Interpretable Lightweight Transformer via Unrolling of Learned Graph Smoothness Priors

Tam Thuc Do
York University
Toronto, Canada
dtamthuc@yorku.ca

Parham Eftekhar
York University
Toronto, Canada
eftekhar@yorku.ca

Seyed Alireza Hosseini
York University
Toronto, Canada
ahoseini@yorku.ca

Gene Cheung
York University
Toronto, Canada
genec@yorku.ca

Philip A. Chou
packet.media
Seattle, USA
pachou@ieee.org

Abstract

We build interpretable and lightweight transformer-like neural networks by unrolling iterative optimization algorithms that minimize graph smoothness priors—the quadratic graph Laplacian regularizer (GLR) and the ℓ_1-norm graph total variation (GTV)—subject to an interpolation constraint. The crucial insight is that a normalized signal-dependent graph learning module amounts to a variant of the basic self-attention mechanism in conventional transformers. Unlike "black-box" transformers that require learning of large key, query and value matrices to compute scaled dot products as affinities and subsequent output embeddings, resulting in huge parameter sets, our unrolled networks employ shallow CNNs to learn low-dimensional features per node to establish pairwise Mahalanobis distances and construct sparse similarity graphs. At each layer, given a learned graph, the target interpolated signal is simply a low-pass filtered output derived from the minimization of an assumed graph smoothness prior, leading to a dramatic reduction in parameter count. Experiments for two image interpolation applications verify the restoration performance, parameter efficiency and robustness to covariate shift of our graph-based unrolled networks compared to conventional transformers.

1 Introduction

Focusing on the *self-attention* mechanism [1] as the basic building block—where the *affinity* between two input tokens is computed as a transformed dot product—*transformers* [2] learn large parameter sets to achieve state-of-the-art (SOTA) performance in a wide range of signal prediction/classification problems [3, 4], outperforming convolutional neural nets (CNNs) and recurrent neural nets (RNNs). However, there are shortcomings: i) lack of mathematical interpretability to characterize general performance[1], ii) requiring substantial training datasets to train sizable parameters [12], and iii) fragility to *covariate shift*—when training and testing data have different distributions [13].

Orthogonally, *algorithm unrolling* [14] implements iterations of a model-based algorithm as a sequence of neural layers to build a feed-forward network, whose parameters can be learned end-to-end via back-propagation from data. A classic example is the unrolling of the *iterative soft-*

[1]While works exist to analyze existing transformer architectures [5, 6, 7, 8, 9], only [10, 11] characterized the performance of a single self-attention layer and a shallow transformer, respectively. In contrast, we build transformer-like networks by unrolling graph-based algorithms, so that each layer is interpretable by construction.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

thresholding algorithm (ISTA) in sparse coding into *Learned* ISTA (LISTA) [15]. Recently, [16] showed that by unrolling an iterative algorithm minimizing a sparse rate reduction (SRR) objective, it can lead to a family of "white-box" transformer-like deep neural nets that are 100% mathematically interpretable. Inspired by [16], in this work we also seek to build white-box transformers via algorithm unrolling, but from a unique *graph signal processing* (GSP) perspective [17, 18, 19].

Over the last decade and a half, GSP studies spectral analysis and processing of discrete signals on structured data kernels described by finite graphs. Specifically, by assuming that the sought signal is smooth (low-pass) with respect to (w.r.t.) a particular graph, a plethora of graph-based restoration algorithms can be designed for practical applications, including image denoising [20], JPEG dequantization [21], interpolation [22], 3D point cloud denoising [23, 24] and super-resolution [25]. At the heart of GSP is the construction of a *similarity graph* that captures pairwise similarities between signal samples on two connected nodes. We first demonstrate that *signal-dependent similarity graph learning with normalization is akin to affinity computation in the self-attention mechanism*. Thus, *our first contribution is to show that unrolling of a graph-based iterative algorithm*[2] *with normalized graph learning results in an interpretable transformer-like feed-forward network*.

Second, computation of a positive weight $w_{i,j} = \exp(-d(i,j))$ of an edge (i,j) connecting two graph nodes i and j often employs *Mahalanobis distance* $d(i,j) = (\mathbf{f}_i - \mathbf{f}_j)^\top \mathbf{M}(\mathbf{f}_i - \mathbf{f}_j)$ between representative (*e.g.*, CNN-computed) *feature vectors* $\mathbf{f}_i$ and $\mathbf{f}_j$, where $\mathbf{f}_i, \mathbf{f}_j \in \mathbb{R}^D$ reside in low-dimensional space [27, 28]. Hence, unlike a conventional transformer that requires large key and query matrices, $\mathbf{K}$ and $\mathbf{Q}$, to compute transformed dot products, the similarity graph learning module can be more parameter-efficient. Moreover, by adding a graph smoothness prior such as *graph Laplacian regularizer* (GLR) [18, 20] or *graph total variation* (GTV) [29, 30, 31] in the optimization objective, once a graph $\mathcal{G}$ is learned, the target signal is simply computed as the low-pass filtered output derived from the minimization of the assumed graph smoothness prior. Thus, a large value matrix $\mathbf{V}$ to compute output embeddings typical in a transformer is also not needed. *Our second contribution is to demonstrate that a* ***lightweight*** *transformer with fewer parameters can be built via unrolling of a graph-based restoration algorithm with a chosen graph signal smoothness prior*.

Specifically, focusing on the signal interpolation problem, we first derive linear-time graph-based algorithms by minimizing GLR or GTV, via *conjugate gradient* (CG) [32] or a modern adaptation of *alternative method of multipliers* (ADMM) for *sparse linear programming* (SLP) [33], respectively. In each iteration, given a learned graph, each algorithm deploys a low-pass graph filter to interpolate the up-sampled observation vector into the target signal. We intersperse unrolled algorithm iterations with graph learning modules into a compact and interpretable neural network. We demonstrate its restoration performance, parameter efficiency (3% of SOTA's parameters in one case), and robustness to covariate shift for two practical applications: image demosaicking, and image interpolation.

Notation: Vectors and matrices are written in bold lowercase and uppercase letters, respectively. The (i,j) element and the j-th column of a matrix $\mathbf{A}$ are denoted by $A_{i,j}$ and $\mathbf{a}_j$, respectively. The i-th element in the vector $\mathbf{a}$ is denoted by a_i. The square identity matrix of rank N is denoted by $\mathbf{I}_N$, the M-by-N zero matrix is denoted by $\mathbf{0}_{M,N}$, and the vector of all ones / zeros of length N is denoted by $\mathbf{1}_N$ / $\mathbf{0}_N$, respectively. Operator $\|\cdot\|_p$ denotes the ℓ-p norm.

2 Preliminaries

2.1 GSP Definitions

A graph $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathbf{W})$ is defined by a node set $\mathcal{N} = \{1, \ldots, N\}$ and an edge set $\mathcal{E}$ of size $|\mathcal{E}| = M$, where $(i,j) \in \mathcal{E}$ means nodes $i, j \in \mathcal{N}$ are connected with weight $w_{i,j} = W_{i,j} \in \mathbb{R}$. In this paper, we consider only *positive* graphs $\mathcal{G}$ with no self-loops, *i.e.*, $w_{i,j} \geq 0, \forall i, j$, and $w_{i,i} = 0, \forall i$. We assume edges are undirected, and thus *adjacency matrix* $\mathbf{W} \in \mathbb{R}^{N \times N}$ is symmetric. The *combinatorial graph Laplacian matrix* is defined as $\mathbf{L} \triangleq \mathbf{D} - \mathbf{W} \in \mathbb{R}^{N \times N}$, where $\mathbf{D} \triangleq \text{diag}(\mathbf{W}\mathbf{1}_N)$ is the *degree matrix*, and $\text{diag}(\mathbf{v})$ returns a diagonal matrix with $\mathbf{v}$ along its diagonal. $\mathbf{L}$ for a positive graph $\mathcal{G}$ is provably *positive semi-definite* (PSD), *i.e.*, all its eigenvalues λ_i's are non-negative [19].

[2]While there exist works on unrolling of graph-based algorithms [26], they typically assume a *fixed* graph and optimize other parameters. In contrast, the key point in our work is that the normalized graph learning module is akin to the basic self-attention mechanism in conventional transformers.

We define also the *incidence matrix* $\mathbf{C} = \mathbb{R}^{M \times N}$: each k-th row of $\mathbf{C}$ corresponds to the k-th edge $(i, j) \in \mathcal{E}$, where $C_{k,i} = w_{i,j}$, $C_{k,j} = -w_{i,j}$, and $C_{k,l} = 0,\ \forall l \neq i, j$. Since our assumed graph $\mathcal{G}$ is undirected, the polarities of $C_{k,i}$ and $C_{k,j}$ are arbitrary, as long as they are opposite.

2.2 Graph Laplacian Regularizer

Given a positive *connected* graph $\mathcal{G}$ with N nodes and M edges, we first define smoothness of a signal $\mathbf{x} \in \mathbb{R}^N$ w.r.t. $\mathcal{G}$ using the *graph Laplacian regularizer* (GLR) [18, 20] as

$$\|\mathbf{x}\|_{\mathcal{G},2} = \mathbf{x}^\top \mathbf{L}\mathbf{x} = \sum_{(i,j)\in\mathcal{E}} w_{i,j}(x_i - x_j)^2 \tag{1}$$

where $\mathbf{L}$ is a combinatorial Laplacian matrix specifying graph $\mathcal{G}$. GLR (1) is non-negative for a positive graph, and thus is suitable as a signal prior for minimization problems [20, 21].

2.3 Graph Total Variation

Instead of GLR (1), we can alternatively define graph signal smoothness using *graph total variation* (GTV) [29, 30, 31] $\|\mathbf{x}\|_{\mathcal{G},1}$ for signal $\mathbf{x} \in \mathbb{R}^N$ as

$$\|\mathbf{x}\|_{\mathcal{G},1} = \|\mathbf{C}\mathbf{x}\|_1 \overset{(a)}{=} \sum_{(i,j)\in\mathcal{E}} w_{i,j}|x_i - x_j| \tag{2}$$

where (a) is true since $\mathcal{G}$ is positive. GTV is also non-negative for positive $\mathcal{G}$, and has been used as a signal prior for restoration problems such as image deblurring [34].

3 Problem Formulation & Optimization using GLR

3.1 Problem Formulation

We first assume a positive, *sparse* and *connected* graph $\mathcal{G}$ with N nodes and M edges specified by graph Laplacian matrix $\mathbf{L}$. By sparse, we mean that M is $\mathcal{O}(N)$ and not $\mathcal{O}(N^2)$. By connected, we mean that any node j can be traversed from any other node i. Given $\mathcal{G}$, we first derive a *linear-time* iterative algorithm to interpolate signal $\mathbf{x}$ by minimizing GLR given observed samples $\mathbf{y}$. In Section 4, we derive an algorithm by minimizing GTV instead given $\mathbf{y}$. In Section 5, we unroll iterations of one of two derived algorithms into neural layers, together with strategically inserted graph learning modules, to construct graph-based lightweight transformer-like neural nets.

We first employ GLR [20] as the objective to reconstruct $\mathbf{x} \in \mathbb{R}^N$ given partial observation $\mathbf{y} \in \mathbb{R}^K$, where $K < N$. Denote by $\mathbf{H} \in \{0, 1\}^{K \times N}$ a *sampling matrix* defined as

$$H_{i,j} = \begin{cases} 1 & \text{if node } j \text{ is the } i\text{-th sample} \\ 0 & \text{o.w.} \end{cases} \tag{3}$$

that picks out K samples from signal $\mathbf{x}$. The optimization is thus

$$\min_{\mathbf{x}}\ \mathbf{x}^\top \mathbf{L}\mathbf{x}, \qquad \text{s.t. } \mathbf{H}\mathbf{x} = \mathbf{y} \tag{4}$$

where $\mathbf{L} \in \mathbb{R}^{N \times N}$ is a graph Laplacian matrix corresponding to a positive graph $\mathcal{G}$ [19]. PSD $\mathbf{L}$ implies that $\mathbf{x}^\top \mathbf{L}\mathbf{x} \geq 0, \forall \mathbf{x}$, and thus (4) has a convex objective with a linear interpolation constraint.

3.2 Optimization

We solve (4) via a standard Lagrangian approach [35] and write its corresponding unconstrained Lagrangian function $f(\mathbf{x}, \boldsymbol{\mu})$ as

$$f(\mathbf{x}, \boldsymbol{\mu}) = \mathbf{x}^\top \mathbf{L}\mathbf{x} + \boldsymbol{\mu}^\top (\mathbf{H}\mathbf{x} - \mathbf{y}) \tag{5}$$

where $\boldsymbol{\mu} \in \mathbb{R}^K$ is the Lagrange multiplier vector. To minimize $f(\mathbf{x}, \boldsymbol{\mu})$ in (5), we take the derivative w.r.t. $\mathbf{x}$ and $\boldsymbol{\mu}$ separately and set them to zero, resulting in the following linear system:

$$\underbrace{\begin{bmatrix} 2\mathbf{L} & \mathbf{H}^\top \\ \mathbf{H} & \mathbf{0}_{K,K} \end{bmatrix}}_{\mathbf{P}} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{N,N} \\ \mathbf{y} \end{bmatrix}. \tag{6}$$

Given that the underlying graph $\mathcal{G}$ is positive and connected, coefficient matrix $\mathbf{P}$ is provably full-rank and thus invertible (see Appendix A.1 for a proof). Hence, (6) has a unique solution $\mathbf{x}^*$.

Suppose we index the sampled nodes $\mathcal{S}$ in $\mathbf{x}$ before the non-sampled nodes $\bar{\mathcal{S}}$, *i.e.*, $\mathbf{x} = [\mathbf{x}_{\mathcal{S}}; \mathbf{x}_{\bar{\mathcal{S}}}]$. Then $\mathbf{H} = [\mathbf{I}_K \;\; \mathbf{0}_{K,N-K}]$, and the second block row in (6) implies $\mathbf{x}_{\mathcal{S}} = \mathbf{y}$. Suppose we write $\mathbf{L} = [\mathbf{L}_{\mathcal{S},\mathcal{S}} \;\; \mathbf{L}_{\mathcal{S},\bar{\mathcal{S}}}; \mathbf{L}_{\bar{\mathcal{S}},\mathcal{S}} \;\; \mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}]$ in blocks also. For the first block row in (6), consider only the non-sampled rows:

$$\left(\begin{bmatrix} 2\mathbf{L} & \mathbf{H}^\top \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\mu} \end{bmatrix} \right)_{\bar{\mathcal{S}}} = 2(\mathbf{L}_{\bar{\mathcal{S}},\mathcal{S}}\, \mathbf{x}_{\mathcal{S}} + \mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\, \mathbf{x}_{\bar{\mathcal{S}}}) = \mathbf{0}_{N-K} \tag{7}$$

where $\left(\mathbf{H}^\top \boldsymbol{\mu}\right)_{\bar{\mathcal{S}}} = \mathbf{0}_{N-K}$ since the non-sampled rows of $\mathbf{H}^\top$ (the non-sampled columns of $\mathbf{H}$) are zeros. Thus, $\mathbf{x}_{\bar{\mathcal{S}}}$ can be computed via the following system of linear equations:

$$\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\, \mathbf{x}_{\bar{\mathcal{S}}} = -\mathbf{L}_{\bar{\mathcal{S}},\mathcal{S}}\, \mathbf{y} \tag{8}$$

where $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ is a symmetric, sparse, and provably *positive definite* (PD) matrix (see Appendix A.2 for a proof). Thus, there exists a unique solution $\mathbf{x}_{\bar{\mathcal{S}}}$ in (8).

Complexity: For notation simplicity, let $\mathcal{L} = \mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$. Linear system (8) can be solved efficiently using *conjugate gradient* (CG), an iterative descent algorithm with complexity $\mathcal{O}(\text{nnz}(\boldsymbol{\mathcal{L}})\sqrt{\kappa(\boldsymbol{\mathcal{L}})}/\log(\epsilon))$, where $\text{nnz}(\boldsymbol{\mathcal{L}})$ is the number of non-zero entries in matrix $\boldsymbol{\mathcal{L}}$, $\kappa(\mathcal{L}) = \frac{\lambda_{\max}(\boldsymbol{\mathcal{L}})}{\lambda_{\min}(\boldsymbol{\mathcal{L}})}$ is the *condition number* of $\boldsymbol{\mathcal{L}}$, $\lambda_{\max}(\boldsymbol{\mathcal{L}})$ and $\lambda_{\min}(\boldsymbol{\mathcal{L}})$ are the respective largest and smallest eigenvalues of $\boldsymbol{\mathcal{L}}$, and ϵ is the convergence threshold of the gradient search [32]. Because $\mathbf{L}$ is sparse by graph construction ($\mathcal{O}(N)$ edges), $\mathcal{L} = \mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ is also sparse, *i.e.*, $\text{nnz}(\mathcal{L}) = \mathcal{O}(N)$. Assuming $\kappa(\mathcal{L})$ can be reasonably lower-bounded for PD $\mathcal{L}$ and ϵ is reasonably chosen, the complexity of solving (8) using CG is $\mathcal{O}(N)$.

Interpretation: To elicit a signal filtering interpretation from (6), we assume for now that $\mathbf{L}$ is PD[3] and thus invertible. Recall that the block matrix inversion formula [36] is

$$\mathbf{P}^{-1} = \left(\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \right)^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{P}/\mathbf{A})\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{P}/\mathbf{A}) \\ -(\mathbf{P}/\mathbf{A})\mathbf{C}\mathbf{A}^{-1} & (\mathbf{P}/\mathbf{A}) \end{bmatrix} \tag{9}$$

where $\mathbf{P}/\mathbf{A} = (\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}$ is the *Schur complement* of block $\mathbf{A}$ of matrix $\mathbf{P}$. Solution $\mathbf{x}^*$ can thus be computed as:

$$\begin{aligned} \mathbf{x}^* &= \mathbf{L}^{-1}\mathbf{H}^\top \left(\mathbf{H}\mathbf{L}^{-1}\mathbf{H}^\top\right)^{-1} \mathbf{y} \\ &= \mathbf{L}^{-1}\mathbf{H}^\top \left((\mathbf{L}^{-1})_{\mathcal{S}}\right)^{-1} \mathbf{y} = \mathbf{L}^{-1}\mathbf{H}^\top \mathbf{L}_{\mathcal{S}}^{\#}\mathbf{y} \end{aligned} \tag{10}$$

where $\mathbf{L}_{\mathcal{S}}^{\#} \triangleq \left((\mathbf{L}^{-1})_{\mathcal{S}}\right)^{-1}$ and $(\mathbf{L}^{-1})_{\mathcal{S}}$ denotes the rows and columns of $\mathbf{L}^{-1}$ corresponding to the sampled nodes. $\mathbf{L}_{\mathcal{S}}^{\#}$ is a high-pass filter similar to $\mathbf{L}_{\mathcal{S}}$. Thus, we can interpret $\mathbf{x}^*$ as a *low-pass filtered output of up-sampled* $\mathbf{H}^\top\mathbf{L}_{\mathcal{S}}^{\#}\mathbf{y}$—with low-pass filter response $r(\boldsymbol{\Lambda}) = \boldsymbol{\Lambda}^{-1}$ where $\mathbf{L} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$, $\boldsymbol{\Lambda} = \text{diag}([\lambda_1, \ldots, \lambda_N])$, is eigen-decomposible with frequencies λ_k's and Fourier modes $\mathbf{v}_k$'s.

4 Problem Formulation & Optimization using GTV

4.1 Problem Formulation

Given a positive connected graph $\mathcal{G}$ specified by incidence matrix $\mathbf{C} \in \mathbb{R}^{M \times N}$ and partial observation $\mathbf{y}$, we now employ instead GTV as the objective to interpolate target signal $\mathbf{x}$, resulting in

$$\min_{\mathbf{x}} \; \|\mathbf{C}\mathbf{x}\|_1, \qquad \text{s.t. } \mathbf{H}\mathbf{x} = \mathbf{y}, \tag{11}$$

(11) is a *linear program* (LP), since both the objective and the lone constraint are linear. Thus, while minimizing GLR leads to a *linear system* (6), minimizing GTV leads to a *linear program* (11).

[3] A combinatorial graph Laplacian matrix for a positive graph with at least one additional positive self-loop is provably PD; see proof in Appendix A.2.

4.1.1 LP in Standard Form

First, we rewrite LP (11) in standard form as follows. Define *upper-bound variable* $\mathbf{z} \in \mathbb{R}^M$ with a pair of linear constraints $\mathbf{z} \geq \pm\mathbf{C}\mathbf{x}$. This enables a linear objective $\mathbf{1}_M^\top \mathbf{z}$ for a minimization problem (thus ensuring the upper bound is tight), *i.e.*, $\mathbf{z} = \|\mathbf{C}\mathbf{x}\|_1$. Second, we introduce non-negative *slack variables* $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^M$ to convert inequality constraints $\mathbf{z} \geq \pm\mathbf{C}\mathbf{x}$ to equality constraints $\mathbf{z} = \mathbf{C}\mathbf{x} + \mathbf{q}_1$ and $\mathbf{z} = -\mathbf{C}\mathbf{x} + \mathbf{q}_2$. Thus, LP (11) can be rewritten as

$$\min_{\mathbf{z},\mathbf{x},\mathbf{q}} \mathbf{1}_M^\top \mathbf{z}, \quad \text{s.t.} \underbrace{\begin{bmatrix} \mathbf{I}_M & -\mathbf{C} & -(\mathbf{I}_M \ \mathbf{0}_{M,M}) \\ \mathbf{I}_M & \mathbf{C} & -(\mathbf{0}_{M,M} \ \mathbf{I}_M) \\ \mathbf{0}_{K,M} & \mathbf{H} & \mathbf{0}_{K,2M} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \mathbf{z} \\ \mathbf{x} \\ \mathbf{q} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{0}_M \\ \mathbf{0}_M \\ \mathbf{y} \end{bmatrix}}_{\mathbf{b}}, \quad \mathbf{q} \geq \mathbf{0}_{2M} \tag{12}$$

where $\mathbf{q} = [\mathbf{q}_1; \mathbf{q}_2] \in \mathbb{R}^{2M}$.

4.2 Optimization Algorithm

Because coefficient matrix $\mathbf{A}$ is sparse, (12) is a *sparse linear program* (SLP). We solve SLP (12) efficiently by adopting an ADMM approach for SLP in [33]. We first define a convex but non-differentiable (non-smooth) *indicator function*:

$$g(\mathbf{q}) = \begin{cases} 0 & \text{if } q_j \geq 0, \ \forall j \\ \infty & \text{o.w.} \end{cases}. \tag{13}$$

We next introduce *auxiliary variable* $\tilde{\mathbf{q}} \in \mathbb{R}^{2M}$ and equality constraint $\tilde{\mathbf{q}} = \mathbf{q}$. We now rewrite (12) with a single equality constraint as

$$\min_{\mathbf{z},\mathbf{x},\mathbf{q},\tilde{\mathbf{q}}} \mathbf{1}_M^\top \mathbf{z} + g(\tilde{\mathbf{q}}), \quad \text{s.t.} \underbrace{\begin{bmatrix} \mathbf{A} \\ \mathbf{0}_{2M,M+N} \ \mathbf{I}_{2M} \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} \mathbf{z} \\ \mathbf{x} \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \tilde{\mathbf{q}} \end{bmatrix}. \tag{14}$$

We can now rewrite (14) into an unconstrained version using the augmented Lagrangian method as

$$\min_{\mathbf{z},\mathbf{x},\mathbf{q},\tilde{\mathbf{q}}} \mathbf{1}_M^\top \mathbf{z} + g(\tilde{\mathbf{q}}) + \boldsymbol{\mu}^\top \left(\mathbf{B} \begin{bmatrix} \mathbf{z} \\ \mathbf{x} \\ \mathbf{q} \end{bmatrix} - \begin{bmatrix} \mathbf{b} \\ \tilde{\mathbf{q}} \end{bmatrix} \right) + \frac{\gamma}{2} \left\| \mathbf{B} \begin{bmatrix} \mathbf{z} \\ \mathbf{x} \\ \mathbf{q} \end{bmatrix} - \begin{bmatrix} \mathbf{b} \\ \tilde{\mathbf{q}} \end{bmatrix} \right\|_2^2 \tag{15}$$

where $\boldsymbol{\mu} \in \mathbb{R}^{4M+K}$ is a Lagrange multiplier vector, and $\gamma > 0$ is a scalar parameter. In the sequel, we write $\boldsymbol{\mu} = [\boldsymbol{\mu}_a; \boldsymbol{\mu}_b; \boldsymbol{\mu}_c; \boldsymbol{\mu}_d; \boldsymbol{\mu}_e]$, where $\boldsymbol{\mu}_a, \boldsymbol{\mu}_b, \boldsymbol{\mu}_d, \boldsymbol{\mu}_e \in \mathbb{R}^M$ and $\boldsymbol{\mu}_c \in \mathbb{R}^K$.

4.2.1 Optimizing Main Variables

As typically done in ADMM, we minimize the unconstrained objective (15) alternately as follows. At iteration t, when $\tilde{\mathbf{q}}^t$ and $\boldsymbol{\mu}^t$ are fixed, the optimization for $\mathbf{z}^{t+1}$, $\mathbf{x}^{t+1}$ and $\mathbf{q}^{t+1}$ becomes

$$\min_{\mathbf{z},\mathbf{x},\mathbf{q}} \mathbf{1}_M^\top \mathbf{z} + (\boldsymbol{\mu}^t)^\top \left(\mathbf{B} \begin{bmatrix} \mathbf{z} \\ \mathbf{x} \\ \mathbf{q} \end{bmatrix} - \begin{bmatrix} \mathbf{b} \\ \tilde{\mathbf{q}} \end{bmatrix} \right) + \frac{\gamma}{2} \left\| \mathbf{B} \begin{bmatrix} \mathbf{z} \\ \mathbf{x} \\ \mathbf{q} \end{bmatrix} - \begin{bmatrix} \mathbf{b} \\ \tilde{\mathbf{q}} \end{bmatrix} \right\|_2^2. \tag{16}$$

The solution to this convex and smooth quadratic optimization is a system of linear equations,

$$\mathbf{z}^{t+1} = -\frac{1}{\gamma}\mathbf{1}_M - \frac{1}{2\gamma}\left(\boldsymbol{\mu}_a^t + \boldsymbol{\mu}_b^t + \boldsymbol{\mu}_d^t + \boldsymbol{\mu}_e^t\right) + \frac{1}{2}(\tilde{\mathbf{q}}_1^t + \tilde{\mathbf{q}}_2^t) \tag{17}$$

$$(\mathbf{C}^\top\mathbf{C} + \mathbf{H}^\top\mathbf{H})\mathbf{x}^{t+1} = \frac{1}{2\gamma}\mathbf{C}^\top\left(\boldsymbol{\mu}_a^t - \boldsymbol{\mu}_b^t + \boldsymbol{\mu}_d^t - \boldsymbol{\mu}_e^t\right) - \frac{1}{\gamma}\mathbf{H}^\top\boldsymbol{\mu}_c^t - \frac{1}{2}\mathbf{C}^\top(\tilde{\mathbf{q}}_1^t - \tilde{\mathbf{q}}_2^t) + \mathbf{H}^\top\mathbf{y} \tag{18}$$

$$\mathbf{q}_1^t = \frac{1}{2}\left(\mathbf{z}^{t+1} - \mathbf{C}\mathbf{x}^{t+1}\right) + \frac{1}{2\gamma}(\boldsymbol{\mu}_a^t - \boldsymbol{\mu}_d^t + \gamma\tilde{\mathbf{q}}_1^t)$$

$$\mathbf{q}_2^t = \frac{1}{2}\left(\mathbf{z}^{t+1} + \mathbf{C}\mathbf{x}^{t+1}\right) + \frac{1}{2\gamma}(\boldsymbol{\mu}_b^t - \boldsymbol{\mu}_e^t + \gamma\tilde{\mathbf{q}}_2^t) \tag{19}$$

See the Appendix A.3 for a derivation.

Linear system (18) is solvable if the coefficient matrix $\boldsymbol{\mathcal{L}} \triangleq \mathbf{L} + \mathbf{H}^\top \mathbf{H}$, where $\mathbf{L} \triangleq \mathbf{C}^\top \mathbf{C}$ is a PSD graph Laplacian for a positive graph, is invertible. See Appendix A.4 for a proof that $\mathcal{L}$ is PD and thus invertible .

Complexity: Linear system (18) again can be solved efficiently using CG with complexity $\mathcal{O}(\text{nnz}(\boldsymbol{\mathcal{L}})\sqrt{\kappa(\boldsymbol{\mathcal{L}})}/\log(\epsilon))$ [32]. Because $\mathbf{C}$ is sparse by graph construction ($\mathcal{O}(N)$ edges) and $\mathbf{H}$ is sparse by definition, $\mathcal{L}$ is also sparse, *i.e.*, $\text{nnz}(\mathcal{L}) = \mathcal{O}(N)$. Thus, assuming $\kappa(\mathcal{L})$ is also upper-bounded and ϵ is reasonably chosen, the complexity of solving (18) using CG is $\mathcal{O}(N)$.

Interpretation: (18) can be interpreted as follows. $\mathbf{H}^\top \mathbf{y}$ is the *up-sampled version* of observation $\mathbf{y}$. $\mathcal{L} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$, $\Lambda = \text{diag}([\lambda_1, \ldots, \lambda_N])$, is an eigen-decomposible *generalized Laplacian matrix*—Laplacian matrix $\mathbf{C}^\top \mathbf{C}$ plus self-loops of weight 1 at sampled nodes due to diagonal matrix $\mathbf{H}^\top \mathbf{H}$, and like a graph Laplacian $\mathbf{L}$ without self-loops, can be interpreted as a high-pass spectral filter [37]. $\mathcal{L}^{-1} = \mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{V}^\top$ is thus a *low-pass* spectral filter with frequency response $r(\boldsymbol{\Lambda}) = \boldsymbol{\Lambda}^{-1}$ to interpolate output $\mathcal{L}^{-1}\mathbf{H}^\top \mathbf{y}$. We interpret the remaining terms on the right-hand side of (18) as bias.

4.2.2 Optimizing Auxiliary Variable

Fixing $\mathbf{z}^{t+1}$, $\mathbf{x}^{t+1}$ and $\mathbf{q}^{t+1}$, the optimization for $\tilde{\mathbf{q}}^{t+1}$ for (15) simplifies to

$$\min_{\tilde{\mathbf{q}}} g(\tilde{\mathbf{q}}) + (\boldsymbol{\mu}_d^t)^\top \left(\mathbf{q}^{t+1} - \tilde{\mathbf{q}}\right) + \frac{\gamma}{2} \left\|\mathbf{q}^{t+1} - \tilde{\mathbf{q}}\right\|_2^2. \tag{20}$$

The solution for optimal $\tilde{\mathbf{q}}^{t+1}$ is term-by-term thresholding:

$$\tilde{q}_i^{t+1} = \begin{cases} q_i^{t+1} + \frac{1}{\gamma}\mu_{d,i}^t & \text{if } q_i^{t+1} + \frac{1}{\gamma}\mu_{d,i}^t \geq 0 \\ 0 & \text{o.w.} \end{cases}, \forall i. \tag{21}$$

See Appendix A.5 for a derivation.

4.2.3 Updating Lagrange Multiplier

The Lagrange multiplier $\boldsymbol{\mu}^{t+1}$ can be updated in the usual manner in an ADMM framework [38]:

$$\boldsymbol{\mu}^{t+1} = \boldsymbol{\mu}^t + \gamma \left(\mathbf{B} \begin{bmatrix} \mathbf{z} \\ \mathbf{x} \\ \mathbf{q} \end{bmatrix} - \begin{bmatrix} \mathbf{b} \\ \tilde{\mathbf{q}} \end{bmatrix} \right). \tag{22}$$

Algorithm Complexity: Given that the number of iterations until ADMM convergence is not a function of input size, it is $\mathcal{O}(1)$. The most time-consuming step in each ADMM iteration is the solving of linear system (18) via CG in $\mathcal{O}(N)$. Thus, we conclude that solving SLP (12) using the aforementioned ADMM algorithm is $\mathcal{O}(N)$.

Algorithm Comparison: Comparing the CG algorithm used to solve linear system (8) and the ADMM algorithm developed to solve SLP (12), we first observe that, given a similarity graph $\mathcal{G}$ specified by Laplacian or incidence matrix, $\mathbf{L}$ or $\mathbf{C}$, both algorithms compute the interpolated signal $\mathbf{x}^*$ as a low-pass filtered output of the up-sampled input $\mathbf{H}^\top \mathbf{y}$ in (10) and (18), respectively. This is intuitive, given the assumed graph smoothness priors, GLR and GTV. We see also that the ADMM algorithm is more intricate: in each iteration, the main variables are computed using CG, while the auxiliary variable is updated via ReLU-like thresholding. As a result, the ADMM algorithm is more amenable to deep algorithm unrolling with better performance in general (see Section 6 for details).

5 Graph Learning & Algorithm Unrolling

We now discuss how a similarity graph $\mathcal{G}$ can be learned from data, specified by graph Laplacian $\mathbf{L}$ for GLR minimization (4) or incidence matrix $\mathbf{C}$ for GTV minimization (11), so that the two proposed graph-based interpolations can take place. Moreover, we show how a normalized graph learning module[4] performs comparable operations to the self-attention mechanism in conventional transformers. Thus, unrolling sequential pairs of graph-based iterative algorithm and graph learning module back-to-back leads to an interpretable "white-box" transformer-like neural net.

[4]While estimating a precision (graph Laplacian) matrix from an empirical covariance matrix computed from data is another graph learning approach [39, 40, 41], we pursue a feature-based approach here [27, 28].

5.1 Self-Attention Operator in Transformer

We first review the self-attention operator in a conventional transformer architecture, defined using a transformed dot product and a softmax operation [1]. Specifically, first denote by $\mathbf{x}_i \in \mathbb{R}^E$ an *embedding* for token i of N tokens. *Affinity* $e(i,j)$ between tokens i and j is defined as the dot product between linear-transformed embeddings $\mathbf{K}\mathbf{x}_i$ and $\mathbf{Q}\mathbf{x}_j$, where $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{E \times E}$ are the *query* and *key* matrices, respectively. Using softmax, a non-linear function that maps a vector of real numbers to a vector of positive numbers that sum to 1, *attention weight* $a_{i,j}$ is computed as

$$a_{i,j} = \frac{\exp(e(i,j))}{\sum_{l=1}^{N} \exp(e(i,l))}, \qquad e(i,j) = (\mathbf{Q}\mathbf{x}_j)^\top (\mathbf{K}\mathbf{x}_i). \tag{23}$$

Given self-attention weights $a_{i,j}$, output embedding $\mathbf{y}_i$ for token i is computed as

$$\mathbf{y}_i = \sum_{l=1}^{N} a_{i,l} \mathbf{x}_l \mathbf{V} \tag{24}$$

where $\mathbf{V} \in \mathbb{R}^{E \times E}$ is a *value* matrix. "Self-attention" here means that input embeddings are weighted to compute output embeddings. A transformer is thus a sequence of embedding-to-embedding mappings via different learned self-attention operations defined by $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ matrices. *Multi-head* attention is possible when multiple query and key matrices $\mathbf{Q}^{(m)}$ and $\mathbf{K}^{(m)}$ are used to compute different attention weights $a_{i,j}^{(m)}$'s for the same input embeddings $\mathbf{x}_i$ and $\mathbf{x}_j$, and the output embedding $\mathbf{y}_i$ is computed using an average of these multi-head attention weights $a_{i,l}^{(m)}$'s.

5.2 Computation of Graph Edge Weights

Consider now how edge weights $w_{i,j}$'s can be computed from data to specify a finite graph $\mathcal{G}$ [27, 28]. A low-dimensional *feature vector* $\mathbf{f}_i \in \mathbb{R}^D$ can be computed for each node i from embedding $\mathbf{x}_i \in \mathbb{R}^E$ via some (possibly non-linear) function $\mathbf{f}_i = F(\mathbf{x}_i)$, where typically $D \ll E$. Edge weight $w_{i,j}$ between nodes i and j in a graph $\mathcal{G}$ can then be computed as

$$w_{i,j} = \exp\left(-d(i,j)\right), \qquad d(i,j) = (\mathbf{f}_i - \mathbf{f}_j)^\top \mathbf{M} (\mathbf{f}_i - \mathbf{f}_j) \tag{25}$$

where $d(i,j)$ is the squared *Mahalanobis distance* given PSD *metric matrix* $\mathbf{M}$ that quantifies the difference between nodes i and j. M edge weights $\{w_{i,j}\}$ compose a graph $\mathcal{G}$, specified by the Laplacian matrix $\mathbf{L}$ for GLR minimization (4) and the incidence matrix $\mathbf{C}^{M \times N}$ for GTV minimization (11). Because $w_{i,j} \geq 0, \forall i,j$, constructed graph $\mathcal{G}$ is positive.

As a concrete example, consider *bilateral filter* (BF) weights commonly used in image filtering [42], where feature $\mathbf{f}_i$ contains the 2D grid location $\mathbf{l}_i$ and color intensity p_i of pixel i, and metric $\mathbf{M} = \text{diag}([1/\sigma_d^2; 1/\sigma_r^2])$ is a diagonal matrix with weights to specify the relative strength of the *domain* and *range* filters in BF. Because BF uses input pixel intensities p_l's to compute weighted output pixel intensities p_i's, BF is *signal-dependent*, similar to self-attention weights in transformers.

Edge weights are often first *normalized* before being used for filtering.

Normalization: For normalization, the symmetric *normalized graph Laplacian* $\mathbf{L}_n$ is defined as $\mathbf{L}_n \triangleq \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, so that the diagonal entries of $\mathbf{L}_n$ are all ones (assuming $\mathcal{G}$ is connected and positive) [18]. We assume normalized $\mathbf{L}_n$ is used for Laplacian $\mathbf{L}$ in GLR minimization in (4).

Alternatively, the asymmetric *random walk graph Laplacian* $\mathbf{L}_{rw}$ is defined as $\mathbf{L}_{rw} \triangleq \mathbf{D}^{-1}\mathbf{L}$, so that the sum of each row of $\mathbf{L}_{rw}$ equals to zero [18]. Interpreting $\mathbf{L}_{rw}$ as a Laplacian matrix to a *directed* graph, the weight sum of edges leaving each node i is one, *i.e.*, $\sum_{l|(i,l)\in\mathcal{E}} \bar{w}_{i,l} = 1, \forall i$. To accomplish this, undirected edges weights $\{w_{i,j}\}$ are normalized to $\{\bar{w}_{i,j}\}$ via

$$\bar{w}_{i,j} = \frac{\exp(-d(i,j))}{\sum_{l|(i,l)\in\mathcal{E}} \exp(-d(i,l))}. \tag{26}$$

For GTV minimization in (11), we normalize edge weights in incidence matrix $\mathbf{C}$ instead using (26). This results in normalized $\bar{\mathbf{C}} \in \mathbb{R}^{2M \times N}$ for a *directed* graph with $2M$ directed edges. Subsequently,

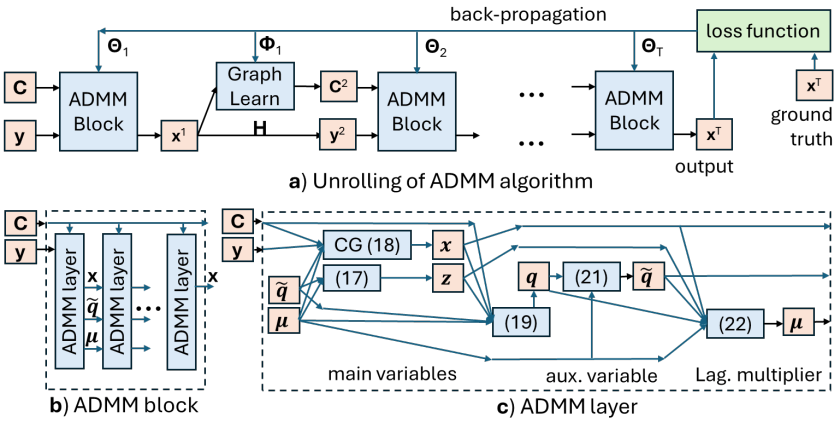


Figure 1: Unrolling of GTV-based signal interpolation algorithm.

we define symmetric graph Laplacian $\bar{\mathbf{L}} = \bar{\mathbf{C}}^\top \bar{\mathbf{C}}$ and generalized graph Laplacian $\bar{\mathcal{L}} = \bar{\mathbf{L}} + \mathbf{H}^\top \mathbf{H}$. Note that $\|\bar{\mathbf{C}}\mathbf{1}\|_1 = \sum_{l=1}^{N} \bar{w}_{i,l} \, |1-1| = 0$ after normalization, as expected for a total variation term on a constant signal $\mathbf{1}$. Further, note that while $\bar{\mathbf{C}}$ is an incidence matrix for a directed graph with $2M$ edges, $\bar{\mathbf{L}}$ is a graph Laplacian for an undirected graph with M edges. See Fig. 3 in Appendix A.6 for an example of incidence matrix $\mathbf{C}$, normalized incidence matrix $\bar{\mathbf{C}}$, and graph Laplacian matrix $\bar{\mathbf{L}}$.

Comparison to Self-Attention Operator: We see how the definitions of edge weights (25) and normalization (26) are similar to attention weights in (23). Specifically, *interpreting the negative squared Mahalanobis distance* $-d(i,j)$ *as affinity* $e(i,j)$*, normalized edge weights* $\bar{w}_{i,j}$ *in* (25) *are essentially the same as attention weights* $a_{i,j}$ *in* (23). There are subtle but important differences: i) how non-negative Mahalanobis distance $d(i,j)$ is computed in (25) using features $\mathbf{f}_i = F(\mathbf{x}_i)$ and metric $\mathbf{M}$ versus how real-valued affinity is computed via a transformed dot product in (23), and ii) how the normalization term is computed in a one-hop neighborhood from node i in (26) versus how it is computed using all N tokens in (23). The first difference conceptually means that edge weight based on Mahalanobis distance $d(i,j)$ is symmetric (*i.e.*, $\bar{w}_{i,j} = \bar{w}_{j,i}$), while attention weight $a_{i,j}$ is not. Both differences have crucial complexity implications, which we will revisit in the sequel.

Further, we note that, given a graph $\mathcal{G}$, the interpolated signal $\mathbf{x}^*$ is computed simply as a low-pass filtered output of the up-sampled input observation $\mathbf{H}^\top \mathbf{y}$ via (10) or (18), depending on the assumed graph smoothness prior, GLR or GTV, while the output embedding $\mathbf{y}_i$ in a conventional transformer requires value matrix $\mathbf{V}$ in (24). This also has a complexity implication.

5.3 Deep Algorithm Unrolling

We unroll T sequential pairs of an iterative interpolation algorithm (GLR- or GTV-based) with a graph learning module into an interpretable neural net. See Fig. 1a for an illustration of the GTV-based algorithm unrolling, where the t-th pair of ADMM block and the graph learning module have respective parameters $\mathbf{\Theta}_t$ and $\mathbf{\Phi}_t$ that are learned from back-propagation via a defined loss function. $\mathbf{\Phi}_t$ include parameters used to define feature function $F(\cdot)$ and metric matrix $\mathbf{M}$ in (25), so the module can construct a graph $\mathcal{G}$ specified by incidence matrix $\mathbf{C}^{t+1}$ given signal $\mathbf{x}^t$. In our implementation, we employ a shallow CNN to map a neighborhood of pixels centered at pixel i to a low-dimensional feature $\mathbf{f}_i$, with a parameter size smaller than query and key matrices, $\mathbf{Q}$ and $\mathbf{K}$, in a conventional transformer. See Section 6.1 for details.

An ADMM block contains multiple ADMM layers that are unrolled iterations of the iterative ADMM algorithm described in Section 4.2. Each ADMM layer updates the main variables $\mathbf{x}, \mathbf{z}, \mathbf{q}$, auxiliary variable $\tilde{\mathbf{q}}$, and Lagrange multiplier $\boldsymbol{\mu}$ in turn using (17) to (22). ADMM weight parameter γ, as well as parameters in CG used to compute linear system (18), are learned via back-propagation. Specifically, two CG parameters α and β that represent step size and momentum during the conjugate gradient descent step are learned. See Appendix A.7 for details.

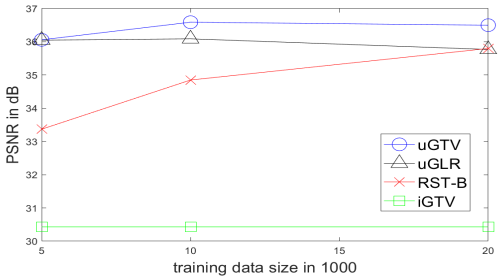


Figure 2: Demosaicking performance vs. training size for different models.

Method	McM PSNR			
	$\sigma = 10$	$\sigma = 20$	$\sigma = 30$	$\sigma = 50$
RST-B	28.01	22.7	19.34	15.03
uGLR	28.24	22.84	19.49	15.203
uGTV	**28.31**	**22.89**	**19.56**	**15.38**

Table 1: Demosaicking performance in noisy scenario, where models are trained on noiseless dataset.

6 Experiments

6.1 Experimental Setup

All models were developed using Python 3.11. We leveraged PyTorch to implement all models and trained them using NVIDIA GeForce RTX 2080 Ti. To train each learned model, we used the DIV2K dataset, which contains 800 and 100 high-resolution (HR) training and validation images, respectively. Since the images are HR, we patchified the images into small images and used only about 1 to 4% of the patches for training and validation sets. We randomly sampled patches of 64×64 pixels to train the model. To test a model, we used the McM [43], Kodak [44], and Urban100 [45] datasets, running each model on the whole images. See Appendix A.8 for more implementation details.

We tested model performance in two imaging applications: demosaicking and image interpolation. Demosaicking reconstructs a full-color image (each pixel contains RGB colors) from a Bayer-patterned image, where each pixel location has only one of Red, Green, or Blue. Interpolation reconstructs empty pixels missing all three colors in an image. To create input images, for the first application, we started from a full-color image and then removed color components per pixel according to the Bayer pattern. For the second application, we directly down-sampled horizontally and vertically a HR image by a factor of 2 to get the corresponding low-resolution (LR) image without any anti-aliasing filtering. This is equivalent to keeping every four pixels in the HR image.

6.2 Experimental Results

For the first application, we evaluated our graph-based models against two variants of RSTCANet [46], RSTCANet-B and RSTCANet-S (RST-B and RST-S for short), a SOTA framework that employs a swin transformer architecture, Menon [47], Malvar [48] and bicubic interpolation. Menon [47] employs a directional approach combined with an *a posteriori* decision, followed by an additional refinement step. Malvar [48] uses a linear filtering technique that incorporates inter-channel information across all channels for demosaicking.

The baselines for our second application are MAIN [49], a multi-scale deep learning framework for image interpolation, SwinIR [50], and bicubic interpolation. SwinIR consists of three stages: shallow feature extraction, deep feature extraction, and a final reconstruction stage; see [50] for details. We use Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM) [51] as our evaluation metrics, common in image quality assessment.

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**	**0.8801**

Table 3: Interpolation performance for different models, trained on 10k sample dataset.

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**

Table 2: Demosaicking performance for different models, trained on 10k sample dataset.

Table 2 shows the demosaicking performance for different models, where all models were trained on the same dataset and the same number of epochs (30), using a subset of DIV2K dataset containing $10K$ of 64×64 patches. We observe that our unrolled GTV model (uGTV) achieved the best overall performance, while the unrolled GLR model (uGLR) and RST-S performed similarly. Both our models (uGTV and uGLR) performed better than RST-B while employing significantly fewer parameters. Crucially, we observe that although our normalized edge weight $\bar{w}_{i,j}$ based on Mahalanobis distance is symmetric while the self-attention weight $a_{i,j}$ is not (due to query and key matrices $\mathbf{Q}$ and $\mathbf{K}$ not being the same in general), the directionality in the self-attention mechanism does not appear to help improve performance of the conventional transformer further, at least for image interpolation tasks. The iterative GTV algorithm (iGTV) without parameter optimization performed the worst, demonstrating the importance of parameter learning.

In Fig. 2, we see the demosaicking performance of different models versus training data size. We see that for small data size, our models (uGTV and uGLR) performed significantly better than RST-B. This is intuitive, since a model with more parameters requires more training data in general. See Appendix A.9 for example visual results.

Next, we test robustness to covariate shift by testing models trained on noiseless data using a dataset artificially injected with Gaussian noise. Table 1 shows the demosaicking performance versus different noise variances. We observe that our models outperformed RST-B in all noisy scenarios.

For image interpolation, we interploated a LR image to a corresponding HR image. We trained all models on the same dataset as the first application with the same number of epochs (15). Table 3 shows that under the same training conditions, our proposed models (uGTV and uGLR) outperformed MAIN in interpolation performance in all three benchmark datasets by about 0.7 dB. Note that for this application we only interpolated Y-channel from YCbCr color space for PSNR computation. Similar to the first application, our models achieved slight performance gain while employing drastically fewer parameters; specifically, uGTV employed only about 3% of the parameters in MAIN.

7 Conclusion

By unrolling iterative algorithms that minimize one of two graph smoothness priors—ℓ_2-norm graph Laplacian regularizer (GLR) or ℓ_1-norm graph total variation (GTV)—we build interpretable and light-weight transformer-like neural nets for the signal interpolation problem. The key insight is that the normalized graph learning module is akin to the self-attention mechanism in a conventional transformer architecture. Moreover, the interpolated signal in each layer is simply the low-pass filtered output derived from the assumed graph smoothness prior, eliminating the need for the value matrix. Experiments in two imaging applications show that interpolation results on par with SOTA can be achieved with a fraction of the parameters used in conventional transformers.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in neural information*

processing systems, vol. 30, 2017.

[3] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9992–10002.

[4] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai, "An image is worth 16x16 words: Transformers for image recognition at scale," in *The Ninth International Conference on Learning Representations (ICLR)*, 2021.

[5] James Vuckovic, Aristide Baratin, and Rémi Tachet des Combes, "A mathematical theory of attention," *ArXiv*, vol. abs/2007.02876, 2020.

[6] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra, "Why are adaptive methods good for attention models?," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds. 2020, vol. 33, pp. 15383–15393, Curran Associates, Inc.

[7] Charles Burton Snell, Ruiqi Zhong, Dan Klein, and Jacob Steinhardt, "Approximating how single head attention learns," *ArXiv*, vol. abs/2103.07601, 2021.

[8] Colin Wei, Yining Chen, and Tengyu Ma, "Statistically meaningful approximation: a case study on approximating turing machines with transformers," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds. 2022, vol. 35, pp. 12071–12083, Curran Associates, Inc.

[9] Hyunjik Kim, George Papamakarios, and Andriy Mnih, "The lipschitz constant of self-attention," in *Proceedings of the 38th International Conference on Machine Learning*, Marina Meila and Tong Zhang, Eds. 18–24 Jul 2021, vol. 139 of *Proceedings of Machine Learning Research*, pp. 5562–5571, PMLR.

[10] Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang, "Inductive biases and variable creation in self-attention mechanisms," in *Proceedings of the 39th International Conference on Machine Learning*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, Eds. 17–23 Jul 2022, vol. 162 of *Proceedings of Machine Learning Research*, pp. 5793–5831, PMLR.

[11] Hongkang Li, Meng Wang, Sijia Liu, and Pin-Yu Chen, "A theoretical understanding of shallow vision transformers: Learning, generalization, and sample complexity," in *The Eleventh International Conference on Learning Representations*, 2023.

[12] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme Ruiz, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd Van Steenkiste, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Collier, Alexey A. Gritsenko, Vighnesh Birodkar, Cristina Nader Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetic, Dustin Tran, Thomas Kipf, Mario Lucic, Xiaohua Zhai, Daniel Keysers, Jeremiah J. Harmsen, and Neil Houlsby, "Scaling vision transformers to 22 billion parameters," in *Proceedings of the 40th International Conference on Machine Learning*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, Eds. 23–29 Jul 2023, vol. 202 of *Proceedings of Machine Learning Research*, pp. 7480–7512, PMLR.

[13] Ruiqi Zhang, Spencer Frei, and Peter L. Bartlett, "Trained transformers learn linear models in-context," *Journal of Machine Learning Research*, vol. 25, no. 49, pp. 1–55, 2024.

[14] Vishal Monga, Yuelong Li, and Yonina C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, 2021.

[15] Karol Gregor and Yann LeCun, "Learning fast approximations of sparse coding," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, Madison, WI, USA, 2010, ICML'10, p. 399–406, Omnipress.

[16] Yaodong Yu, Sam Buchanan, Druv Pai, Tianzhe Chu, Ziyang Wu, Shengbang Tong, Benjamin Haeffele, and Yi Ma, "White-box transformers via sparse rate reduction," in *Advances in Neural Information Processing Systems*, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds. 2023, vol. 36, pp. 9422–9457, Curran Associates, Inc.

[17] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," in *IEEE Signal Processing Magazine*, May 2013, vol. 30, no.3, pp. 83–98.

[18] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," in *Proceedings of the IEEE*, May 2018, vol. 106, no.5, pp. 808–828.

[19] G. Cheung, E. Magli, Y. Tanaka, and M. Ng, "Graph spectral image processing," in *Proceedings of the IEEE*, May 2018, vol. 106, no.5, pp. 907–930.

[20] J. Pang and G. Cheung, "Graph Laplacian regularization for inverse imaging: Analysis in the continuous domain," in *IEEE Transactions on Image Processing*, April 2017, vol. 26, no.4, pp. 1770–1785.

[21] X. Liu, G. Cheung, X. Wu, and D. Zhao, "Random walk graph Laplacian based smoothness prior for soft decoding of JPEG images," *IEEE Transactions on Image Processing*, vol. 26, no.2, pp. 509–524, February 2017.

[22] Fei Chen, Gene Cheung, and Xue Zhang, "Manifold graph signal restoration using gradient graph Laplacian regularizer," *IEEE Transactions on Signal Processing*, vol. 72, pp. 744–761, 2024.

[23] Jin Zeng, Jiahao Pang, Wenxiu Sun, and Gene Cheung, "Deep graph Laplacian regularization for robust denoising of real images," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 1759–1768.

[24] Chinthaka Dinesh, Gene Cheung, and Ivan V. Bajić, "Point cloud denoising via feature graph laplacian regularization," *IEEE Transactions on Image Processing*, vol. 29, pp. 4143–4158, 2020.

[25] Chinthaka Dinesh, Gene Cheung, and Ivan V. Bajić, "Point cloud video super-resolution via partial point coupling and graph smoothness," *IEEE Transactions on Image Processing*, vol. 31, pp. 4117–4132, 2022.

[26] Yongyi Yang, Tang Liu, Yangkun Wang, Jinjing Zhou, Quan Gan, Zhewei Wei, Zheng Zhang, Zengfeng Huang, and David Wipf, "Graph neural networks inspired by classical iterative algorithms," in *Proceedings of the 38th International Conference on Machine Learning*, Marina Meila and Tong Zhang, Eds. 18–24 Jul 2021, vol. 139 of *Proceedings of Machine Learning Research*, pp. 11773–11783, PMLR.

[27] Wei Hu, Xiang Gao, Gene Cheung, and Zongming Guo, "Feature graph learning for 3D point cloud denoising," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2841–2856, 2020.

[28] Cheng Yang, Gene Cheung, and Wei Hu, "Signed graph metric learning via Gershgorin disc perfect alignment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 7219–7234, 2022.

[29] Abderrahim Elmoataz, Olivier Lezoray, and SÉbastien Bougleux, "Nonlocal discrete regularization on weighted graphs: A framework for image and manifold processing," *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1047–1060, 2008.

[30] Camille Couprie, Leo Grady, Laurent Najman, Jean-Christophe Pesquet, and Hugues Talbot, "Dual constrained tv-based regularization on graphs," *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1246–1273, 2013.

[31] Peter Berger, Gabor Hannak, and Gerald Matz, "Graph signal recovery via primal-dual algorithms for total variation minimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 842–855, 2017.

[32] J. R Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Tech. Rep., USA, 1994.

[33] Sinong Wang and Ness Shroff, "A new alternating direction method for linear programming," in *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio,

H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. 2017, vol. 30, Curran Associates, Inc.

[34] Y. Bai, G. Cheung, X. Liu, and W. Gao, "Graph-based blind image deblurring from a single photograph," *IEEE Transactions on Image Processing*, vol. 28, no.3, pp. 1404–1418, 2019.

[35] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge, 2004.

[36] G. Golub and C. F. Van Loan, *Matrix Computations (Johns Hopkins Studies in the Mathematical Sciences)*, Johns Hopkins University Press, 2012.

[37] E. BrianDavies, Graham M. L. Gladwell, Josef Leydold, Peter F. Stadler, and Peter F. Stadler, "Discrete nodal domain theorems," *Linear Algebra and its Applications*, vol. 336, pp. 51–60, 2000.

[38] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," in *Foundations and Trends in Optimization*, 2011, vol. 3, no.1, pp. 1–122.

[39] H. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under Laplacian and structural constraints," in *IEEE Journal of Selected Topics in Signal Processing*, July 2017, vol. 11, no.6, pp. 825–841.

[40] Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 44–63, 2019.

[41] Saghar Bagheri, Tam Thuc Do, Gene Cheung, and Antonio Ortega, "Spectral graph learning with core eigenvectors prior via iterative GLASSO and projection," *IEEE Transactions on Signal Processing*, vol. 72, pp. 3958–3972, 2024.

[42] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the IEEE International Conference on Computer Vision*, Bombay, India, 1998.

[43] Lei Zhang, Xiaolin Wu, Antoni Buades, and Xin Li, "Color demosaicking by local directional interpolation and nonlocal adaptive thresholding," *Journal of Electronic imaging*, vol. 20, no. 2, pp. 023016–023016, 2011.

[44] Eastman Kodak, "Kodak lossless true color image suite (photocd pcd0992)," *URL http://r0k. us/graphics/kodak*, vol. 6, pp. 2, 1993.

[45] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5197–5206.

[46] Wenzhu Xing and Karen Egiazarian, "Residual swin transformer channel attention network for image demosaicing," in *2022 10th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 2022, pp. 1–6.

[47] Daniele Menon, Stefano Andriani, and Giancarlo Calvagno, "Demosaicing with directional filtering and a posteriori decision," *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 132–141, 2007.

[48] H.S. Malvar, Li wei He, and R. Cutler, "High-quality linear interpolation for demosaicing of bayer-patterned color images," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2004, vol. 3, pp. iii–485.

[49] Jiahuan Ji, Baojiang Zhong, and Kai-Kuang Ma, "Image interpolation using multi-scale attention-aware inception network," *IEEE Transactions on Image Processing*, vol. 29, pp. 9413–9428, 2020.

[50] Jingyun Liang, Jiezhang Cao, Guolei Sun, Kai Zhang, Luc Van Gool, and Radu Timofte, "Swinir: Image restoration using swin transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2021, pp. 1833–1844.

[51] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[52] Yurii Nesterov, *Introductory lectures on convex optimization: A basic course*, vol. 87, Springer Science & Business Media, 2013.

A Appendix

A.1 Full-Rankness of Matrix $\mathbf{P}$ in (4)

Given that the underlying graph $\mathcal{G}$ is positive and connected, we prove that coefficient matrix $\mathbf{P}$ in (4) is full-rank and thus invertible. We prove by contradiction: suppose $\mathbf{P}$ is *not* full-rank, and there exists a vector $\mathbf{v} = [\mathbf{x};\ \boldsymbol{\mu}]$ such that $\mathbf{P}\mathbf{v} = \mathbf{0}_{N+K}$. Suppose we order K sampled entries $\mathbf{x}_{\mathcal{S}}$ before $N-K$ non-sampled entries $\mathbf{x}_{\bar{\mathcal{S}}}$ in $\mathbf{x}$, *i.e.*, $\mathbf{x} = [\mathbf{x}_{\mathcal{S}}; \mathbf{x}_{\bar{\mathcal{S}}}]$. First, given sampling matrix $\mathbf{H} = [\mathbf{I}_K\ \ \mathbf{0}_{K,N-K}] \in \{0,1\}^{K\times N}$, focusing on the second block row of $\mathbf{P}$, $[\mathbf{H}\ \ \mathbf{0}_{K,K}]\mathbf{v} = \mathbf{0}_K$ means $\mathbf{H}\mathbf{x} = [\mathbf{I}_k\ \mathbf{0}_{K,N-K}][\mathbf{x}_{\mathcal{S}}; \mathbf{x}_{\bar{\mathcal{S}}}] = \mathbf{0}_K$. Thus, sampled entries of $\mathbf{x}$ must be zeros, *i.e.*, $\mathbf{x}_{\mathcal{S}} = \mathbf{0}_K$.

Second, suppose we write Laplacian $\mathbf{L}$ in blocks, *i.e.*, $\mathbf{L} = [\mathbf{L}_{\mathcal{S},\mathcal{S}}\ \mathbf{L}_{\mathcal{S},\bar{\mathcal{S}}}; \mathbf{L}_{\bar{\mathcal{S}},\mathcal{S}}\ \mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}]$. Then, the non-sampled rows of the first block row of $\mathbf{M}\mathbf{v}$ are

$$\left([2\mathbf{L}\ \ \mathbf{H}^\top]\begin{bmatrix}\mathbf{x}\\ \boldsymbol{\mu}\end{bmatrix}\right)_{\bar{\mathcal{S}}} = 2\left(\mathbf{L}_{\bar{\mathcal{S}},\mathcal{S}}\mathbf{x}_{\mathcal{S}} + \mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\mathbf{x}_{\bar{\mathcal{S}}}\right) \tag{27}$$

where $(\mathbf{H}^\top\boldsymbol{\mu})_{\bar{\mathcal{S}}} = \mathbf{0}_{N-K}$ since non-sampled rows of $\mathbf{H}^\top = [\mathbf{I}_K; \mathbf{0}_{N-K,K}]$ are all zeros. From above, we know $\mathbf{x}_{\mathcal{S}} = \mathbf{0}_K$, and thus $([2\mathbf{L}\ \ \mathbf{H}^\top]\mathbf{v})_{\bar{\mathcal{S}}} = \mathbf{0}_{N-K}$ implies that we require $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\mathbf{x}_{\bar{\mathcal{S}}} = \mathbf{0}_N$. However, since $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ is a combinatorial Laplacian matrix for a positive sub-graph connecting non-sampled nodes *plus* at least one strictly positive self-loop (representing an edge from a non-sampled node to a sample node), given $\mathcal{G}$ is a connected positive graph, $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ must be PD (see Appendix A.2 below). Thus, $\nexists \mathbf{x}_{\bar{\mathcal{S}}} \neq \mathbf{0}$ s.t. $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\mathbf{x}_{\bar{\mathcal{S}}} = \mathbf{0}_{N-K}$, a contradiction. Therefore, we can conclude that $\mathbf{P}$ must be full-rank.

A.2 Positive Definiteness of Matrix $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ in (8)

Given that the underlying graph $\mathcal{G}$ is positive and connected, we prove that coefficient matrix $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ in (8) is PD. By definition $\mathbf{L} \triangleq \text{diag}(\mathbf{W}\mathbf{1}_N) - \mathbf{W}$, where $\mathbf{W}$ is an adjacency matrix for a positive graph without self-loops, *i.e.*, $W_{i,j} \geq 0, \forall i \neq j$ and $W_{i,i} = 0, \forall i$. Thus, $L_{i,i} = \sum_j W_{i,j}, \forall i$, and $L_{i,j} = -W_{i,j} \leq 0, \forall i \neq j$. For sub-matrix $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$, $L_{i,i} = \sum_{j\in\bar{\mathcal{S}}} W_{i,j} + \sum_{j\in\mathcal{S}} W_{i,j}, \forall i \in \bar{\mathcal{S}}$. Define $\mathbf{L}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ as a graph Laplacian matrix for nodes in $\bar{\mathcal{S}}$ considering only edges between nodes in $\bar{\mathcal{S}}$, *i.e.*, $L_{i,i} = \sum_{j\in\bar{\mathcal{S}}} W_{i,j}, \forall i \in \bar{\mathcal{S}}$. Define $\mathbf{D}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ as a diagonal degree matrix for nodes in $\bar{\mathcal{S}}$ considering only edges between $\bar{\mathcal{S}}$ and $\mathcal{S}$, *i.e.*, $D'_{i,i} = \sum_{j\in\mathcal{S}} W_{i,j}, \forall i \in \bar{\mathcal{S}}$. Note that $D'_{i,i} \geq 0, \forall i$. We can now write $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ as

$$\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}} = \mathbf{L}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}} + \mathbf{D}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}}. \tag{28}$$

$\mathbf{L}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ is a combinatorial graph Laplacian for a positive graph without self-loops, and thus is provably PSD [19]. $\mathbf{D}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ is a non-negative diagonal matrix, and thus is also PSD. By Weyl's inequality, $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ is also PSD.

We prove by contradiction: suppose $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ is not PD, and $\exists \mathbf{x} \neq \mathbf{0}$ such that $\mathbf{x}^\top\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\mathbf{x} = 0$. $\mathbf{x}^\top\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\mathbf{x} = 0$ iff $\mathbf{x}^\top\mathbf{L}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\mathbf{x} = 0$ and $\mathbf{x}^\top\mathbf{D}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\mathbf{x} = 0$ simultaneously. Denote by $\bar{\mathcal{S}}_1$ and $\bar{\mathcal{S}}_2$ the indices of nodes in $\bar{\mathcal{S}}$ with and without connections to nodes in $\mathcal{S}$, respectively. $\bar{\mathcal{S}}_1 \neq \emptyset$, since $\mathcal{G}$ is connected. Suppose first $\bar{\mathcal{S}}_2 = \emptyset$. Then $\mathbf{D}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ has strictly positive diagonal entries and is PD, and there is no $\mathbf{x} \neq \mathbf{0}$ s.t. $\mathbf{x}^\top\mathbf{D}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\mathbf{x} = \mathbf{0}$, a contradiction.

Suppose now $\bar{\mathcal{S}}_2 \neq \emptyset$. First, $\mathbf{x}^\top\mathbf{D}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\mathbf{x} = 0$ implies $\mathbf{x}_{\bar{\mathcal{S}}_1} = \mathbf{0}$. Then,

$$\mathbf{x}^\top\mathbf{L}'_{\bar{\mathcal{S}},\bar{\mathcal{S}}}\mathbf{x} = \sum_{i,j\in\bar{\mathcal{S}}} W_{i,j}(x_i - x_j)^2 \geq \sum_{i\in\bar{\mathcal{S}}_1, j\in\bar{\mathcal{S}}_2} W_{i,j}(x_i - x_j)^2. \tag{29}$$

Since each term in the sum is non-negative, the sum is zero only if for each $(i,j) \in \mathcal{E}$ where $i \in \bar{\mathcal{S}}_1$ and $j \in \bar{\mathcal{S}}_2$, $0 = x_i = x_j$. For nodes $k \in \bar{\mathcal{S}}_2$ connected only to nodes $j \in \bar{\mathcal{S}}_2$ connected to $i \in \bar{\mathcal{S}}_1$, $x_k = x_j = x_i = 0$ necessarily, and for nodes $l \in \bar{\mathcal{S}}_2$ connected to $k \in \bar{\mathcal{S}}_2$ must have $x_l = x_k = 0$, and so on. Thus, $\mathbf{x} = \mathbf{0}$, a contradiction. Thus, we can conclude that $\mathbf{L}_{\bar{\mathcal{S}},\bar{\mathcal{S}}}$ is PD.

A.3 Derivation of (17), (18) and (19)

We define $\boldsymbol{\phi} = [\mathbf{z}; \mathbf{x}; \mathbf{q}_1; \mathbf{q}_2]$ and rewrite the objective (16) to

$$\min_{\boldsymbol{\phi}} \begin{bmatrix} \mathbf{1}_M \\ \mathbf{0}_{N+2M} \end{bmatrix}^\top \boldsymbol{\phi} + (\boldsymbol{\mu}^t)^\top \left(\begin{bmatrix} \mathbf{A} \\ \mathbf{0}_{2M,M+N}\ \mathbf{I}_{2M} \end{bmatrix} \boldsymbol{\phi} - \begin{bmatrix} \mathbf{b} \\ \tilde{\mathbf{q}}^t \end{bmatrix} \right) + \frac{\gamma}{2} \left\| \underbrace{\begin{bmatrix} \mathbf{A} \\ \mathbf{0}_{2M,M+N}\ \mathbf{I}_{2M} \end{bmatrix}}_{\mathbf{B}} \boldsymbol{\phi} - \begin{bmatrix} \mathbf{b} \\ \tilde{\mathbf{q}}^t \end{bmatrix} \right\|_2^2. \tag{30}$$

(30) is a convex quadratic objective, and so we take the derivative w.r.t. $\boldsymbol{\phi}$ and set it to $\mathbf{0}$:

$$\begin{bmatrix} \mathbf{1}_M \\ \mathbf{0}_{N+2M} \end{bmatrix} + \begin{bmatrix} \mathbf{A}^\top & \mathbf{0}_{M+N,2M} \\ & \mathbf{I}_{2M} \end{bmatrix} (\boldsymbol{\mu}^t) + \frac{\gamma}{2} \left(2 \begin{bmatrix} \mathbf{A}^\top & \mathbf{0}_{M+N,2M} \\ & \mathbf{I}_{2M} \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \mathbf{0}_{2M,M+N}\ \mathbf{I}_{2M} \end{bmatrix} \boldsymbol{\phi} - 2 \begin{bmatrix} \mathbf{A}^\top & \mathbf{0}_{M+N,2M} \\ & \mathbf{I}_{2M} \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \tilde{\mathbf{q}}^t \end{bmatrix} \right) = \mathbf{0}. \tag{31}$$

Given that $\mathbf{B}$ is the following matrix:

$$\mathbf{B} = \begin{bmatrix} \mathbf{I}_M & -\mathbf{C} & -\mathbf{I}_M & \mathbf{0}_M \\ \mathbf{I}_M & \mathbf{C} & \mathbf{0}_M & -\mathbf{I}_M \\ \mathbf{0}_{K,M} & \mathbf{H} & \mathbf{0}_{K,M} & \mathbf{0}_{K,M} \\ \mathbf{0}_{M,M} & \mathbf{0}_{M,N} & \mathbf{I}_M & \mathbf{0}_M \\ \mathbf{0}_{M,M} & \mathbf{0}_{M,N} & \mathbf{0}_M & \mathbf{I}_M \end{bmatrix} \tag{32}$$

Hence, $\mathbf{B}^\top \mathbf{B}$ is

$$\mathbf{B}^\top \mathbf{B} = \begin{bmatrix} 2\mathbf{I}_M & \mathbf{0}_{M,N} & -\mathbf{I}_M & -\mathbf{I}_M \\ \mathbf{0}_{N,M} & 2\mathbf{C}^\top\mathbf{C} + \mathbf{H}^\top\mathbf{H} & \mathbf{C}^\top & -\mathbf{C}^\top \\ -\mathbf{I}_M & \mathbf{C} & 2\mathbf{I}_M & \mathbf{0}_{M,M} \\ -\mathbf{I}_M & -\mathbf{C} & \mathbf{0}_{M,M} & 2\mathbf{I}_M \end{bmatrix}. \tag{33}$$

Note that adding two of row 1 to rows 3 and 4, we get $[2\mathbf{I}_M\ \mathbf{0}_{M,N}\ \mathbf{0}_{M,M}\ \mathbf{0}_{M,M}]$.

Solving for $\boldsymbol{\phi}$ in (31), we get

$$\gamma \mathbf{B}^\top \mathbf{B} \boldsymbol{\phi} = -\begin{bmatrix} \mathbf{1}_M \\ \mathbf{0}_N \\ \mathbf{0}_M \\ \mathbf{0}_M \end{bmatrix} - \mathbf{B}^\top \left(\begin{bmatrix} \boldsymbol{\mu}_a^t \\ \boldsymbol{\mu}_b^t \\ \boldsymbol{\mu}_c^t \\ \boldsymbol{\mu}_d^t \\ \boldsymbol{\mu}_e^t \end{bmatrix} - \gamma \begin{bmatrix} \mathbf{0}_M \\ \mathbf{0}_M \\ \mathbf{y} \\ \tilde{\mathbf{q}}_1^t \\ \tilde{\mathbf{q}}_2^t \end{bmatrix} \right) \tag{34}$$

$$= \begin{bmatrix} -\mathbf{1}_M - \boldsymbol{\mu}_a^t - \boldsymbol{\mu}_b^t \\ \mathbf{C}^\top \boldsymbol{\mu}_a^t - \mathbf{C}^\top \boldsymbol{\mu}_b^t - \mathbf{H}^\top \boldsymbol{\mu}_c^t + \gamma \mathbf{H}^\top \mathbf{y} \\ \boldsymbol{\mu}_a^t - \boldsymbol{\mu}_d^t + \gamma \tilde{\mathbf{q}}_1^t \\ \boldsymbol{\mu}_b^t - \boldsymbol{\mu}_e^t + \gamma \tilde{\mathbf{q}}_2^t \end{bmatrix} \tag{35}$$

We can solve for $\mathbf{z}^{t+1}$ directly; by adding two of row 1 to rows 3 and 4 of (35), we get

$$\begin{aligned} 2\gamma \mathbf{z}^{t+1} &= -2(\mathbf{1}_M) - \boldsymbol{\mu}_a^t - \boldsymbol{\mu}_b^t - \boldsymbol{\mu}_d^t - \boldsymbol{\mu}_e^t + \gamma(\tilde{\mathbf{q}}_1^t + \tilde{\mathbf{q}}_2^t) \\ \mathbf{z}^{t+1} &= -\frac{1}{\gamma}\mathbf{1}_M - \frac{1}{2\gamma}\left(\boldsymbol{\mu}_a^t + \boldsymbol{\mu}_b^t + \boldsymbol{\mu}_d^t + \boldsymbol{\mu}_e^t\right) + \frac{1}{2}(\tilde{\mathbf{q}}_1^t + \tilde{\mathbf{q}}_2^t). \end{aligned} \tag{36}$$

Subtracting row 4 from row 3 of (35), we get

$$\begin{aligned} 2\gamma \mathbf{C}\mathbf{x}^{t+1} + 2\gamma(\mathbf{q}_1^{t+1} - \mathbf{q}_2^{t+1}) &= \boldsymbol{\mu}_a^t - \boldsymbol{\mu}_b^t - \boldsymbol{\mu}_d^t + \boldsymbol{\mu}_e^t + \gamma(\tilde{\mathbf{q}}_1^t - \tilde{\mathbf{q}}_2^t) \\ \gamma(\mathbf{q}_1^{t+1} - \mathbf{q}_2^{t+1}) &= -\gamma \mathbf{C}\mathbf{x}^{t+1} + \frac{1}{2}\left(\boldsymbol{\mu}_a^t - \boldsymbol{\mu}_b^t - \boldsymbol{\mu}_d^t + \boldsymbol{\mu}_e^t\right) + \frac{\gamma}{2}(\tilde{\mathbf{q}}_1^t - \tilde{\mathbf{q}}_2^t). \end{aligned} \tag{37}$$

Thus, row 2 can be rewritten as

$$\gamma(2\mathbf{C}^\top\mathbf{C}+\mathbf{H}^\top\mathbf{H})\mathbf{x}^{t+1}+\mathbf{C}^\top\gamma(\mathbf{q}_1^{t+1}-\mathbf{q}_2^{t+1})=\mathbf{C}^\top\boldsymbol{\mu}_a^t-\mathbf{C}^\top\boldsymbol{\mu}_b^t-\mathbf{H}^\top\boldsymbol{\mu}_c^t+\gamma\mathbf{H}^\top\mathbf{y}$$

$$\gamma(\mathbf{C}^\top\mathbf{C}+\mathbf{H}^\top\mathbf{H})\mathbf{x}^{t+1}=\frac{1}{2}\mathbf{C}^\top\left(\boldsymbol{\mu}_a^t-\boldsymbol{\mu}_b^t+\boldsymbol{\mu}_d^t-\boldsymbol{\mu}_e^t\right)-\mathbf{H}^\top\boldsymbol{\mu}_c^t$$

$$-\frac{\gamma}{2}\mathbf{C}^\top(\tilde{\mathbf{q}}_1^t-\tilde{\mathbf{q}}_2^t)+\gamma\mathbf{H}^\top\mathbf{y}. \tag{38}$$

Finally, from rows 3 and 4, $\mathbf{q}_1^t$ and $\mathbf{q}_2^t$ can be computed as

$$\mathbf{q}_1^t=\frac{1}{2}\left(\mathbf{z}^{t+1}-\mathbf{C}\mathbf{x}^{t+1}\right)+\frac{1}{2\gamma}(\boldsymbol{\mu}_a^t-\boldsymbol{\mu}_d^t+\gamma\tilde{\mathbf{q}}_1^t)$$

$$\mathbf{q}_2^t=\frac{1}{2}\left(\mathbf{z}^{t+1}+\mathbf{C}\mathbf{x}^{t+1}\right)+\frac{1}{2\gamma}(\boldsymbol{\mu}_b^t-\boldsymbol{\mu}_e^t+\gamma\tilde{\mathbf{q}}_2^t). \tag{39}$$

A.4 Invertibility of $\boldsymbol{\mathcal{L}}=\mathbf{C}^\top\mathbf{C}+\mathbf{H}^\top\mathbf{H}$

Clearly $\boldsymbol{\mathcal{L}}=\mathbf{C}^\top\mathbf{C}+\mathbf{H}^\top\mathbf{H}$ is real, symmetric, and positive semi-definite. Thus its eigenvalues are real and non-negative. To show that it is invertible, it suffices to show that its minimum eigenvalue $\lambda_{\min}(\boldsymbol{\mathcal{L}})$ is strictly greater than zero. But $\lambda_{\min}(\boldsymbol{\mathcal{L}})=\min_{\mathbf{x}:||\mathbf{x}||=1}\mathbf{x}^\top\boldsymbol{\mathcal{L}}\mathbf{x}$. Hence it suffices to show that $\mathbf{x}^\top\boldsymbol{\mathcal{L}}\mathbf{x}=0$ implies $\mathbf{x}=\mathbf{0}$. Now observe that $\mathbf{x}^\top\boldsymbol{\mathcal{L}}\mathbf{x}=\sum_{(i,j)\in\mathcal{E}}w_{i,j}^2(x_i-x_j)^2+\sum_{i\in\mathcal{S}}x_i^2$, where $\mathcal{S}$ is the set of nodes with constrained values. If $\mathbf{x}^\top\boldsymbol{\mathcal{L}}\mathbf{x}=0$ then all terms must be zero, meaning that all x_i in $\mathcal{S}$ are zero, and hence all of their neighbors, and all of *their* neighbors, and so on. Thus, $\boldsymbol{\mathcal{L}}$ is invertible if there exists at least one node with a constrained value (i.e., a self-loop) in each connected component of the graph.

A.5 Derivation of (21)

We derive the solution to optimization (20). Ignoring the first convex but non-smooth term $g(\tilde{\mathbf{q}})$, the remaining two terms in the objective are convex and smooth. Taking the derivative w.r.t. variable $\tilde{\mathbf{q}}$ and setting it to $\mathbf{0}$, we get

$$-\boldsymbol{\mu}_d^t-\gamma\mathbf{q}^{t+1}+\gamma\tilde{\mathbf{q}}^*=\mathbf{0}_M$$

$$\tilde{\mathbf{q}}^*=\mathbf{q}^{t+1}+\frac{1}{\gamma}\boldsymbol{\mu}_d^t. \tag{40}$$

This solution is valid iff $g(\tilde{\mathbf{q}}^*)=0$; otherwise the first term $g(\tilde{\mathbf{q}}^*)$ dominates and $\tilde{\mathbf{q}}^*=\mathbf{0}_M$. Given that (40) can be computed entry-by-entry separately, (21) follows.

A.6 Example of Edge Weight Normalization for the Incidence Matrix

In Fig. 3(top), we show an example three-node *undirected* graph with three edge weights $w_{1,2}=1/2$, $w_{1,3}=1/2$, and $w_{2,3}=1/3$, and the corresponding incidence matrix $\mathbf{C}$. Normalizing edge weights using (26), we see in Fig. 3(middle) a *directed* graph with six edges where the sum of normalized edge weights leaving a node is 1, resulting in normalized incidence matrix $\bar{\mathbf{C}}$. Finally, we see in Fig. 3(bottom) an *undirected* graph with three edges corresponding to graph Laplacian $\bar{\mathbf{L}}=\bar{\mathbf{C}}^\top\bar{\mathbf{C}}$. Note that $\|\bar{\mathbf{C}}\mathbf{1}_3\|_1=\mathbf{0}_6$ as expected.

A.7 Parameters Learning in Conjugate Gradient Algorithm (CG)

The linear systems that we need to solve—(8) for GLR minimization and (18) for GTV minimization—have the follow form,

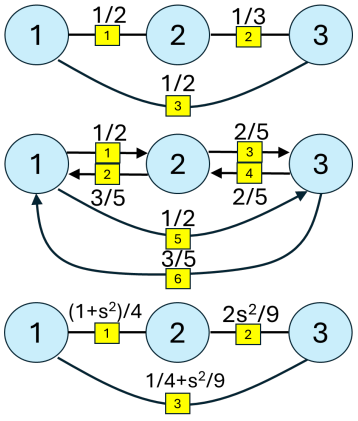
$$\boldsymbol{\mathcal{L}}\mathbf{x}=\mathbf{b}. \tag{41}$$

Given $\boldsymbol{\mathcal{L}}$ is PD, we consider the minimization problem,

$$\min_{\mathbf{x}}Q(\mathbf{x})=\frac{1}{2}\mathbf{x}^\top\boldsymbol{\mathcal{L}}\mathbf{x}-\mathbf{b}^\top\mathbf{x} \tag{42}$$

with gradient

$$\frac{\delta Q(\mathbf{x})}{\delta\mathbf{x}}=\boldsymbol{\mathcal{L}}\mathbf{x}-\mathbf{b}. \tag{43}$$



$$\mathbf{C} = \begin{bmatrix} 1/2 & -1/2 & 0 \\ 0 & 1/3 & -1/3 \\ 1/2 & 0 & -1/3 \end{bmatrix}$$

$$\bar{\mathbf{C}} = \begin{bmatrix} 1/2 & -1/2 & 0 \\ -3/5 & 3/5 & 0 \\ 0 & 2/5 & -2/5 \\ 0 & -2/5 & 2/5 \\ 1/2 & 0 & -1/2 \\ -3/5 & 0 & 3/5 \end{bmatrix}$$

$$\bar{\mathbf{L}} = \begin{bmatrix} \frac{1}{2} + \frac{s^2}{4} + \frac{s^2}{9} & -\frac{1+s^2}{4} & -\frac{1}{4} - \frac{s^2}{9} \\ -\frac{1+s^2}{4} & \frac{1+s^2}{4} + \frac{2s^2}{9} & -\frac{2s^2}{9} \\ -\frac{1}{4} - \frac{s^2}{9} & -\frac{2s^2}{9} & \frac{1}{4} + \frac{s^2}{3} \end{bmatrix}$$

Figure 3: 3-node graph for incidence matrix $\mathbf{C}$ (top), 3-node graph for normalized incidence matrix $\bar{\mathbf{C}}$ (middle), 3-node graph for graph Laplacian $\bar{\mathbf{L}} = \bar{\mathbf{C}}^\top \bar{\mathbf{C}}$ where $s = \frac{6}{5}$ (bottom).

Thus, the simple gradient descent has the following update rules with α_t commonly known as learning rate,

$$\mathbf{g}^t = \boldsymbol{\mathcal{L}}\mathbf{x}^t - \mathbf{b} = \mathbf{g}^{t-1} - \alpha_t \boldsymbol{\mathcal{L}}\mathbf{g}^{t-1} \tag{44}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha_t \mathbf{g}^t. \tag{45}$$

Next, a momentum term β_t and the cumulative gradients term $\mathbf{v}^t$ are added, resulting in the well-known Accelerated Gradient Descent Algorithm [52], also known as Conjugated Gradient Descent. The new update rules are

$$\mathbf{g}^t = \mathbf{g}^{t-1} - \alpha_t \boldsymbol{\mathcal{L}}\mathbf{v}^{t-1} \tag{46}$$

$$\mathbf{v}^t = \mathbf{g}^t + \beta_t \mathbf{v}^{t-1} \tag{47}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \alpha_t \mathbf{v}^t \tag{48}$$

where both α_t and β_t in each iteration t are considered trainable parameters.

A.8 Experimental Details

Initially, we developed our *unrolled* ADMM model without training any parameters. The target signal $\mathbf{x}$ was estimated using linear interpolation of known values in 5×5 pixel neighborhood. RGB values and pixel locations were combined to form feature vectors for computing edge weights $w_{i,j}$, which were shared across the three channels. The metric matrix $\mathbf{M}$ was initialized as a diagonal matrix with all entries set to 1.5. Vectors μ_a, μ_b, μ_c, μ_d, and μ_e were initialized with all entries equal to 0.1. The parameters γ, α, and β in CG were set to 10, 0.5, and 0.3, respectively. For the *learned* ADMM block, the parameters γ, α, β, and the metric matrix $\mathbf{M}$ were learned during training. A training dataset was created consisting of 5000, 10000, or 20000 image patches, each of size 64×64, to train the model.

All matrix multiplications in our models are implemented to take advantage of the sparsity of the constructed graphs, which were restricted to be a window of size $5 \times 5 = 25$ nearest neighbors of each pixel. This ensures that our graphs are always *connected* and *sparse*. We stacked vertically 4 Graph Learning modules coupled with ADMM block, so that we can learn multiple graphs in parallel. This is commonly known as multi-head in the transformer architecture. We also stacked 5 graph learning modules and ADMM blocks horizontally to further learn more graphs. In all ADMM blocks, we set the number of ADMM iterations to 5 and the number of CG iterations to 10.

To extract high-level features, a shallow CNN was employed, consisting of four convolutional layers with 48 feature maps (12 features for each graph learning head). After each convolutional layer, a ReLU activation function was applied. Convolutional layers utilized 3×3 kernels without any down-sampling, generating 48 features for each pixel. The feature maps were divided into four sets,

(a) Urban100: image062.png (b) Original (c) iGTV (d) uGTV (e) uGLR (f) RST-B (g) RST-S

Figure 4: Visual demosaicking results for image *Urban062*.

with each set serving as the input for a Graph Learning module, allowing for the parallel production of four graphs. A simple weighted average scheme was then used to combine the outputs into a single output $\mathbf{x}^t$. Additionally, skip connections were introduced between the convolutional layers to facilitate the training process.

A.9 Additional Experimental Results

Fig. 4 shows visual results for a test image for all models, including our uGTV, uGLR and the baselines. Two variants of RSTCANet, uGTV and uGLR are trained on 10000 images patches of size 64×64 for 30 epochs. We observe that our two models, especially uGTV, has better performance compared to RST-B and comparable performance with RST-S in selected high-frequency area.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Main claims are clearly stated in the abstract and Introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We state the limitation of our edge weight computation, as compared to self-attention weights in conventional transformers, in Section 5.2.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: All proofs and derivations are included in the Appendices.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Details of our experiments are explained in Section 6.1 and Appendix A.8.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We used open source data: McM [43], Kodak [44], and Urban100 [45] datasets.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We described the experiments setup in Section 6.1 and Appendix A.8.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We did provide various metrics for many datasets to compare our models with the baseline models.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We described the experimental setup in Section 6.1 and Appendix A.8.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: As authors, we have made every effort to conform to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The performed work is purely computational and does not have direct societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.