1^M$ is a padding identification vector.
- Pad all graphs in training set to have the same size M.

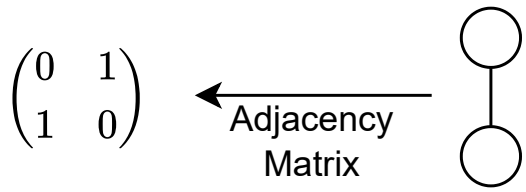
End-to-end SGP pipeline



Target Graph

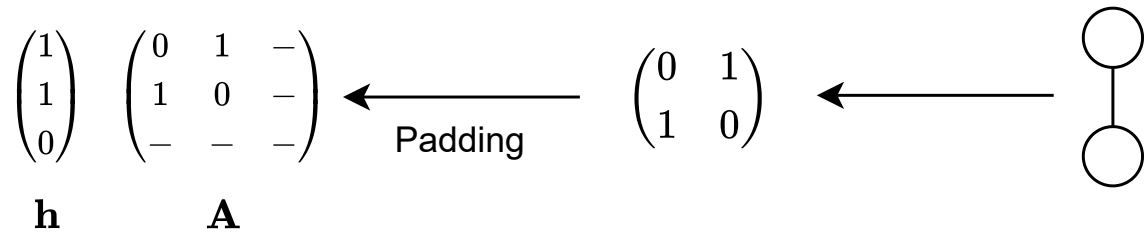
- Pad target graphs to have same size M.
- Predict with f_θ (continuous) size M graph with padding vector $\hat{\cdot}\boldsymbol{h}$.
- Minimize OT loss L between predicted and padded target graphs.
- At test time, thresholding recovers discrete graph.

End-to-end SGP pipeline



- Pad target graphs to have same size M.
- Predict with f_θ (continuous) size M graph with padding vector $\hat{\cdot}\boldsymbol{h}$.
- Minimize OT loss L between predicted and padded target graphs.
- At test time, thresholding recovers discrete graph.

End-to-end SGP pipeline

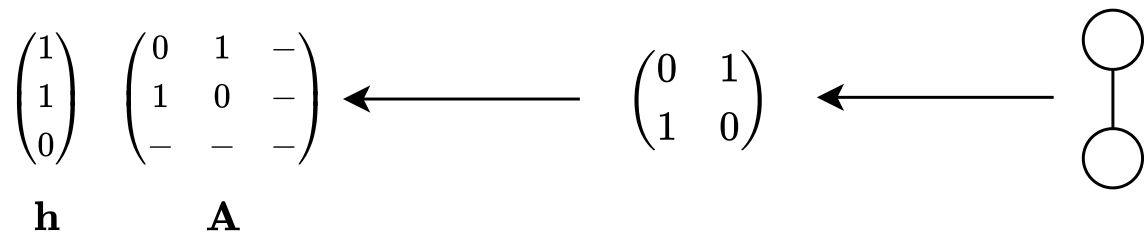


- Pad target graphs to have same size M.
- Predict with f_θ (continuous) size M graph with padding vector $\hat{h}$.
- Minimize OT loss L between predicted and padded target graphs.
- At test time, thresholding recovers discrete graph.

End-to-end SGP pipeline

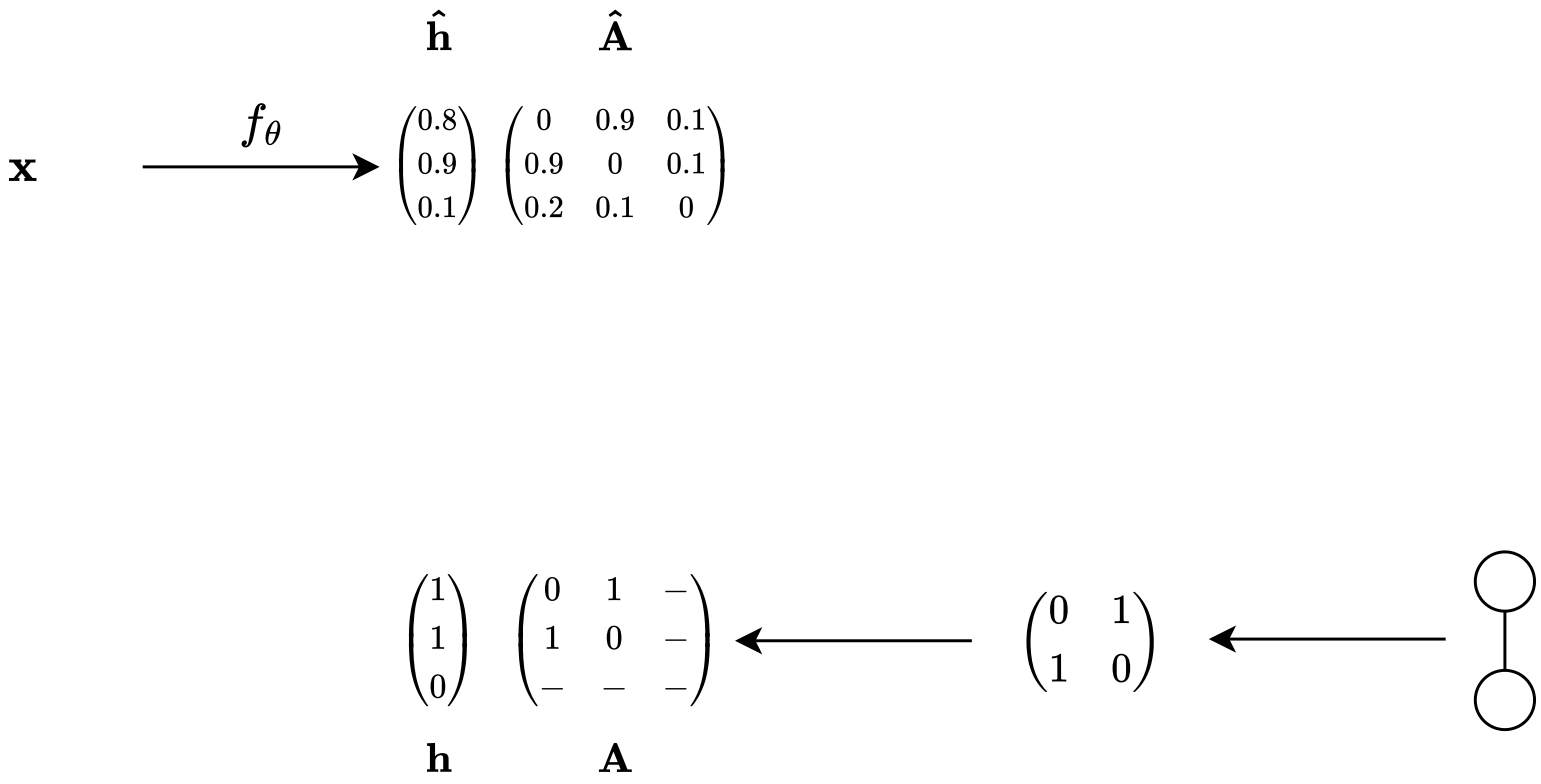
Input

$\mathbf{x}$



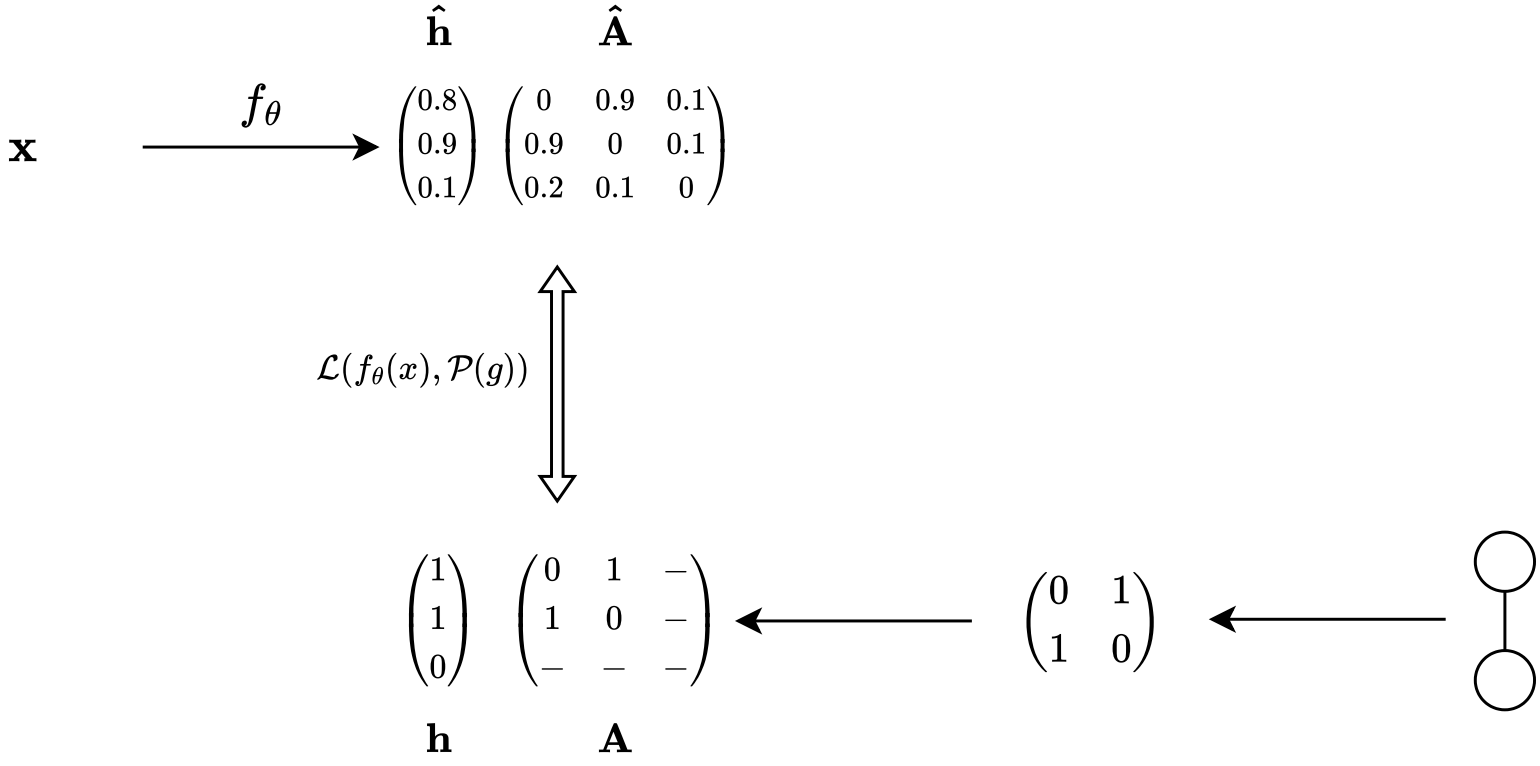
- Pad target graphs to have same size M.
- Predict with f_θ (continuous) size M graph with padding vector $\hat{}\boldsymbol{h}$.
- Minimize OT loss L between predicted and padded target graphs.
- At test time, thresholding recovers discrete graph.

End-to-end SGP pipeline



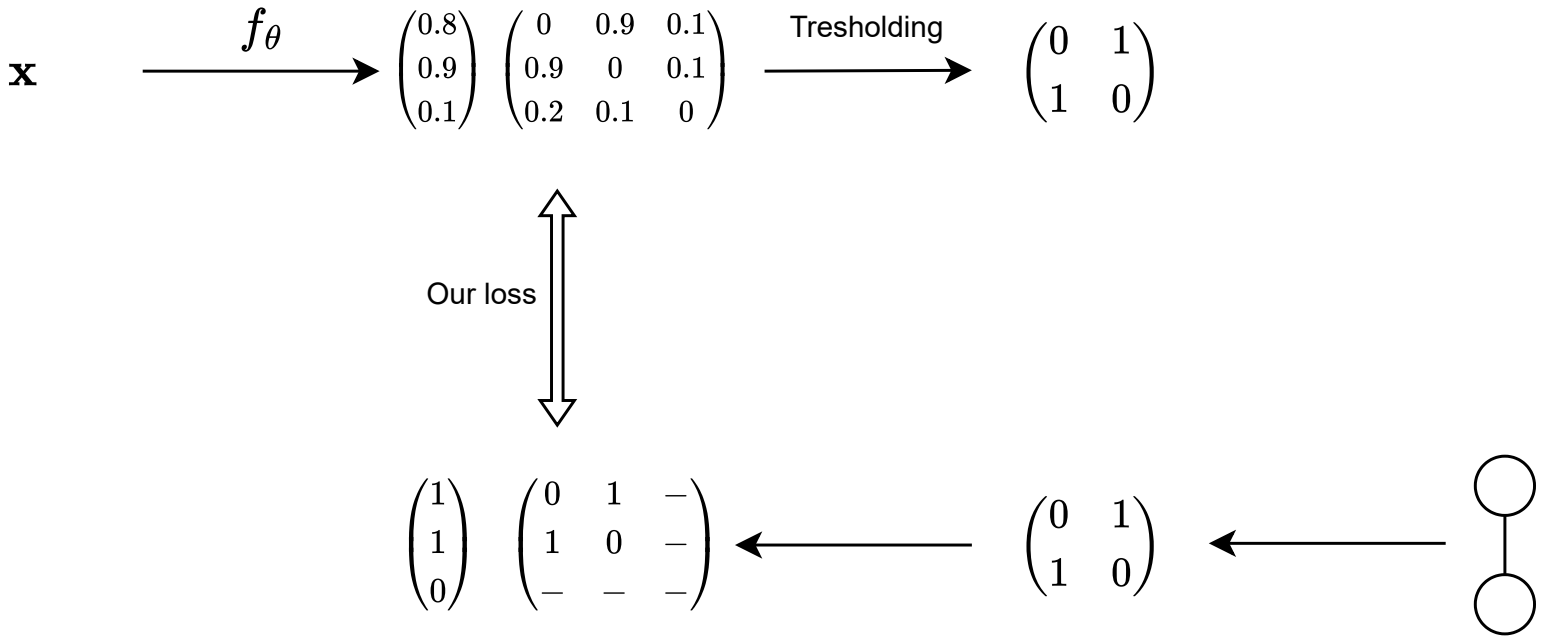
- Pad target graphs to have same size M.
- Predict with f_θ (continuous) size M graph with padding vector $\hat{\boldsymbol{h}}$.
- Minimize OT loss L between predicted and padded target graphs.
- At test time, thresholding recovers discrete graph.

End-to-end SGP pipeline



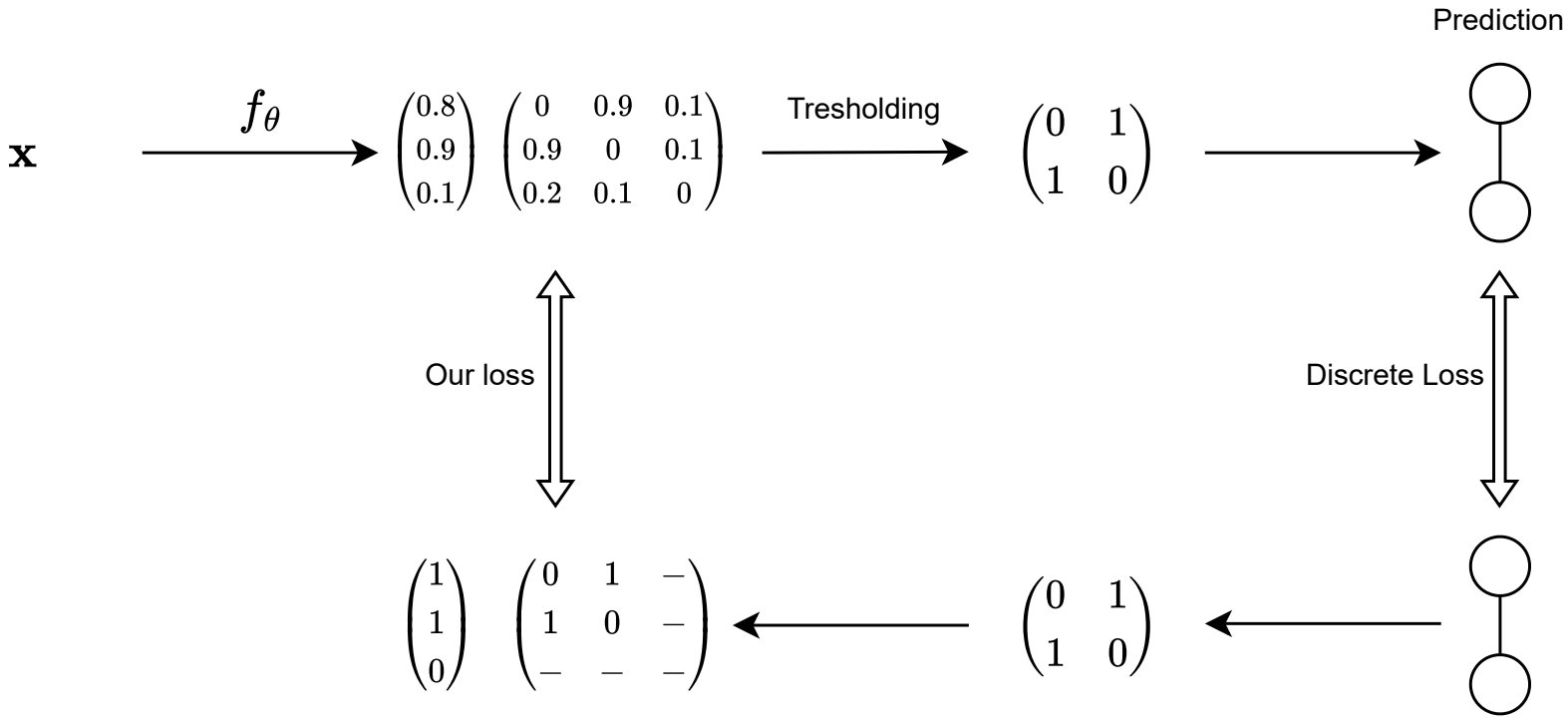
- Pad target graphs to have same size M.
- Predict with f_θ (continuous) size M graph with padding vector $\hat{.}\boldsymbol{h}$.
- Minimize OT loss L between predicted and padded target graphs.
- At test time, thresholding recovers discrete graph.

End-to-end SGP pipeline



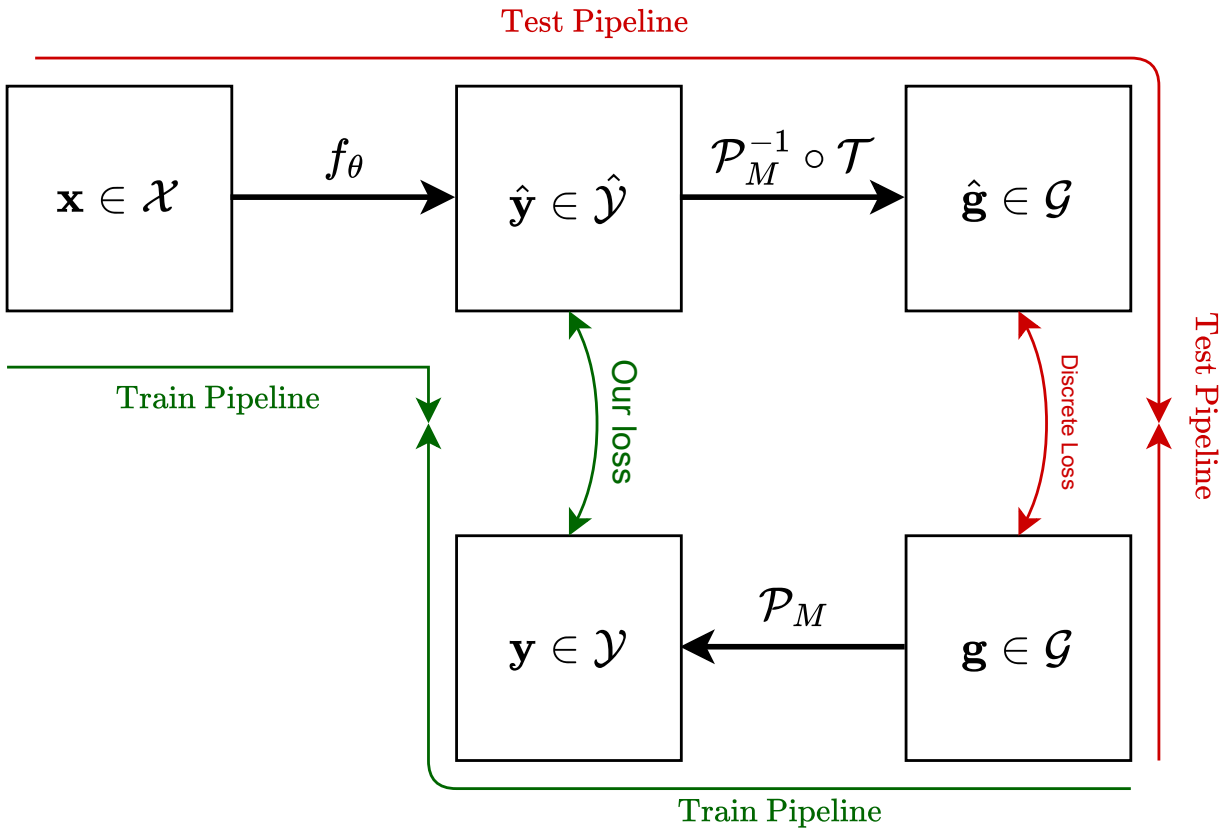
- Pad target graphs to have same size M.
- Predict with f_θ (continuous) size M graph with padding vector $\hat{.}\boldsymbol{h}$.
- Minimize OT loss L between predicted and padded target graphs.
- At test time, thresholding recovers discrete graph.

End-to-end SGP pipeline



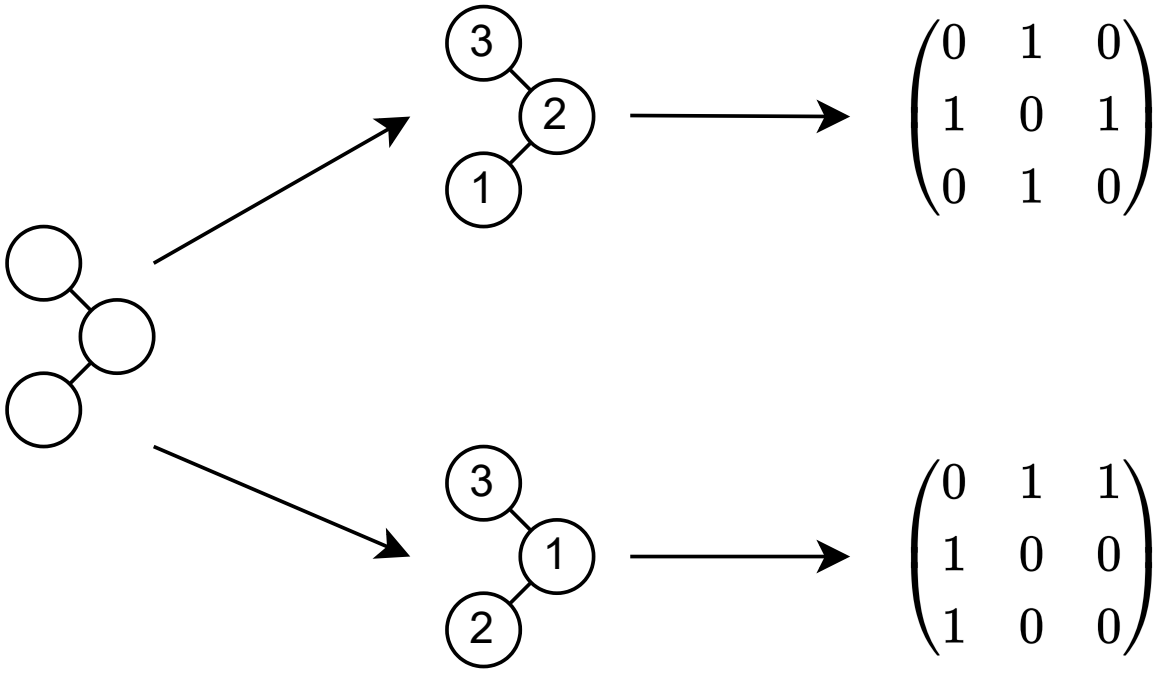
- Pad target graphs to have same size M.
- Predict with f_θ (continuous) size M graph with padding vector $\hat{\boldsymbol{h}}$.
- Minimize OT loss L between predicted and padded target graphs.
- At test time, thresholding recovers discrete graph.

End-to-end SGP pipeline



- Pad target graphs to have same size M.
- Predict with f_θ (continuous) size M graph with padding vector $\hat{\boldsymbol{h}}$.
- Minimize OT loss L between predicted and padded target graphs.
- At test time, thresholding recovers discrete graph.

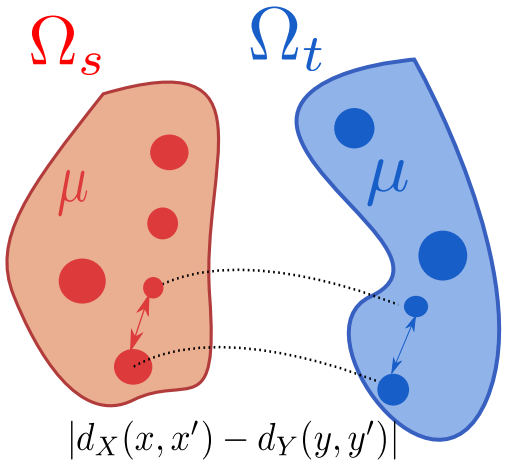
Data fitting loss for graph prediction

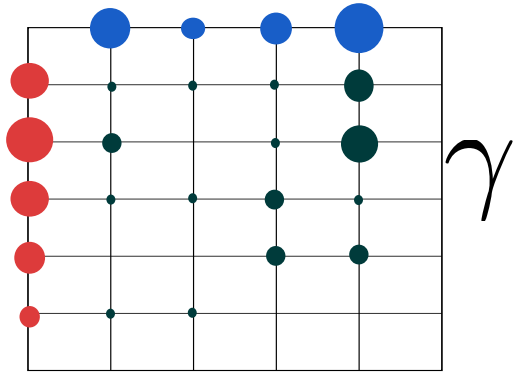


Requirements for loss $\mathcal{L}$

- Works between continuous predicted triplet $\hat{y} = (\hat{\mathbf{h}}, \hat{\mathbf{F}}, \hat{\mathbf{A}})$ and discrete target triplet $y = (\mathbf{h}, \mathbf{F}, \mathbf{A})$ with padding.
- Permutation (graph isomorphism) Invariant.
- Differentiable : for end-to-end training.
- Fused Gromov Wasserstein proposed in [Brogat-Motte et al., 2022].

Gromov-Wasserstein and Fused Gromov-Wasserstein





Inspired from Gabriel Peyré

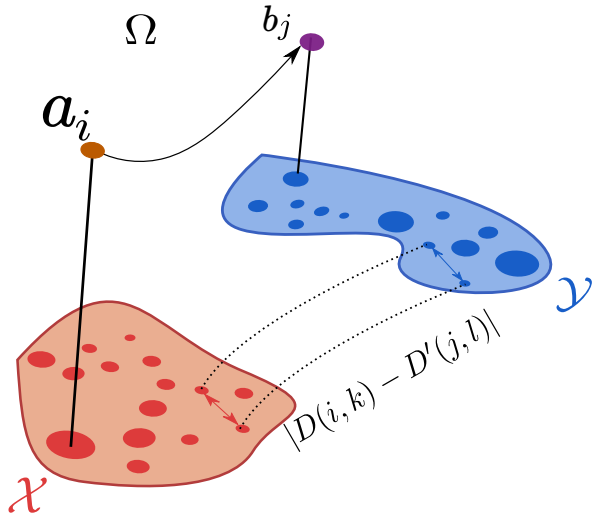
GW for discrete distributions [Memoli, 2011]

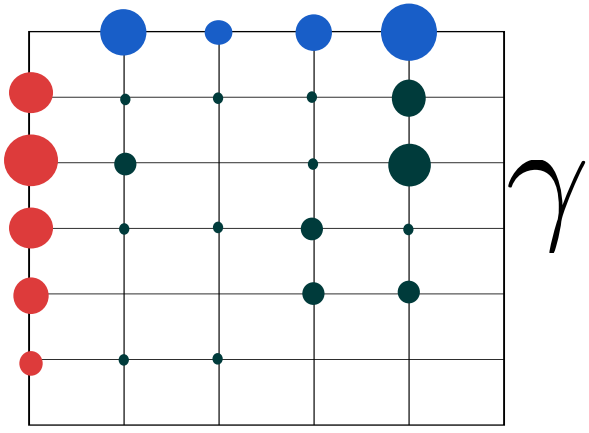
$$\mathcal{GW}_p^p(y_s, y_t)) = \min_{T \in \Pi} \sum_{i,j,k,l} |D_{i,k} - D'_{j,l}|^p T_{i,j}\, T_{k,l}$$

with $\Pi = \left\{ \mathbf{T} \in (\mathbb{R}^+)^{n_s \times n_t} \,|\, \mathbf{T}\mathbf{1}_{n_t} = \mathbf{a}, \mathbf{T}^T \mathbf{1}_{n_s} = \mathbf{b} \right\}$

- $\mathbf{D}$ and $\mathbf{D}'$ encode relationships between nodes (adjacency, shortest path).
- $\mathbf{a}$ and $\mathbf{b}$ are node weights of same total mass.
- Entropy regularized GW proposed in [Peyré et al., 2016].
- Fused GW interpolates between Wass. and GW [Vayer et al., 2018] with $C_{i,j}$ the cost between labels across graphs.

Gromov-Wasserstein and Fused Gromov-Wasserstein





FGW for discrete distributions [Vayer et al., 2018]

$$\mathcal{FGW}_p^p(y_s, y_t)) = \min_{T \in \Pi} \sum_{i,j,k,l} \left((1-\alpha)C_{i,j}^q + \alpha|D_{i,k} - D'_{j,l}|^q\right)^p T_{i,j}\, T_{k,l}$$

with $\Pi = \left\{\mathbf{T} \in (\mathbb{R}^+)^{n_s \times n_t} \,|\, \mathbf{T}\mathbf{1}_{n_t} = \mathbf{a}, \mathbf{T}^T\mathbf{1}_{n_s} = \mathbf{b}\right\}$

- $\mathbf{D}$ and $\mathbf{D}'$ encode relationships between nodes (adjacency, shortest path).
- $\mathbf{a}$ and $\mathbf{b}$ are node weights of same total mass.
- Entropy regularized GW proposed in [Peyré et al., 2016].
- Fused GW interpolates between Wass. and GW [Vayer et al., 2018] with $C_{i,j}$ the cost between labels across graphs.

Partially-Masked Fused Gromov-Wasserstein (PM-FGW)

Definition of PM-FGW

$$\text{PM-FGW}(\hat{y}, y) = \min_{\mathbf{T} \in \Pi_M} \mathcal{L}_{\mathbf{T}}(\hat{y}, y)$$

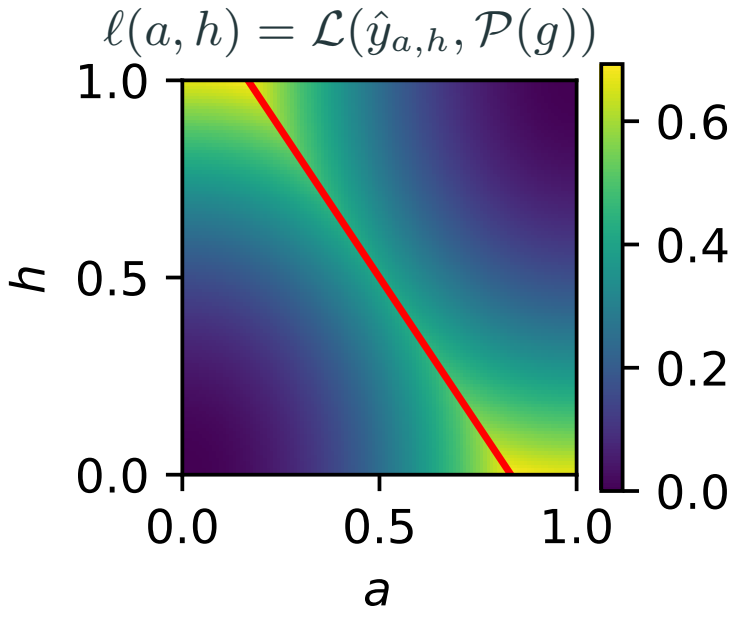
$$\text{with } \mathcal{L}_{\mathbf{P}}(\hat{y}, y) = \frac{\alpha_{\mathsf{h}}}{M} \sum_{i,j} T_{i,j} \ell_h(\hat{h}_i, h_j) \qquad \text{Padding loss}$$

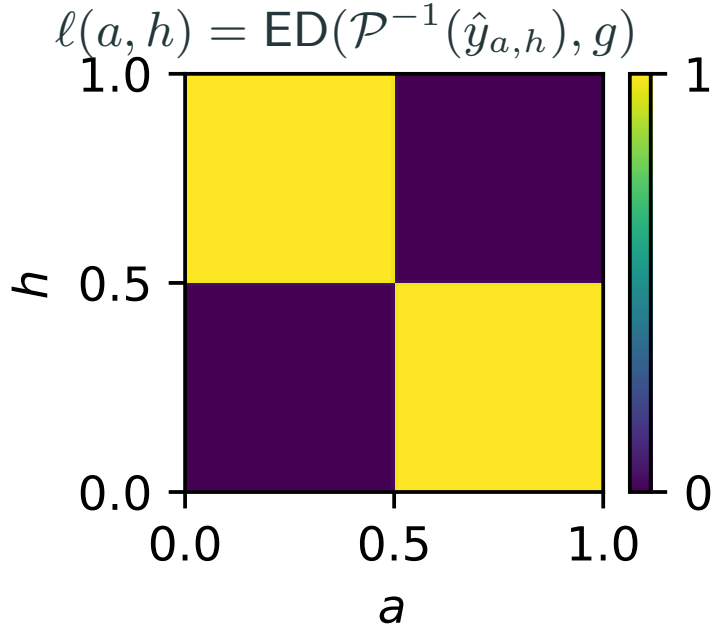
$$+ \frac{\alpha_{\mathsf{f}}}{m} \sum_{i,j} T_{i,j} \ell_f(\hat{\mathbf{f}}_i, \mathbf{f}_j) h_j \qquad \text{Feature loss}$$

$$+ \frac{\alpha_{\mathsf{A}}}{m^2} \sum_{i,j,k,l} T_{i,j} T_{k,l} \ell_A(\hat{A}_{i,k}, A_{j,l}) h_j h_l. \qquad \text{Structure loss}$$

- ℓ_h, ℓ_f and ℓ_A are loss functions for node, feature and adjacency matrix discrepancies (Kullback-Leibler when target discrete, Squared loss when continuous feature).
- α_{h}, α_{f} and α_{A} are hyperparameters on the simplex.
- Loss is highly asymmetric due to the right masking by $\mathbf{h}$.
- Can be solved by Conditional Gradient with $O(M^3 \log M)$ iteration.

Illustration of PM-FGW loss





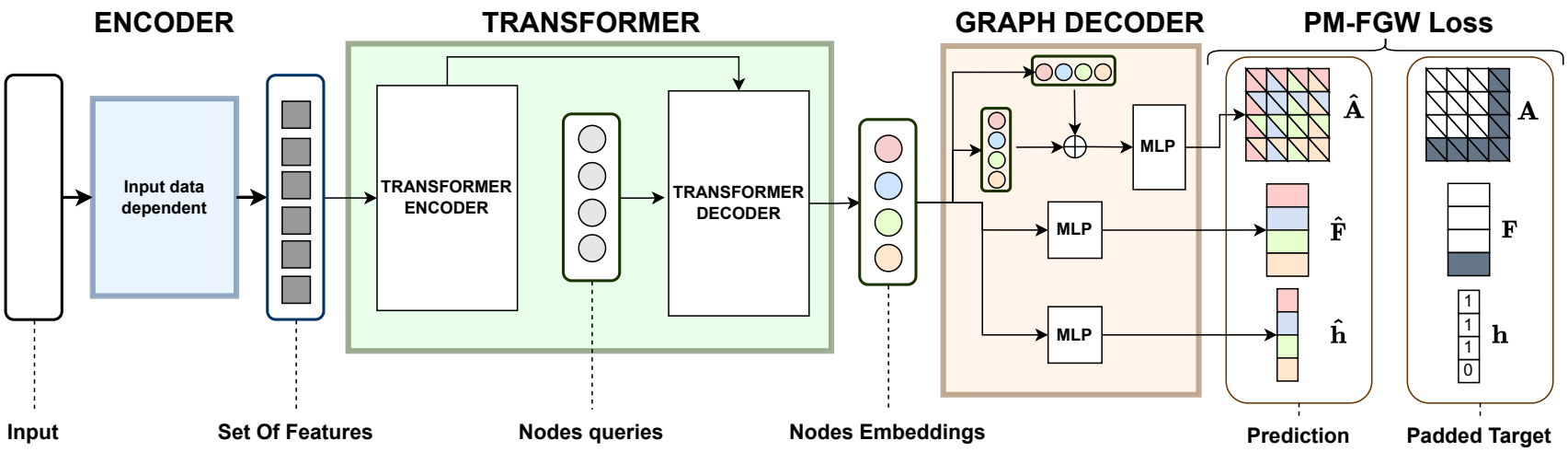
- The target graph is $g = (\mathbf{F}, \mathbf{A})$ with

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

- The prediction $\hat{y}_{a,h} = (\hat{\mathbf{h}}, \hat{\mathbf{F}}, \hat{\mathbf{A}})$ is

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

Neural network architecture



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

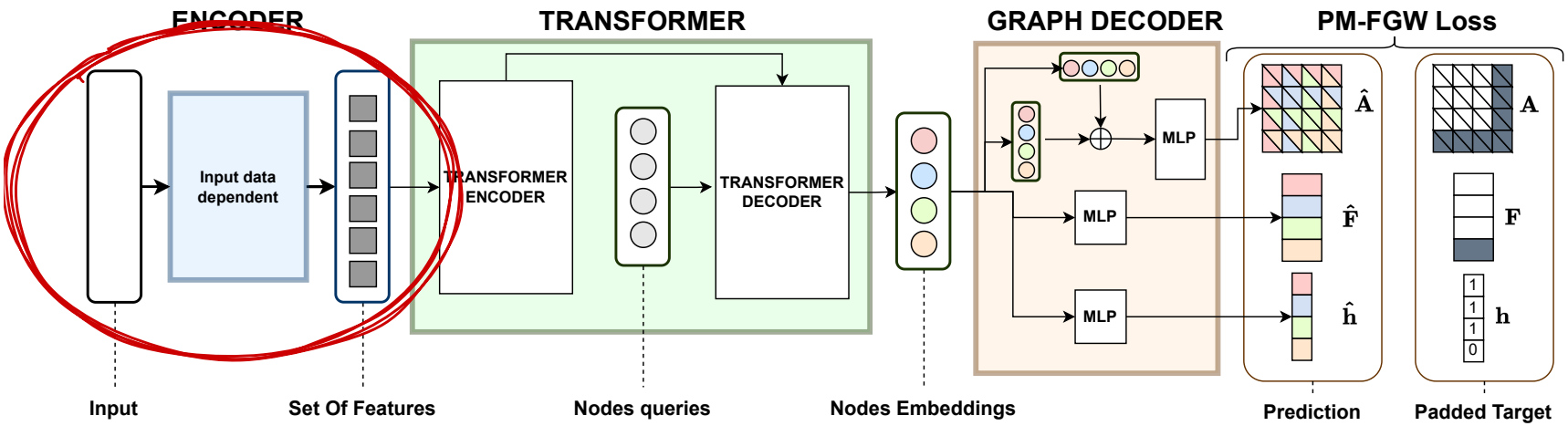
$$\hat{h}_i = \sigma(\mathrm{MLP}_m(\mathbf{z}_i)) \qquad \forall i \in \{1, \ldots, M\}$$

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

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

- Similar to Relationformer [Shit et al., 2022] but with symmetric adjacency matrix.

Neural network architecture : the encoder

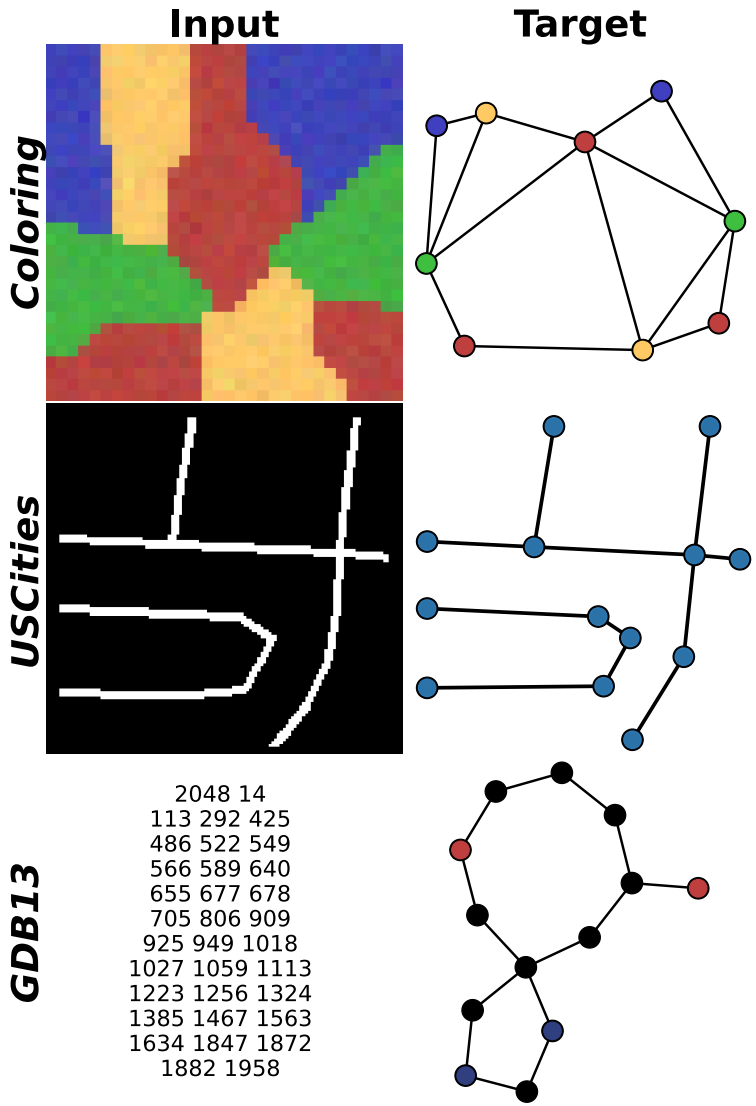


About the encoder

- Architecture adapts to input modality.
- Can leverage pretrained models.
- Must extract a **list of features** to avoid vector bottleneck.
- List of feature can be seen as a distribution [De Bie et al., 2019].
- Examples:
  - Text: token embeddings.
  - Image: CNN features (no global pooling).
  - Graph: GNN features (no global pooling).

Numerical experiments

Graph prediction experiments



Datasets

- Coloring
  - Input: 2D RGB images (100k train)
  - Output: 4-color graph (up to 20 nodes).
- Toulouse & USCities
  - Input: binary images (80k & 130k train)
  - Output: road network (up to 20 nodes).
- QM9 & GDB13
  - Input: Fingerprint (120k & 1300k train)
  - Output: molecule graph (up to 9 & 13 nodes).

Compared methods

- FGWBary (known size) [Brogat-Motte et al., 2022]
- Relationformer [Shit et al., 2022]
- Any2Graph (Ours)

FD : feature diffusion : $\mathbf{F} \mapsto [\mathbf{F}, \mathbf{AF}]$ [Barbe et al., 2020]

Prediction performances

Dataset	Model	Graph Level			Edge Level		Node Level Acc.	
		Edit Dist. ↓	GI Acc. ↑	PMFGW ↓	Prec. ↑	Rec. ↑	Node ↑	Size ↑
Coloring	FGWBary-NN*	6.73	1.00	0.91	75.19	84.99	77.58	n.a.
	FGWBary-ILE*	7.60	0.90	0.93	72.17	83.81	79.15	n.a.
	Relationformer	5.47	18.14	0.32	80.39	86.34	92.68	99.32
	Any2Graph	**0.20**	**85.20**	**0.03**	**99.15**	**99.37**	**99.95**	**99.50**
Toulouse	FGWBary-NN*	8.11	0.00	1.15	84.09	79.68	10.10	n.a.
	FGWBary-ILE*	9.00	0.00	1.21	72.52	56.30	1.62	n.a.
	Relationformer	**0.13**	93.28	**0.02**	99.25	99.24	99.25	98.30
	Any2Graph	**0.13**	**93.62**	**0.02**	**99.34**	**99.26**	**99.39**	**98.81**
USCities	Relationformer	2.09	55.00	0.13	**92.96**	87.98	95.18	**79.80**
	Any2Graph	**1.86**	**58.10**	**0.12**	92.91	**90.85**	**95.70**	78.95
QM9	FGWBary-NN*	5.55	1.00	0.96	87.81	70.78	78.62	n.a.
	FGWBary-ILE*	3.54	7.10	0.59	80.40	75.14	91.47	n.a.
	FGWBary-ILE* + FD	2.84	28.95	0.28	82.96	79.76	92.99	n.a.
	Relationformer	9.15	0.05	0.48	21.42	4.77	99.28	91.80
	Relationformer + FD	3.80	9.95	0.22	86.07	73.31	99.34	**96.0**
	Any2Graph	3.44	7.50	0.21	86.21	77.27	99.26	93.65
	Any2Graph + FD	**2.13**	**29.85**	**0.14**	**90.19**	**88.08**	**99.77**	95.45
GDB13	Relationformer	11.40	0.00	0.43	81.96	31.49	97.77	97.45
	Relationformer + FD	8.83	0.01	0.29	84.14	55.89	97.57	**98.65**
	Any2Graph	7.45	0.05	0.22	87.20	60.41	99.41	96.15
	Any2Graph + FD	**3.63**	**16.25**	**0.11**	**90.83**	**84.86**	**99.80**	98.15

- Any2Graph is SOTA for graph prediction on all datasets.
- Feature diffusion helps on molecules [Brogat-Motte et al., 2022].

Sensitivity analysis

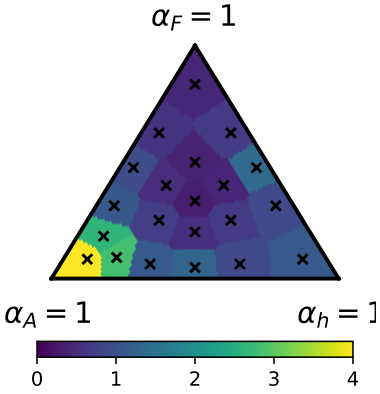


Figure 1: Edit dist.

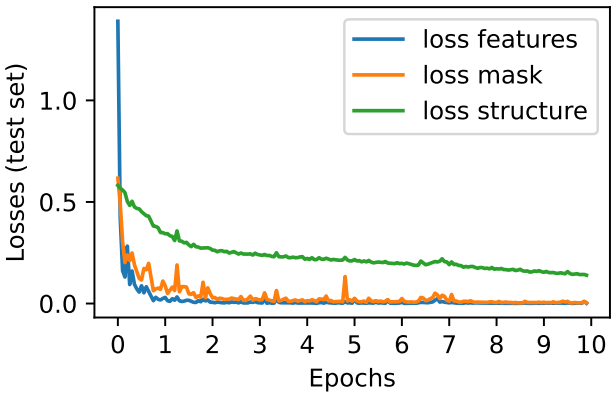


Figure 2: $\alpha = [1, 1, 1]$.

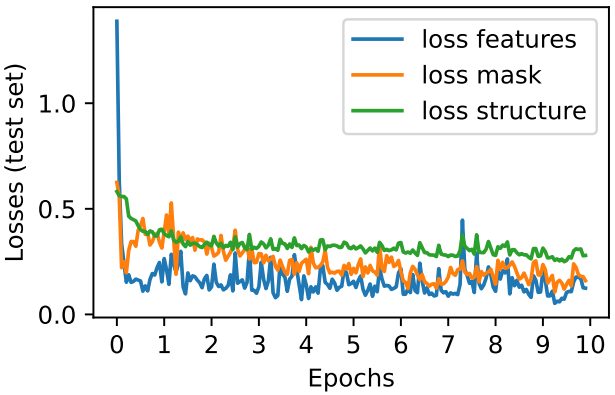


Figure 3: $\alpha = [10, 1, 1]$.

Results on Coloring dataset

- Robustness to α (Fig 1.).
- Mask and features learned first (Fig 2.).
- Failure for large α_A (Fig 1. and 3.).
- For increasing M edit distance is stable.
- Effective active nodes number do not increase with M (Fig 4.)

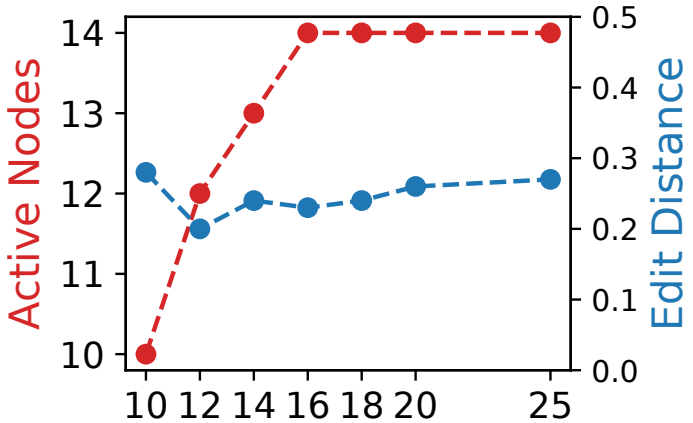


Figure 4: Effect of M.

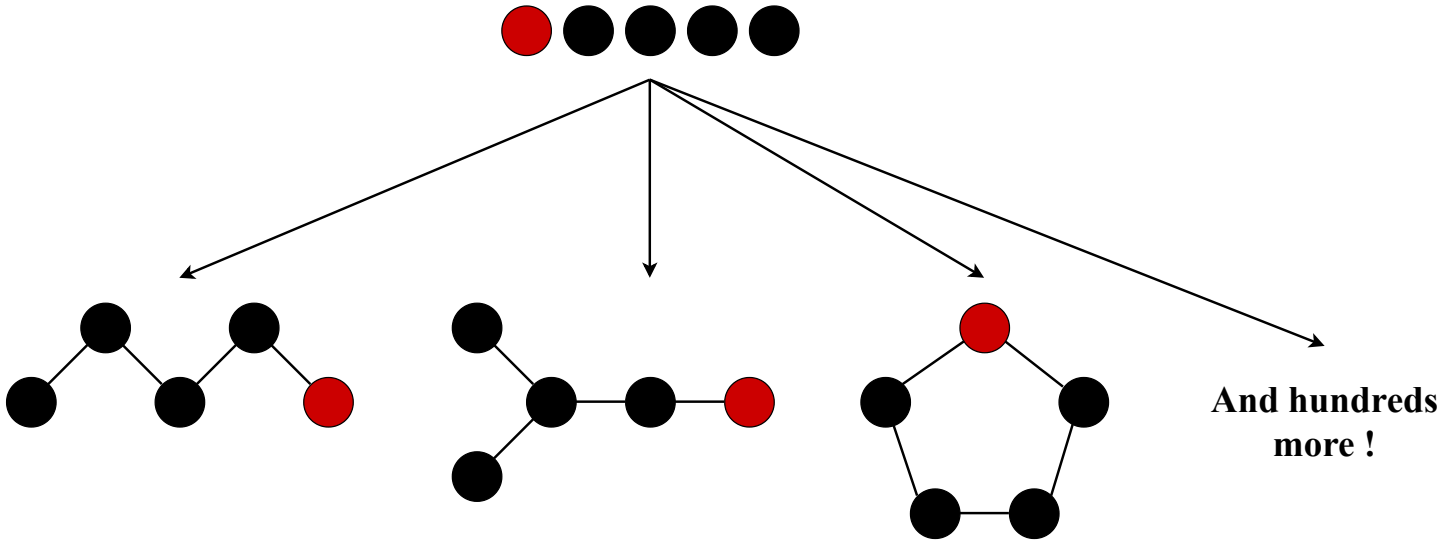
Computational Performances

Method	Training	Inference
FGWBary-ILE (K=25)	n.a.	1
FGWBary-NN (K=10)	10	10
Relationformer	9k	20k
PM-FGW (ours)	3k	20k

Table 1: Computational Performances (In graph per second)

- FGWBary-ILE uses thresholding of weights yet still slow.
- FGWBary-NN with learned dictionary is a little faster.
- Relationformer and Any2Graph are fastest at inference.
- Any2Graph is slightly slower that Relationforme during training.

Why diffusion of feature on molecules?

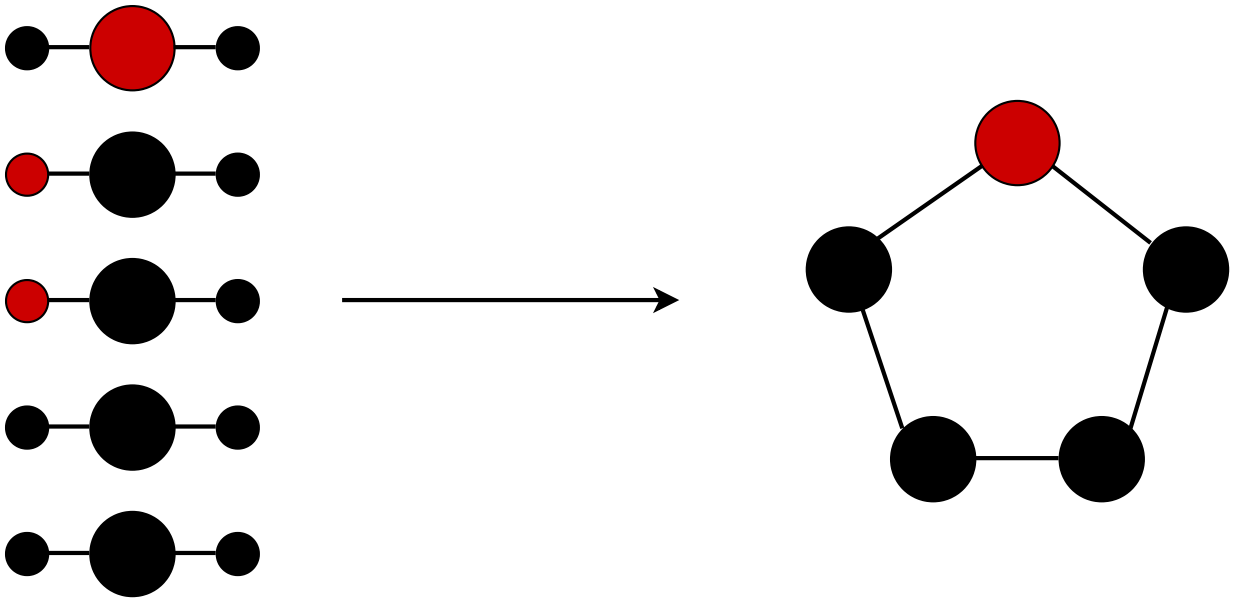


Diffusion of features [Barbe et al., 2020]

$$\mathbf{F} \mapsto [\mathbf{F}, \mathbf{AF}]$$

- On some graphs, nodes features are not enough to predict the structure.
- Diffusion of features encode labels + labels of neighbors.
- Helps the model for structure identification with more information in the linear term for molecule graphs.

Why diffusion of feature on molecules?



Diffusion of features [Barbe et al., 2020]

$$\mathbf{F} \mapsto [\mathbf{F}, \mathbf{AF}]$$

- On some graphs, nodes features are not enough to predict the structure.
- Diffusion of features encode labels + labels of neighbors.
- Helps the model for structure identification with more information in the linear term for molecule graphs.

Why diffusion of feature on molecules?

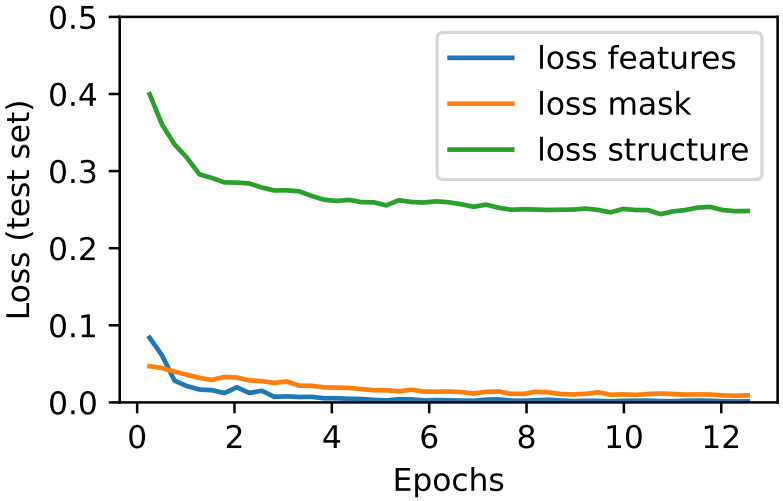


Figure 5: Without Feature Diffusion.

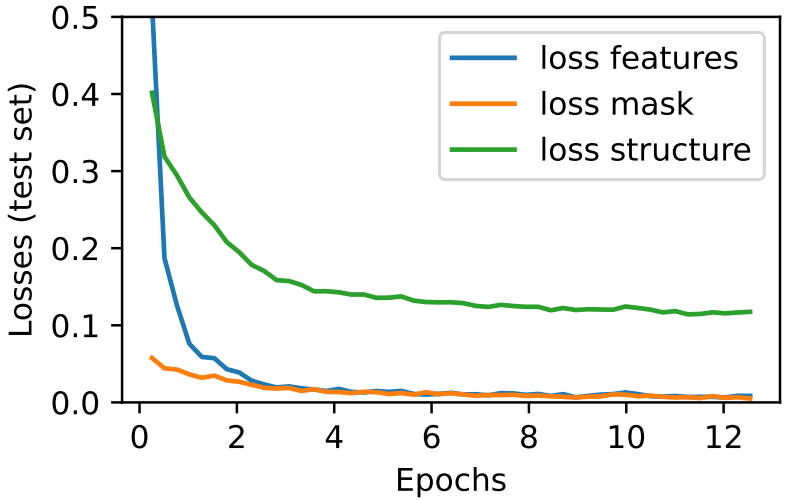


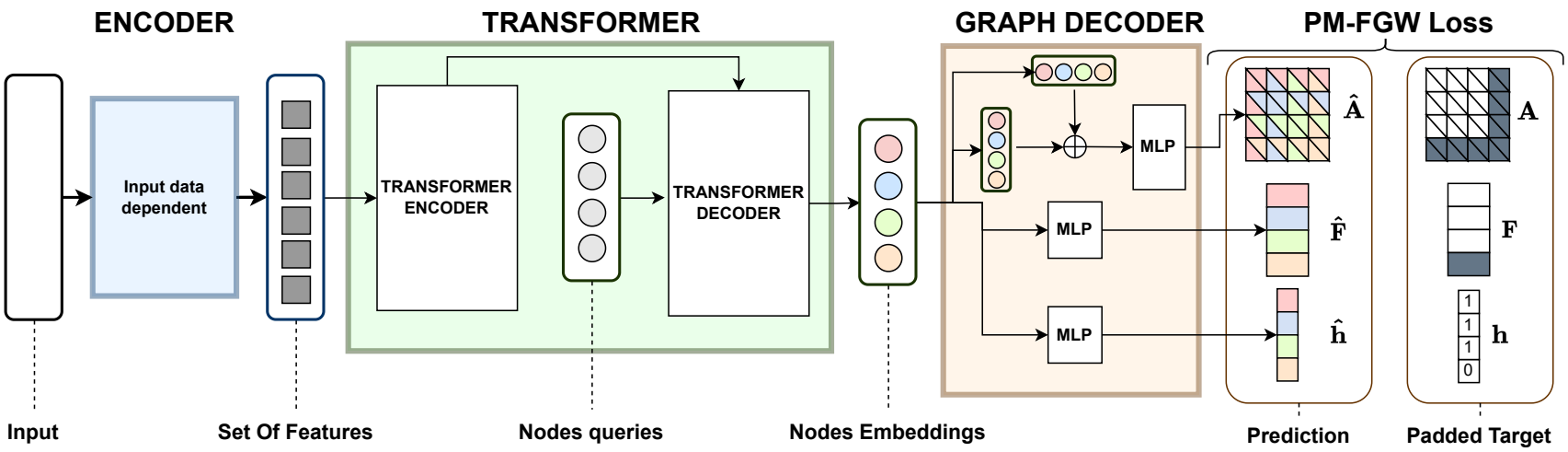
Figure 6: With Feature Diffusion.

Diffusion of features [Barbe et al., 2020]

$$\mathbf{F} \mapsto [\mathbf{F}, \mathbf{AF}]$$

- On some graphs, nodes features are not enough to predict the structure.
- Diffusion of features encode labels + labels of neighbors.
- Helps the model for structure identification with more information in the linear term for molecule graphs.

Conclusion



Any2Graph [Krzakala et al., 2024]

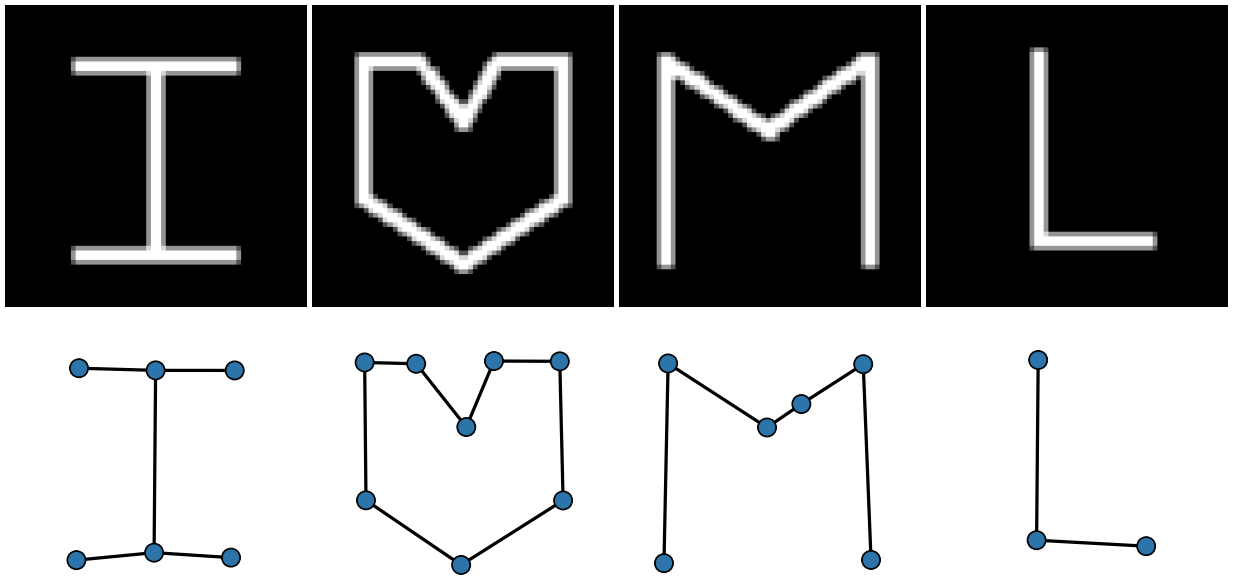
- Generic framework for supervised graph prediction from any input.
- End-to-end trainable with a neural network with OT loss.
- Good performance on variety of (input/output) datasets.

Limits and future works

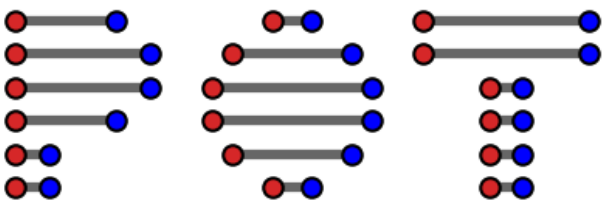
- Only tested on relatively small graphs ($M \leq 20$).
- Use higher order relationship matrices (Power of Laplacian L^k).
- Investigate graph auto-encoders.

Thank you for your $\texttt{softmax}(QK^T)V$!

Any2Graph model trained on Toulouse :



POT Python Optimal Transport library



- `https://pot.readthedocs.io/en/stable/`
- Solvers for OT problems.
- Backends in Numpy/Pytorch.

References i

Barbe, A., Sebban, M., Gonçalves, P., Borgnat, P., and Gribonval, R. (2020).

Graph diffusion wasserstein distances.

In *ECML PKDD 2020-European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 1–16.

Brogat-Motte, L., Flamary, R., Brouard, C., Rousu, J., and d'Alché Buc, F. (2022).

Learning to predict graphs with fused gromov-wasserstein barycenters.

In *International Conference in Machine Learning (ICML)*.

Brouard, C., Shen, H., Dührkop, K., d'Alché-Buc, F., Böcker, S., and Rousu, J. (2016).

Fast metabolite identification with input output kernel regression.

Bioinformatics, 32(12):i28–i36.

References ii

Chapel, L., Alaya, M. Z., and Gasso, G. (2020).

Partial optimal tranport with applications on positive-unlabeled learning.

Advances in Neural Information Processing Systems, 33:2903–2913.

De Bie, G., Peyré, G., and Cuturi, M. (2019).

Stochastic deep networks.

In *International Conference on Machine Learning*, pages 1556–1565. PMLR.

Dozat, T. and Manning, C. D. (2017).

Deep biaffine attention for neural dependency parsing.

In *International Conference on Learning Representations, ICLR*. OpenReview.net.

References iii

El Ahmad, T., Brogat-Motte, L., Laforgue, P., and d'Alché Buc, F. (2024).

Sketch in, sketch out: Accelerating both learning and inference for structured prediction with kernels.

In *International Conference on Artificial Intelligence and Statistics*, pages 109–117. PMLR.

Frogner, C., Zhang, C., Mobahi, H., Araya, M., and Poggio, T. A. (2015).

Learning with a wasserstein loss.

In *Advances in Neural Information Processing Systems*, pages 2053–2061.

Krzakala, P., Yang, J., Flamary, R., d'Alché Buc, F., Laclau, C., and Labeau, M. (2024).

Any2graph: Deep end-to-end supervised graph prediction with an optimal transport loss.


Melnyk, I., Dognin, P., and Das, P. (2022).

Knowledge graph generation from text.

In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1610–1622.


Memoli, F. (2011).

Gromov wasserstein distances and the metric approach to object matching.

Foundations of Computational Mathematics, pages 1–71.


Peyré, G., Cuturi, M., and Solomon, J. (2016).

Gromov-wasserstein averaging of kernel and distance matrices.

In *ICML*, pages 2664–2672.

References v

Shit, S., Koner, R., Wittmann, B., Paetzold, J., Ezhov, I., Li, H., Pan, J., Sharifzadeh, S., Kaissis, G., Tresp, V., et al. (2022).

Relationformer: A unified framework for image-to-graph generation.

In *European Conference on Computer Vision*, pages 422–439. Springer.

Thual, A., Tran, H., Zemskova, T., Courty, N., Flamary, R., Dehaene, S., and Thirion, B. (2022).

Aligning individual brains with fused unbalanced gromov-wasserstein.

In *Neural Information Processing Systems (NeurIPS)*.

Vayer, T., Chapel, L., Flamary, R., Tavenard, R., and Courty, N. (2018).

Fused gromov-wasserstein distance for structured objects: theoretical foundations and mathematical properties.

Vayer, T., Chapel, L., Flamary, R., Tavenard, R., and Courty, N. (2020).

Fused gromov-wasserstein distance for structured objects.

Algorithms, 13 (9):212.

Why not [X] OT loss?

Fused Gromov-Wasserstein [Vayer et al., 2020]

- $\mathbf{h}$ and $\hat{\mathbf{h}}$ can be normalized to sum to 1 and used as marginal weights.
- But : (Sub)-gradients on the masses are not very numerically stable.
- For multi-label learning use unbalanced OT [Frogner et al., 2015].

Unbalanced FGW [Thual et al., 2022]

- Works out of the bow with $\mathbf{h}$ and $\hat{\mathbf{h}}$ as marginal weights.
- But: many parameters to tune (did not manage to make it work).

Partial (F)GW [Chapel et al., 2020]

- Can force to move only the actual number of nodes.
- But: No weights on not active nodes so node classifier cannot be learned.

Solving for PM-FGW

Conditional Gradient solver

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

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

- Solve iteratively linearized version of the loss with $\mathbf{C}^{(k)} = \mathbf{U} + \mathbf{L} \otimes \mathbf{T}^{(k)}$.
- Each step requires :
  - Solving a standard OT problem $O(M^3 \log M)$.
  - Selecting a linesearch step.

Computational complexity

- $O(M^3)$ factorization for PM-FGW (similar to [Peyré et al., 2016]).
- Closed form for the linesearch at each iteration.
- Typical convergence is 5-10 iteration in experiments.