Analysis Report on *Interpretable Lightweight Transformer via Unrolling of Learned Graph Smoothness Priors*

Kenneth Browder [1] **Fabien Lagnieu** [1]

Abstract

This report analyzes the paper *Interpretable Lightweight Transformer via Unrolling of Learned Graph Smoothness Priors* (Do et al., 2024). The authors propose an efficient and interpretable transformer-like model for graph-signal interpolation. Their method uses unrolled optimization algorithms based on graph smoothness priors. The first section presents its main contributions, methodology, and experimental results, and describes the reproduction of one key experiment and an additional experiment. The second section examines a related paper, discusses its relevance, and compares its approach with the first.

Section 1: Analysis of the Main Paper

Interpretable Lightweight Transformer via Unrolling of Learned Graph Smoothness Priors
Tam Thuc Do, Parham Eftekhar, Seyed Alireza Hosseini, Gene Cheung, Philip Chou - Published in June 2024.

1.1. Introduction

Transformer architectures are central to modern machine learning, particularly in natural language processing and computer vision. Despite their success, they face two main limitations: high computational cost and lack of interpretability. These issues restrict their use in resource-constrained settings or applications requiring transparency.

The paper proposes a simple and effective solution by combining Graph Signal Processing (GSP) with neural network architectures. GSP models data structured as graphs by leveraging their relational information. The authors unroll optimization algorithms for graph signal interpolation into a neural network, resulting in a transformer-like model that is both lightweight and interpretable.

[*]Equal contribution [1]Institut Polytechnique de Paris, France. Correspondence to: Kenneth Browder <kenneth.browder@ip-paris.fr>, Fabien Lagnieu <fabien.lagnieu@polytechnique.edu>.

Machine Learning with Graphs course - IPP, Palaiseau, France, 2025.

1.2. State of the Art

Graph-based learning has made important progress in recent years, so many methods focus on learning from data structured as graphs. GSP provides a solid framework for analyzing such data, it extends traditional signal processing tools to irregular graph domains and allows processing signals that live on nodes of a graph by using the graph structure (Shuman et al., 2013).

Two key tools in GSP are the Graph Laplacian Regularizer (GLR) and Graph Total Variation (GTV). GLR promotes smoothness, encouraging similar values for connected nodes (Ortega et al., 2018). In contrast, GTV allows sharp changes and preserves edges and discontinuities (Cheung et al., 2018).

In the analyzed paper, GLR and GTV guide both graph learning and signal interpolation, ensuring smoothness and edge preservation in the reconstructed signals.

1.3. Main Contributions of the Proposed Method

The proposed method rethinks the transformer paradigm by integrating graph learning and optimization unrolling into a unified framework for graph signal interpolation. Its core contributions are outlined below.

1.3.1. Overview of the Transformer-like Architecture

The architecture alternates between learning a similarity graph and updating the interpolated signal. At each layer, the graph and signal are refined through an unrolled optimization process. This iterative structure mimics transformers but relies on graph-based operations, ensuring interpretability and lower complexity.

1.3.2. Graph Learning via Mahalanobis Distance

The graph learning module constructs a similarity graph by computing pairwise distances between nodes. It uses the *Mahalanobis distance* (Hu et al., 2020), which measures the similarity between two feature vectors $\mathbf{f}_i$ and $\mathbf{f}_j$.

The *Mahalanobis distance* between two nodes i and j is

defined as: $d(i,j) = (\mathbf{f}_i - \mathbf{f}_j)^\top \mathbf{M}(\mathbf{f}_i - \mathbf{f}_j)$, where $\mathbf{M}$ is a learnable, positive semi-definite matrix that adapts the metric to the data during training. And the edge weights $w_{i,j}$ between nodes are computed as: $w_{i,j} = \exp(-d(i,j))$. A local softmax normalizes outgoing edges so their weights sum to one, similar to transformer attention.

This approach guarantees the symmetry of the learned graph and provides an interpretable alternative to the key-query-value attention commonly used in standard transformers.

1.3.3. Optimization Unrolling for Signal Interpolation

Once the graph is defined, the model performs signal interpolation guided by graph smoothness priors. Two optimization strategies are employed: the *Conjugate Gradient* (CG) method used to minimize the ℓ_2-norm GLR (Shewchuk, 1994) and the *Alternating Direction Method of Multipliers* (ADMM) applied to minimize the ℓ_1-norm GTV (Wang & Shroff, 2017).

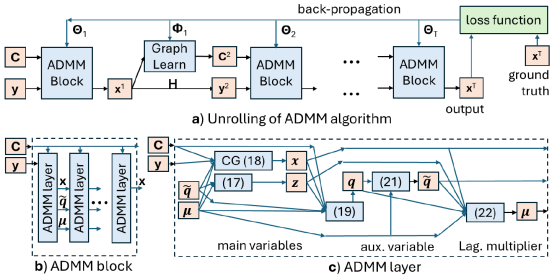

Figure 1. Unrolling of GTV-based signal interpolation algorithm (Do et al., 2024).

Each iteration of these algorithms is implemented as a neural network layer, enabling end-to-end training through backpropagation. Each iteration of these algorithms is implemented as a neural network layer, enabling end-to-end training through backpropagation. A schematic algorithm illustrating the iterative structure is provided in Appendix.

1.3.4. Interpretability and Efficiency

Each layer performs a well-defined, interpretable operation, replacing traditional self-attention with a graph learning module guided by graph smoothness priors, avoiding the opacity of key-query-value mechanisms.

By leveraging unrolled optimization and efficient graph-based operations, the model eliminates large parameter matrices, reducing computational cost while maintaining strong performance.

This model advances the state-of-the-art by showing that transformer-like architectures can achieve competitive performance with fewer parameters, while offering greater interpretability than conventional attention mechanisms.

1.4. Experimental Validation

1.4.1. Experimental Setup

All experiments are implemented in PyTorch. Models are trained on the DIV2K dataset, containing 800 high-resolution (HR) training images and 100 validation images. To reduce computational cost, the authors extract 64×64 pixel patches and sample only 1% to 4% of all available patches for training. Testing is performed on three standard benchmarks: McM (Zhang et al., 2011), Kodak (Eastman Kodak, 1993), and Urban100 (Huang et al., 2015).

Two tasks are considered: *Image Interpolation* (reconstructing high-resolution images from sparsely sampled data) and *Demosaicking* (reconstructing full-color images from Bayer-patterned inputs).

Performance is evaluated using Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) (Wang et al., 2004).

1.4.2. Results on Image Interpolation

The authors evaluate their models on image interpolation tasks. The proposed models, uGLR and uGTV, are compared to established methods including MAIN (Ji et al., 2020) and SwinIR-lightweight (Liang et al., 2021), as well as a simple Bicubic interpolation baseline.

uGTV consistently achieves the best PSNR and SSIM scores across all datasets. Notably, it outperforms MAIN and SwinIR, despite having significantly fewer parameters (319k versus 10.9M for MAIN and 0.9M for SwinIR).

1.4.3. Results on Demosaicking

For demosaicking, uGTV shows competitive performance with a lightweight design. Table 1 presents selected results on the Kodak and Urban100 datasets. To maintain readability, only the most relevant models (Bilinear, RSTCANet-B/S (Xing et al., 2022), uGLR, and uGTV) are reported.

uGTV achieves top PSNR and SSIM on Urban100 and closely matches RST-S on Kodak, while using nearly ten times fewer parameters. These results confirm uGTV's balance between accuracy and efficiency. Its performance generalizes well across datasets without complex architectures.

Table 1. Demosaicking performance (PSNR / SSIM) for selected models, trained on 10k samples from DIV2K.

Method	Params #	Kodak	Urban100
Bilinear	-	28.22 / 0.890	24.18 / 0.873
RST-B	931,763	38.75 / 0.986	32.82 / 0.973
RST-S	3,162,211	**39.81 / 0.988**	33.87 / 0.978
uGLR	323,410	37.88 / 0.982	33.60 / 0.977
uGTV	323,435	39.11 / 0.986	**34.01 / 0.979**

Table 2. Demosaicking performance (PSNR) under Gaussian noise (trained on noiseless data).

METHOD	$\sigma = 10$	$\sigma = 20$	$\sigma = 30$	$\sigma = 50$
RST-B	28.01	22.70	19.34	15.03
UGLR	28.24	22.84	19.49	15.20
UGTV	**28.31**	**22.89**	**19.56**	**15.38**

1.4.4. ROBUSTNESS TO COVARIATE SHIFT

The authors also assess the robustness of their models under covariate shift by adding Gaussian noise to the inputs during testing. Table 2 shows the demosaicking PSNR values for different noise levels (σ). uGTV consistently outperforms RST-B, especially as noise increases.

These results confirm that both uGLR and uGTV are more robust to noise compared to RST-B, with uGTV showing the best stability across all tested noise levels.

1.4.5. DISCUSSION

The experimental results confirm the effectiveness of the proposed method. Despite having significantly fewer parameters, uGLR and uGTV outperform or match the performance of more complex models like RSTCANet and MAIN. Additionally, they demonstrate better robustness under covariate shifts.

1.5. Reproduction and Additional Experiments

In this section, we describe the experiments we performed to reproduce the key results of the paper, and an additional experiment designed to assess the robustness and generalization capabilities of the proposed method.

1.5.1. REPRODUCTION OF A KEY EXPERIMENT

Objective: Validate the demosaicking performance of the proposed uGTV model on the ZzZzZ dataset.

Methodology: ZzZzZ ZzZzZ ZzZzZ ZzZzZ ZzZzZ ZzZzZ ZzZzZ

Results: ZzZzZ ZzZzZ ZzZzZ ZzZzZ ZzZzZ ZzZzZ

Discussion: ZzZzZ ZzZzZ ZzZzZ ZzZzZ ZzZzZ ZzZzZ

1.5.2. ADDITIONAL EXPERIMENT

Objective: ZzZzZ ZzZzZ ZzZzZ ZzZzZ ZzZzZ

Methodology: ZzZzZ ZzZzZ ZzZzZ ZzZzZ ZzZzZ

Results: ZzZzZ ZzZzZ ZzZzZ ZzZzZ ZzZzZ

Discussion: ZzZzZ ZzZzZ ZzZzZ ZzZzZ ZzZzZ

Further details on these experiments are provided in the appendix. The full codebase is available at:

https://github.com/XwaeK/ML_with_Graphs_Project.git

1.6. Critical Analysis

The proposed model successfully combines interpretability, computational efficiency, and strong empirical results, achieving state-of-the-art performance with significantly fewer parameters.

However, its scalability to very large graphs remains uncertain, and it has so far only been applied to image-based signal interpolation and demosaicking tasks; future work could explore its extension to more diverse graph learning problems such as node classification or link prediction.

Section 2: Analysis of a Related Paper

Inpainting-Driven Graph Learning via Explainable Neural Networks
Minh Trieu Trinh, Tam V. Nguyen, Thanh-Toan Do, Dinh Phung, Svetha Venkatesh - Published in 2024.

2.1. ZzZzZ ZzZzZ Kenneth ZzZzZ ZzZzZ

References

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

Do, T. T., Eftekhar, P., Hosseini, S. A., Cheung, G., and Chou, P. A. 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, 2024. PMLR.

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

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

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

Ji, J., Zhong, B., and Ma, K.-K. Image interpolation using multi-scale attention-aware inception network. *IEEE Transactions on Image Processing*, 29:9413–9428, 2020.

Liang, J., Cao, J., Sun, G., Zhang, K., Van Gool, L., and Timofte, R. Swinir: Image restoration using swin transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 1833–1844, 2021.

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

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

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

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

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

Xing, M., Liu, Y., Ye, P., Zhu, L., Yang, Q., and Wang, C. Residual spatial-transformer channel attention network for image demosaicking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1852–1861, 2022.

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

Algorithm 1 Unrolling of GTV-based signal interpolation algorithm.

Input: Initial signal observations y, initial graph C_1
Parameters: Number of unrolled layers T, ADMM iterations K, operator $H(\cdot)$
Learnable: Mahalanobis matrix M, ADMM parameters, H
Initialization: $y_1 \leftarrow y$
for $t = 1$ **to** T **do**
  Initialize $x_t^{(0)}$, $q_t^{(0)}$, $\mu_t^{(0)}$
  for $k = 1$ **to** K **do**
    $(x_t^{(k)}, q_t^{(k)}, \mu_t^{(k)}) \leftarrow \text{ADMM\_Layer}(C_t, y_t, q_t^{(k-1)}, \mu_t^{(k-1)})$
  end for
  $x_t \leftarrow x_t^{(K)}$ *# Final signal at layer t*
  if $t < T$ **then**
    $C_{t+1} \leftarrow \text{GraphLearn}(x_t; M)$ *# Learnable graph from x_t*
    $y_{t+1} \leftarrow H(x_t)$ *# Feature transform for next iteration*
  end if
end for
Compute loss: $\mathcal{L}(x_T, x_{gt})$ *# x ground truth x_{gt}*
Backpropagate gradients through all layers to update parameters

A. You *can* have an appendix here.

You can have as much text here as you want. The main body must be at most 8 pages long. For the final version, one more page can be added. If you want, you can use an appendix like this one.

The `\onecolumn` command above can be kept in place if you prefer a one-column appendix, or can be removed if you prefer a two-column appendix. Apart from this possible change, the style (font size, spacing, margins, page numbering, etc.) should be kept the same as the main body.