Interpretable Lightweight Transformer via Unrolling of Learned Graph Smoothness Priors

Tam Thuc Do, Parham Eftekhar, Seyed Alireza Hosseini, Gene Cheung, Philip Chou

—

Final Report Presentation

Machine Learning with Graphs – IPP 2025

Kenneth Browder - Fabien Lagnieu

Institut Polytechnique de Paris

April 1, 2025

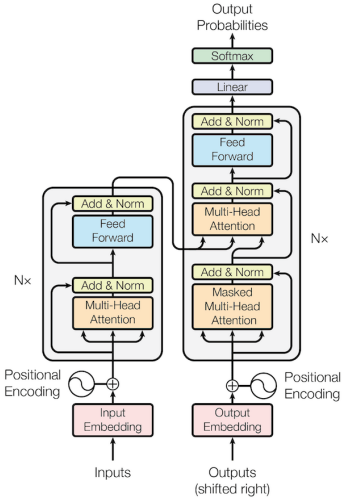
Agenda

1. Context & Motivation
2. State of the Art
3. Main Contributions
4. Methodology
5. Paper Experimentation
6. Critical Analysis
7. Conclusion & Questions

1. Context & Motivation

- **Transformers** are state-of-the-art in NLP & CV but suffer from:
  - High computational cost
  - Lack of interpretability

- The paper proposes an **interpretable and lightweight** model:
  - Combining **Graph Signal Processing (GSP)**
  - And **unrolled optimization**



Transformer architecture ([Vaswani et al., 2017]).

2. State of the Art

- **Graph Neural Networks (GNNs)**
  Learn node and graph-level representations through message passing.
  - Powerful but complex and hard to interpret.
  - Struggle with scalability and over-smoothing in deep architectures.

- **Graph Signal Processing (GSP)**
  Provides well-defined priors for graph data.
  - **Graph Laplacian Regularizer (GLR)**: enforces smoothness across the graph. [Ortega et al., 2018]
  - **Graph Total Variation (GTV)**: preserves edges and discontinuities. [Cheung et al., 2018]

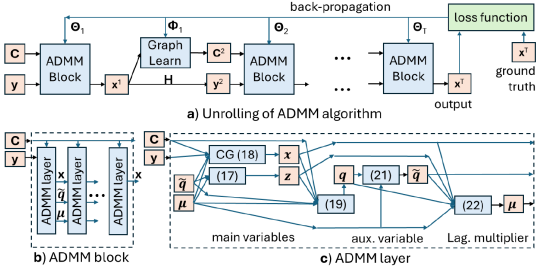
- **Optimization Unrolling (brief)**
  Converts iterative optimization algorithms into neural network layers.
  - Improves interpretability and model efficiency.

3. Main Contributions

- **Transformer-like architecture via optimization unrolling**
  - Each layer corresponds to an iteration of an optimization algorithm.
  - Brings interpretability to layer operations and reduces computational cost.
- **Mahalanobis distance-based graph learning**
  - Learns a similarity graph by adapting the Mahalanobis distance. [Hu et al., 2020]
  - Ensures symmetric adjacency and interpretable edge weights.
- **Lightweight, interpretable, and efficient for graph signal interpolation**
  - Achieves state-of-the-art performance with significantly fewer parameters.
  - Demonstrates robustness and generalization across tasks.

4. Methodology Overview

- **Alternating graph learning and signal interpolation**
  - At each layer, the similarity graph and node features are jointly refined.
- **Unrolled optimization replacing attention mechanisms**
  - Conjugate Gradient (GLR) and ADMM (GTV) unfold as layers.
  - Each iteration is explicitly controlled and interpretable.



GTV-based unrolled algorithm [Do et al., 2024]

4.1 Graph Learning via Mahalanobis Distance

- Learn a similarity graph $\rightarrow$ pairwise distances.
- Mahalanobis distance: [Hu et al., 2020]

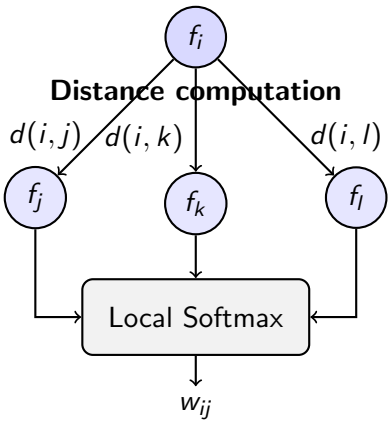
$$d(i,j) = (\mathbf{f}_i - \mathbf{f}_j)^\top M(\mathbf{f}_i - \mathbf{f}_j)$$

- M is a learnable, positive semi-definite matrix.
- Output edge weights:

$$w_{ij} = \exp(-d(i,j))$$

- Local Softmax normalization:

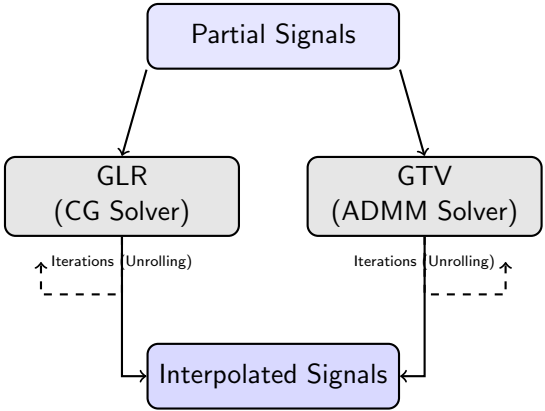
$$w_{ij} = \frac{\exp(-d(i,j))}{\sum_{j' \in \mathcal{N}(i)} \exp(-d(i,j'))}$$



Learned edge weights w_{ij} based on Mahalanobis distance and local softmax normalization.

4.2 Signal Interpolation via Unrolled Optimization

- After graph learning, node signals are interpolated: **two methods**
  - **GLR**: Conjugate Gradient (CG) [Shewchuk, 1994]
  - **GTV**: Alternating Direction Method of Multipliers (ADMM) [Wang and Shroff, 2017]
- **Unrolled optimization**: Each layer corresponds to an iteration, enabling interpretability and convergence control.
- **Output**: High-quality, edge-preserving interpolation on graph signals.



GLR and GTV unrolled optimization algorithms

5.1 Experimental Setup

- **Training Dataset: DIV2K**
  - 800 HR images for training, 100 for validation
  - Extracted 64×64 patches, using only 1% - 4% for training
- **Test Datasets**
  - McM [Zhang et al., 2011]
  - Kodak [Eastman Kodak, 1993]
  - Urban100 [Huang et al., 2015].
- **Tasks**
  - **Demosaicking**: reconstruct RGB images from Bayer pattern
  - **Image Interpolation**: reconstruct HR images from LR inputs
- **Evaluation Metrics**
  - Peak Signal-to-Noise Ratio (PSNR)
  - Structural Similarity Index Measure (SSIM) [Wang et al., 2004]

5.2 Experimental Results

Demosaicking

Method	Params#	McM		Kodak		Urban100	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bilinear	-	29.71	0.9304	28.22	0.8898	24.18	0.8727
RST-B [46]	931763	34.85	0.9543	38.75	0.9857	32.82	0.973
RST-S [46]	3162211	35.84	0.961	**39.81**	**0.9876**	33.87	0.9776
Menon [47]	-	32.68	0.9305	38.00	0.9819	31.87	0.966
Malvar [48]	-	32.79	0.9357	34.17	0.9684	29.00	0.9482
iGLR	-	29.39	0.8954	27.50	0.8487	23.13	0.8406
iGTV	-	30.43	0.8902	28.66	0.8422	24.91	0.8114
uGLR	323410	36.09	0.9650	37.88	0.9821	33.60	0.9772
uGTV	323435	**36.59**	**0.9665**	39.11	0.9855	**34.01**	**0.9792**

- **uGTV** achieves top PSNR and SSIM on Urban100 & McM.
- Nearly matches RST-S on Kodak with far fewer parameters.

Image Interpolation

Method	Params#	McM		Kodak		Urban100	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	-	29.01	0.8922	26.75	0.8299	22.95	0.7911
MAIN [49]	10942977	32.72	0.822	28.23	0.728	25.46	0.806
SwinIR-lightweight [50]	904744	32.24	0.9354	28.62	0.8794	25.08	0.8553
iGLR	-	28.53	0.8537	26.71	0.8005	22.87	0.7549
iGTV	-	30.41	0.887	28.05	0.832	24.26	0.7855
uGLR	319090	33.31	0.9431	**29.10**	0.8870	25.94	0.8777
uGTV	319115	**33.36**	**0.9445**	29.08	**0.8888**	**26.12**	**0.8901**

- **uGTV** consistently outperforms SOTA models across datasets.
- Maintains efficiency with only 300k parameters versus multi-million parameter baselines.

6.1 Experiment 1: Reproducing the Paper Setup

Objective

- Validate uGLR on Div2K dataset
- Upscaling: from 32×32 to 64×64
- One pixel out of four is retained

Methodology

- No official code found → Full reimplementation in PyTorch
- CG and ADMM coded from scratch
- Sparse ops via PyTorch only
- Hardware: A100 (40 GB VRAM)

Model Parameters

- 64×64 images ⇒ 4096 nodes
- 5×5 neighborhood (∼24 edges/node)
- ADMM block: 5 layers × 10 CG steps
- CNN: 4 layers, 48 features → 4×12 heads

Discussion

- $\sim 10^5$ edges ⇒ GPU overflow
- Sparse ops needed but PyTorch support limited
- Batched design complicated sparse integration
- Reproduction failed due to memory limits

```
OutOfMemoryError                          Traceback (most recent call last)
Cell In[20], line 19
    17 optimizer.zero_grad()
    18 print("Computing backward")
---> 19 loss.backward()
    20 print("Computing backward")
    21 optimizer.step()

File ~/jp/.venv/lib/python3.12/site-packages/torch/_tensor.py:581, in Tensor.backward(self, gradient, retain_graph, create_graph, in
   571 if has_torch_function_unary(self):
   572     return handle_torch_function(
   573         Tensor.backward,
   574         (self,),
  (...)
   579         inputs=inputs,
   580     )
--> 581 torch.autograd.backward(
   582     self, gradient, retain_graph, create_graph, inputs=inputs
   583 )

File ~/jp/.venv/lib/python3.12/site-packages/torch/autograd/__init__.py:347, in backward(tensors, grad_tensors, retain_graph, create
   342     retain_graph = create_graph
   346 # The reason we repeat the same comment below is that
   345 # some Python versions print out the first line of a multi-line function
...
    32 # else:
    33 #     grad_M2 = grad_M2.to_dense()
    30 return grad_M1, grad_M2

OutOfMemoryError: CUDA out of memory. Tried to allocate 6.71 GiB. GPU 2 has a total capacity of 39.50 GiB of which 5.42 GiB is free.
```

CUDA Out-of-Memory on A100 (loss.backward step)

6.2 Experiment 2: Scaled-Down Model on 32×32 Images

Objective

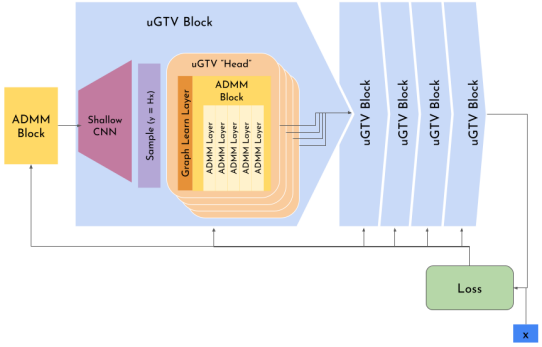
- Train a reduced uGLR model for image upscaling
- Input: 32×32 image, Output: 64×64

Methodology

- Same PyTorch implementation as before
- Lowered model complexity for memory feasibility
- Fewer features, fewer neighbors per node

Remarks

- Successfully trained on constrained hardware
- Visual inspection confirms signal reconstruction



Reduced uGLR architecture used in additional experiment

7. Critical Analysis

Strengths

- **Interpretability**: Each layer has a clear mathematical meaning, making the model transparent and easier to analyze.
- **Efficiency**: The architecture drastically reduces the number of parameters.
- **Strong Results**: Achieves state-of-the-art performance on demosaicking and image interpolation tasks, with robust generalization across datasets.

Limitations & Future Work

- **Scalability**: The model's performance and efficiency on very large graphs (e.g., social networks or biological networks) remain unexplored.
- **Task Diversity**: The method was only evaluated on image-based interpolation; its applicability to other tasks (e.g., node classification, link prediction) is uncertain.

8. Conclusion & Questions

Key Takeaways

- Introduced an interpretable, lightweight transformer-like model for graph signal interpolation.
- Combines Graph Signal Processing with unrolled optimization (GLR / GTV priors).
- Achieves state-of-the-art results with significantly fewer parameters.
- Demonstrates robustness and efficiency on image-based tasks.

Thank You!

Questions?

References

Cheung, G., Leus, G., and Ortega, A. (2018).
Graph spectral image processing.
In *Proceedings of the IEEE*, volume 106, pages 907–930.

Do, T. T., Eftekhar, P., Hosseini, S. A., Cheung, G., and Chou, P. A. (2024).
Interpretable lightweight transformer via unrolling of learned graph smoothness priors.
In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS 2024)*, Vancouver, Canada. PMLR.

Eastman Kodak (1993).
Kodak lossless true color image suite (photocd pcd0992).
`http://r0k.us/graphics/kodak`.

Hu, W., Gao, X., Cheung, G., and Guo, Z. (2020).
Feature graph learning for 3d point cloud denoising.
IEEE Transactions on Signal Processing, 68:2841–2856.

Huang, J.-B., Singh, A., and Ahuja, N. (2015).
Single image super-resolution from transformed self-exemplars.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5197–5206.

Ortega, A., Frossard, P., Kovacevic, J., Moura, J. M., and Vandergheynst, P. (2018).
Graph signal processing: Overview, challenges, and applications.
Proceedings of the IEEE, 106(5):808–828.

Shewchuk, J. R. (1994).
An introduction to the conjugate gradient method without the agonizing pain.
Technical report, Pittsburgh, PA, USA: Carnegie Mellon University.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Łukasz Kaiser, and Polosukhin, I. (2017).
Attention is all you need.
In *Advances in Neural Information Processing Systems (NeurIPS)*.

Wang, S. and Shroff, N. (2017).
A new alternating direction method for linear programming.
In *Advances in Neural Information Processing Systems*, volume 30.

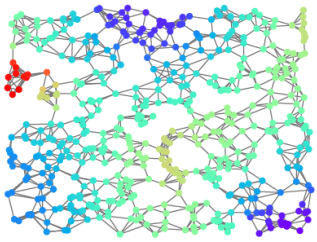
Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004).
Image quality assessment: From error visibility to structural similarity.
IEEE Transactions on Image Processing, 13(4):600–612.

Zhang, L., Wu, X., Buades, A., and Li, X. (2011).
Color demosaicking by local directional interpolation and nonlocal adaptive thresholding.
Journal of Electronic Imaging, 20(2):023016–023016.

Key Concepts: Graph Signal Processing (GSP)

- **GSP** extends traditional signal processing to irregular domains represented as graphs.
- A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of nodes $\mathcal{V}$ and edges $\mathcal{E}$, typically described by an adjacency matrix A or Laplacian L.
- **Graph signals**: data defined on graph nodes (e.g., image pixels, sensors in a network).
- **Key operations**:
  - **Graph Fourier Transform (GFT)**: analyzes signals in the graph spectral domain.
  - **Graph Filtering**: smooths or enhances graph signals by manipulating graph frequencies.
  - **Graph Laplacian**: central operator for defining smoothness and spectral properties.

- **In this paper**:
  - GSP provides priors for signal smoothness over learned graphs.
  - It ensures edge-aware filtering, improving interpolation tasks.



Graph signals on nodes, smoothness across edges.

Key Concepts: Graph Laplacian Regularizer (GLR)

- **Objective:** Encourage smoothness of signals across the graph by penalizing differences between neighboring nodes.
- **Smoothness assumption:** Connected nodes should have similar signal values.
- **GLR formula:**

$$\text{GLR}(\mathbf{x}) = \mathbf{x}^\top L\mathbf{x} = \sum_{i,j} w_{ij}(x_i - x_j)^2$$

  - L is the **graph Laplacian**: $L = D - W$
  - W is the adjacency matrix (weights w_{ij})
  - D is the degree matrix
- **In this paper:** GLR is unrolled via Conjugate Gradient (CG) optimization to refine signals layer by layer.

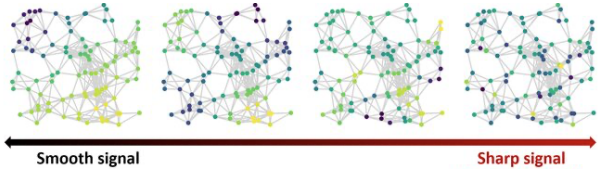


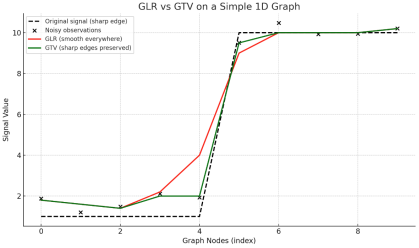
Illustration of graph signals with varying levels of smoothness.

Key Concepts: Graph Total Variation (GTV)

- **Objective:** Promote piecewise-smooth signals while preserving sharp edges or discontinuities on the graph.
- **Key Idea:** Unlike GLR, GTV tolerates sharp signal changes.
- **Mathematical expression:**

$$\mathrm{GTV}(\mathbf{x}) = \sum_{i,j} w_{ij} |x_i - x_j|$$

- **In this paper:**
  - Solved via ADMM, unrolled as layers.
  - Enables edge-preserving interpolation.
- **Use cases:** Useful in tasks requiring edge preservation, such as image denoising, segmentation, and interpolation.



Difference GLR/GTV: GTV preserves edges.

Key Concepts: Mahalanobis Distance in Graph Learning

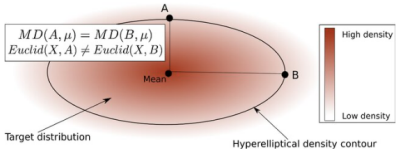
- **Purpose:** Measure similarity between nodes by considering feature correlations.
- **Mahalanobis Distance (MD):**

$$MD(\mathbf{x}, \mu) = \sqrt{(\mathbf{x} - \mu)^\top S^{-1} (\mathbf{x} - \mu)}$$

  - μ is the mean of the distribution.
  - S is the covariance matrix (captures feature correlations).
- **Why MD instead of Euclidean?**
  - MD accounts for feature scaling and correlations.
  - More appropriate for high-dimensional, correlated data.

In this paper:

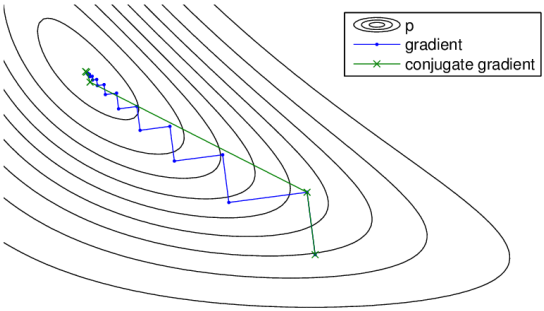
- Mahalanobis distance is **learned** via a matrix M, which is positive semi-definite.
- It defines adaptive edge weights in the similarity graph as $w_{ij} = \exp(-d(i,j))$, where $d(i,j) = (\mathbf{f}_i - \mathbf{f}_j)^\top M (\mathbf{f}_i - \mathbf{f}_j)$.
- This ensures the learned graph structure reflects the underlying data geometry.



Mahalanobis distance reflects the data distribution geometry.

Key Concepts: Conjugate Gradient (CG)

- **Purpose:** Solve linear systems of the form $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ is symmetric and positive-definite.
- **Advantages:**
  - Efficient for large, sparse systems.
  - No need to compute $\mathbf{A}^{-1}$.
  - Faster convergence than basic gradient descent.
- **In this paper:**
  - CG solves the Graph Laplacian Regularizer (GLR) problem.
  - It solves $\mathbf{Lx} = \mathbf{b}$, where $\mathbf{L}$ is the graph Laplacian matrix.
- **Interpretability:** Each CG iteration corresponds to a layer in the network. This makes the optimization process interpretable and controllable.



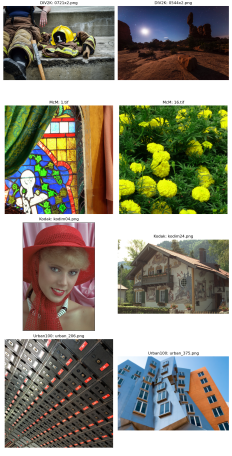
Gradient descent (blue) and conjugate gradient (green).

Key Concepts: ADMM (Alternating Direction Method of Multipliers)

- **Purpose:** Solve complex optimization problems by splitting them into simpler subproblems, updated alternately.
- **How it works:** ADMM addresses problems of the form: minimize $f(\mathbf{x}) + g(\mathbf{z})$ subject to $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}$. It alternates updates for $\mathbf{x}$, $\mathbf{z}$, and the dual variable.
- **Why ADMM?**
  - Handles non-smooth objectives like Total Variation (TV).
  - Decomposes large problems into smaller, tractable steps.
  - Efficient convergence for structured problems.
- **In this paper:** ADMM solves the **Graph Total Variation (GTV)** problem for graph signal interpolation.
  - GTV preserves sharp edges by penalizing absolute differences across edges.
  - ADMM efficiently manages the non-smooth TV regularizer.
  - Each ADMM iteration is unrolled as a network layer, ensuring interpretability.

Key Concepts: Datasets Used in Experiments

- **Purpose:** Evaluate model performance on **demosaicking** and **image interpolation** tasks.
- **Training Dataset: DIV2K**
  - 800 high-resolution images for training, 100 for validation.
  - Rich in detail and diversity.
  - Patches: 64×64, with sparse sampling (1–4% of pixels).
- **Test Datasets:**
  - **McM** – Color demosaicking benchmark.
  - **Kodak** – 24 high-quality natural images.
  - **Urban100** – Urban scenes with repetitive structures.
- **In this paper:**
  - Training on DIV2K.
  - Testing on McM, Kodak, Urban100.
  - Tasks: demosaicking and interpolation.

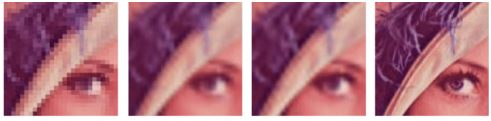


Sample images from DIV2K, McM, Kodak, Urban100 datasets.

Key Concepts: Task Definitions

Image Interpolation

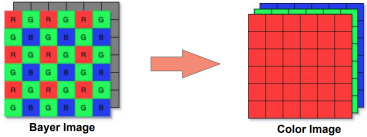
- **Purpose:** Restore missing pixels in an image from sparse or degraded inputs.
- **Dataset:** DIV2K, Urban100, Kodak
- **Challenge:** Maintain sharp edges and textures while recovering fine details.
- **In this paper:** Graph-based priors guide smoothness and edge preservation.



Example of image interpolation at different levels of quality.

Demosaicking

- **Purpose:** Reconstruct full RGB images from raw Bayer-patterned inputs.
- **Dataset:** McM, Kodak
- **Challenge:** Remove color artifacts and preserve fine structures.
- **In this paper:** Unrolled optimization ensures faithful color recovery.



From Bayer image to reconstructed color image.

Key Concepts: Evaluation Metrics

PSNR (Peak Signal-to-Noise Ratio)

- Measures pixel-level fidelity between the reconstructed and reference images.
- Formula:

$$PSNR = 10 \log_{10} \left(\frac{MAX_I^2}{MSE} \right)$$

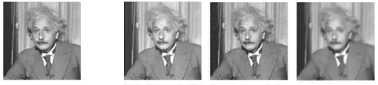
- MAX_I: maximum pixel value (e.g., 255)
- Higher PSNR $\rightarrow$ Less distortion
- Common in image compression and reconstruction
- **Not reflecting human perception**
- Insensitive to structural or perceptual distortions

SSIM (Structural Similarity Index)

- Evaluates perceptual similarity: Luminance, Contrast & Structure
- Formula (simplified):

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

- Values in $[0, 1]$
- Closer to 1 $\rightarrow$ More similarity



Images with the same PSNR but decreasing SSIM values show visible perceptual degradation.

Related Work: Feedback-based Graph Learning

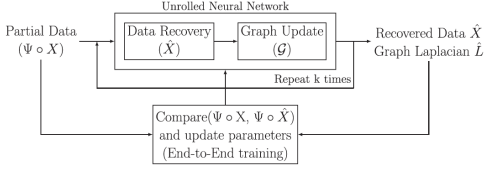
Inpainting-Driven Graph Learning via Explainable Neural Networks
Lingxiao Zhang, Gene Cheung, Xavier Bresson, and Ching-Yun Ko - 2025

Main contributions:

- Learns both signal and graph **jointly**, without ground-truth.
- Designed for **temporal signals** (e.g., fMRI, sensors).
- **Feedback loop:** signal recovery → graph update → repeat.
- Fully **unsupervised**, interpretable and task-adaptive.

Compared to our main paper:

- Focus on **dynamic graphs** vs static.
- **Closed-loop unrolling** vs fixed iteration blocks.
- **Unsupervised** learning vs supervised training with ground-truth signals.
- Full **end-to-end feedback architecture** vs modular design (graph then signal).



Joint unrolling of signal recovery and graph learning.