TUNING-FREE GRAPH-BASED SEMI-SUPERVISED LEARNING IN THE EXTREME LOW-LABEL REGIME

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph-based semi-supervised learning spreads a small set of labels across a graph whose edges encode similarity between data points, and it is the natural tool wherever annotation is costly but relations between samples are known. The underlying task is combinatorial, since each vertex must receive exactly one label and each class must retain its share of the vertices. Established methods relax those requirements, in the sense of losing tightness, solving a continuous surrogate and reading a labeling off it afterwards; we keep the problem discrete throughout. An auxiliary variable linearizes the quadratic objective, so that every iteration becomes a transportation problem with a totally unimodular constraint matrix, whose solution is integral without any rounding and meets the prescribed class sizes exactly. We prove that the iteration ascends monotonically, that the objective sequence converges, and that its limit points are stationary. Because the problem is non-convex, the initialization also matters, and we therefore unroll the iteration into a network whose ingredients are learned and which keeps the prescribed class sizes exact at inference. Our study is directed at the extreme low-label regime, where only a few labels per class are observed and where graph-based methods are most needed. On citation benchmarks in that regime the proposed method outperforms the state of the art in accuracy and labels each graph in a fraction of a second.

1 INTRODUCTION

In many real-world areas, from image recognition [17, 26] to the analysis of citation and document networks [28], the high cost of labeling data is a major challenge, and this has encouraged the development of semi-supervised learning (SSL). The main idea of SSL is to use the structure found in a large amount of unlabeled data to help a small set of labeled data, so as to learn a better predictor than the labeled data alone would allow. Graph-based methods are a leading approach in this area; they represent data points as the vertices of a graph and impose a smoothness rule that pushes neighbouring vertices to carry similar labels [1, 27, 29–31]. Early techniques of this kind, such as Laplace learning [31] and Poisson learning [4], compute a smooth extension of the observed labels across the graph.

These foundational methods nevertheless encounter significant challenges when labeled data is extremely scarce [8, 19]. The harmonic extension concentrates at the labeled vertices, creating sharp spikes while leaving distant ones largely unaffected, and the thresholding step that converts the continuous solution into discrete labels can amplify the errors that follow. To mitigate the issues arising from imbalanced class distributions, practitioners add volume-constraint mechanisms to the propagation, one of them alternating diffusion with an assignment computed under volume and box constraints [4, 15]. These corrections underscore the core issue, in that assigning binary one-hot labels under exact class-size constraints is an inherently combinatorial optimization task, and a requirement imposed on a continuous rule after the fact is not the same as one the formulation carries throughout.

Recognizing this, the task has been posed directly as a cardinality-constrained minimum-cut partitioning problem, yet tractability has continued to be bought with continuous relaxations, in the sense of losing tightness. Procrustes-SSL [12] relaxes the non-convex binary constraints by embedding the labels onto the Stiefel manifold, aligning a Laplacian eigenmap to the labeled vertices by an orthogonal transform, so that one-hot assignments give way to orthogonality. A relaxation of this kind need not be tight across all problem dimensions, so that the solution it returns may fail to correspond

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

to the optimum of the original discrete problem however accurately the relaxed problem is solved, and the tractability it buys is paid for in exactly that loss [3]. CutSSL [13] keeps the cut objective and perturbs it instead, shifting the Laplacian by a multiple of the degree matrix so that the minimizers of the relaxed program migrate towards the extreme points of the feasible set and, under a condition relating that multiple to the degrees and the edge weights, land on integral labelings. Its exactness therefore rests on the size of that shift, the primary hyperparameter of the method, which is raised along a sequence of perturbed programs, and the value adopted in practice falls short of the condition under which integrality is guaranteed, so that the labeling it returns lies outside the reach of that guarantee. CombCut [21] stays closer to the problem as posed, leaving the objective unperturbed and descending on it with projections onto the transportation polytope. The perturbation is nevertheless present in what it returns, since the descent is started from the solution of the perturbed program above, so that the parameter governing it, and the landscape it induces, reach the final labeling through the initialization even though they appear nowhere in the objective being minimized.

A different response leaves the propagation continuous and changes instead what is propagated or how it is regularized. The p-Laplace method [9] replaces the Laplacian by the non-linear p-Laplacian, which resists the spikes that the harmonic extension develops at low label rates; Correct & Smooth [14] spreads the residuals of a simple base classifier over the graph and then smooths the corrected scores; and GEM [18] trains a one-hop model by minimizing the entropy of its predictions over edges, drawing supervision from the uncategorized vertices themselves. None of the three ever constrains the class proportions, and each imposes the one-hot structure only at the very end, by reading a label off the largest continuous score at each vertex, so that the difficulty the combinatorial statement was meant to remove returns in a different guise.

To overcome the limitations common to these approaches, we introduce **DASH SSL**, a framework whose name records the principles it is built on, being **D**iscrete, **A**lternating, **S**imple and **H**yperparameter-free (no user-chosen parameter biases the solution, a convergence tolerance not being one [25]). Our main innovation is a reformulation of the challenging quadratic integer program, through a shift that makes its objective convex and an auxiliary variable that turns it into a joint maximization. Every label update then maximizes a lower bound on the objective that is exact at the current labeling, and it is a structured linear assignment problem whose constraint matrix we show to be totally unimodular. Solutions obtained via efficient network-flow algorithms are therefore inherently integral, no heuristic rounding is needed, and the tightness of the original formulation is preserved throughout the solver.

Main Contribution

The main contributions of this work are given as summary:

(i) **A discrete formulation solved without relaxation.** We pose multi-class graph semi-supervised learning directly over one-hot label assignments and keep it there, using an auxiliary variable that linearizes the quadratic objective so that every iteration reduces to a transportation problem whose solution needs no rounding.

(ii) **Integral iterates through total unimodularity.** We show that the constraint matrix of this subproblem is totally unimodular, so that its relaxation attains only integral extreme points and the prescribed class sizes are met without any rounding and without perturbing the problem.

(iii) **Monotonic ascent and stationarity.** We prove that the iteration never decreases the discrete objective, that the objective sequence is bounded and therefore convergent, and that any limit point of the iterates satisfies the first-order optimality conditions of the original problem.

(iv) **The role of initialization.** Because the problem is non-convex, the initialization also matters, and we treat the point from which the iteration starts as part of the method, a design choice that the extreme low-label regime makes it necessary to address explicitly.

(v) **From a fixed iteration to a learned one.** We unroll the iteration into a network whose layers keep its structure while the previously hand-fixed quantities become parameters fit to the graph, and an integral assignment step is retained at inference so that the class-size guarantee survives.

(vi) **An evaluation in the extreme low-label regime.** We evaluate our framework on the citation benchmarks at the sparsest label rates, against the state-of-the-art and benchmark methods of graph-based semi-supervised learning.

Notation. Throughout this paper we use boldface lowercase letters for vectors and boldface uppercase letters for matrices. The transpose operator is denoted by $(\cdot)^T$, the trace of a square matrix by $\mathrm{tr}(\cdot)$, and the identity matrix by $\mathbf{I}$. The all-ones vector is written $\mathbf{1}$, with its dimension inferred from the context, and $\mathbf{e}_j$ denotes the j-th standard basis vector. For a vector $\mathbf{a}$, the notation $\mathrm{diag}(\mathbf{a})$ refers to the diagonal matrix carrying the entries of $\mathbf{a}$ on its diagonal. We write $\lambda_{\max}(\mathbf{A})$ for the largest eigenvalue of a symmetric matrix $\mathbf{A}$ and $\lambda_2(\mathbf{A})$ for its smallest nonzero eigenvalue, and $\mathbf{A} \succeq \mathbf{0}$ indicates that $\mathbf{A}$ is positive semidefinite. The squared Frobenius norm is written $\|\cdot\|_F^2$, the Kronecker product is denoted by $\otimes$, and $\mathrm{vec}(\cdot)$ stacks the columns of a matrix into a single vector. The graph is denoted by G, its vertex set by $\mathcal{V}$, and the labeled and unlabeled vertex sets by ℓ and $\mathcal{U}$ respectively; k is reserved for the number of classes, k' for the number of neighbours used in a nearest-neighbour graph, m for the number of labeled vertices and n for the number of unlabeled ones, and $\mathcal{C}$ denotes the transportation polytope of label matrices whose rows each sum to one and whose columns sum to the prescribed class sizes. Iterates of an algorithm carry a parenthesized superscript, so that $\mathbf{X}^{(t)}$ is the label matrix at iteration t.

2 Problem Formulation

We define the problem on an undirected weighted graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with vertex set $\mathcal{V} = \{v_1, \ldots, v_M\}$ and a symmetric weight matrix $\mathbf{W} \in \mathbb{R}^{M\times M}$ whose non-negative entries w_{ij} represent the similarity between v_i and v_j. The degree of a vertex is $d_i = \sum_{j=1}^{M} w_{ij}$, and these values are collected into a diagonal degree matrix $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_M)$. Without loss of generality, the vertex set is partitioned so that the initial m vertices, forming $\ell = \{v_1, \ldots, v_m\}$, carry known one-hot labels $\{\mathbf{y}_1, \ldots, \mathbf{y}_m\} \subset \{\mathbf{e}_1, \ldots, \mathbf{e}_k\}$, a subset of the standard basis vectors in $\mathbb{R}^k$. In the standard semi-supervised setting $0 < m \ll M$, and the remaining $n = M - m$ vertices comprise the unlabeled set $\mathcal{U} = \{v_{m+1}, \ldots, v_M\}$. The objective is to propagate the known labels from ℓ across $\mathcal{U}$ consistently with the structure of the graph.

Based on Laplace learning [12, 13, 21, 31], a soft-label matrix $\mathbf{X}_0 \in \mathbb{R}^{M\times k}$ is determined by solving

$$\min_{\mathbf{X}_0 \in \mathbb{R}^{M\times k}} \ \mathrm{tr}\big(\mathbf{X}_0^T \mathbf{L}\, \mathbf{X}_0\big) \quad \text{s.t. } (\mathbf{X}_0)_i = \mathbf{y}_i,\ i = 1, \ldots, m, \tag{1}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the combinatorial Laplacian and the final label of v_i is recovered as $\hat{\mathbf{y}}_i = \arg\max_{1\le j\le k} (\mathbf{X}_0)_{ij}$. To focus on the unlabeled vertices, we partition $\mathbf{L}$ and $\mathbf{X}_0$ into blocks corresponding to the labeled and unlabeled sets,

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{\ell\ell} & \mathbf{L}_{\ell u} \\ \mathbf{L}_{u\ell} & \mathbf{L}_{uu} \end{pmatrix}, \qquad \mathbf{X}_0 = \begin{pmatrix} \mathbf{Y} \\ \mathbf{X} \end{pmatrix},$$

in which $\mathbf{Y} \in \{0,1\}^{m\times k}$ holds the known one-hot labels and $\mathbf{X} \in \mathbb{R}^{n\times k}$ the unknown ones. Substituting these blocks into (1) and removing constant terms gives a reduced objective depending only on $\mathbf{X}$,

$$\min_{\mathbf{X} \in \{0,1\}^{n\times k}} \ \tfrac{1}{2}\,\mathrm{tr}\big(\mathbf{X}^T \mathbf{L}_{uu}\, \mathbf{X}\big) \ - \ \mathrm{tr}\big(\mathbf{X}^T \mathbf{B}\big), \tag{2}$$

in which the matrix $\mathbf{B} \in \mathbb{R}^{n\times k}$ is the *supervision term*, the only part of the objective carrying information about the observed labels.

That substitution produces the choice $\mathbf{B}_{\mathrm{lap}} = -\,\mathbf{L}_{u\ell}\,\mathbf{Y}$, which is *one-hop*, since $\mathbf{L}_{u\ell}$ is nonzero only in rows corresponding to unlabeled vertices adjacent to a labeled one, so that $\mathbf{B}_{\mathrm{lap}}$ is supported on the immediate neighborhood of ℓ. When labeled vertices are abundant this is immaterial, but in the extreme low-label regime it is decisive. With one label per class, $\mathbf{B}_{\mathrm{lap}}$ is supported on the neighbors of k vertices out of thousands, the objective in (2) is numerically dominated by the label-agnostic cut term $\frac{1}{2}\,\mathrm{tr}(\mathbf{X}^T\mathbf{L}_{uu}\mathbf{X})$, and the optimizer is then, to good approximation, solving an *unsupervised* balanced partitioning problem.

We therefore take $\mathbf{B}$ to be a *global* supervision field, obtained from the Poisson formulation of graph-based SSL [4]. Let $\mathcal{V}_1, \ldots, \mathcal{V}_c$ be the connected components of G, with indicator vectors

$\mathbf{1}_{\mathcal{V}_r} \in \{0,1\}^M$, let $\bar{\mathbf{y}}_r$ denote the mean of the labels observed in $\mathcal{V}_r$, and let $\mathbf{U} \in \mathbb{R}^{M\times k}$ solve the graph Poisson equation over the entire vertex set,

$$\mathbf{L}\,\mathbf{U} \;=\; \mathbf{F}, \qquad \mathbf{F}_i = \begin{cases} \mathbf{y}_i - \bar{\mathbf{y}}_r, & v_i \in \ell \cap \mathcal{V}_r, \\ \mathbf{0}, & v_i \in \mathcal{U}, \end{cases} \qquad \mathbf{1}_{\mathcal{V}_r}^T \mathbf{U} = \mathbf{0}^T, \quad r = 1, \ldots, c, \tag{3}$$

and set $\mathbf{B} = \mathbf{B}_{\mathrm{poi}} \triangleq \mathbf{U}_{\mathcal{U}}$, the restriction of $\mathbf{U}$ to the unlabeled block, scaled to the Frobenius norm of $\mathbf{B}_{\mathrm{lap}}$. Centering the sources within each component is what makes (3) solvable, since the null space of $\mathbf{L}$ is spanned by the indicators $\mathbf{1}_{\mathcal{V}_r}$, and the zero-mean conditions fix the remaining constant modes; on a connected graph this is the global centering of Poisson learning, and a component that carries no label receives no source. Replacing label *values* by mean-zero *sources* is the correction proposed for the degeneracy of Laplace learning at low label rates, and it yields a term supported on the whole graph, not on a one-hop shell. The field $\mathbf{B}_{\mathrm{poi}}$ is computed *once* per problem, at the cost of k sparse linear solves, and it enters (2) *linearly in* $\mathbf{X}$, so that every result in Subsections 2.1 and 3.1 depends on it only through the linearity of $\mathrm{tr}(\mathbf{X}^T\mathbf{B})$ and the choice of supervision term is *orthogonal* to the solver.

We now impose constraints guaranteeing that the assignments for $\mathcal{U}$ are integral one-hot vectors, namely $\mathbf{X} \in \{0,1\}^{n\times k}$ together with $\mathbf{X}\,\mathbf{1}_k = \mathbf{1}_n$, $\mathbf{X}^T\,\mathbf{1}_n = \mathbf{m}$ and $\mathbf{1}_k^T\mathbf{m} = n$, where $\mathbf{m} \in \mathbb{R}^k$ specifies the desired number of examples in each class. Each unlabeled vertex is thereby assigned to exactly one class and the resulting class sizes are met, so that our discrete formulation of the semi-supervised learning problem is given by

$$\begin{aligned} \min_{\mathbf{X}\in\{0,1\}^{n\times k}} \quad & F(\mathbf{X}) = \tfrac{1}{2}\,\mathrm{tr}\big(\mathbf{X}^T\,\mathbf{L}_{uu}\,\mathbf{X}\big) \;-\; \mathrm{tr}\big(\mathbf{X}^T\,\mathbf{B}\big) \\ \text{s.t.} \quad & \mathbf{X}\,\mathbf{1}_k = \mathbf{1}_n, \qquad \mathbf{X}^T\,\mathbf{1}_n = \mathbf{m}. \end{aligned} \tag{4}$$

In the following subsection we present a method that solves (4) directly, without resorting to a continuous relaxation, in the sense of losing tightness, that can lead to suboptimal solutions.

2.1 The DASH SSL Optimization Framework

The problem stated in (4) requires the minimization of a non-convex quadratic function over a discrete domain, a task that is generally NP-hard. We first transform it into an equivalent maximization problem, a reformulation facilitated by the following property of the one-hot assignment matrix.

Lemma 1. *Let $\mathbf{X} \in \{0,1\}^{n\times k}$ satisfy $\mathbf{X}\,\mathbf{1}_k = \mathbf{1}_n$. Then $\|\mathbf{X}\|_F^2 = n$ (for the proof, see Appendix A).*

We set the scalar s to $1.01\,\lambda_{\max}(\mathbf{L}_{uu})$, the one percent serving only as a numerical margin, so that $\mathbf{M} = s\,\mathbf{I}_n - \mathbf{L}_{uu}$ is positive semi-definite and s is computed from the graph, not chosen. Using this construction and Lemma 1, the objective from (4) can be rewritten for any feasible $\mathbf{X}$ as

$$\begin{aligned} \tfrac{1}{2}\mathrm{tr}(\mathbf{X}^T\mathbf{L}_{uu}\,\mathbf{X}) - \mathrm{tr}(\mathbf{X}^T\mathbf{B}) &= -\big(\tfrac{1}{2}\mathrm{tr}(\mathbf{X}^T\mathbf{M}\,\mathbf{X}) + \mathrm{tr}(\mathbf{X}^T\mathbf{B})\big) \;+\; \tfrac{1}{2}\,s\,\mathrm{tr}(\mathbf{X}^T\mathbf{X}) \\ &= -\big(\tfrac{1}{2}\mathrm{tr}(\mathbf{X}^T\mathbf{M}\,\mathbf{X}) + \mathrm{tr}(\mathbf{X}^T\mathbf{B})\big) \;+\; \tfrac{1}{2}\,s\,n, \end{aligned} \tag{5}$$

in which the term $\frac{1}{2}sn$ is constant with respect to $\mathbf{X}$ and can therefore be omitted. The original minimization problem is consequently equivalent to the maximization of a convex objective,

$$\begin{aligned} \max_{\mathbf{X}\in\{0,1\}^{n\times k}} \quad & f(\mathbf{X}) = \tfrac{1}{2}\,\mathrm{tr}(\mathbf{X}^T\mathbf{M}\,\mathbf{X}) \;+\; \mathrm{tr}(\mathbf{X}^T\mathbf{B}) \\ \text{s.t.} \quad & \mathbf{X}\,\mathbf{1}_k = \mathbf{1}_n, \qquad \mathbf{X}^T\mathbf{1}_n = \mathbf{m}. \end{aligned} \tag{6}$$

Although the objective in (6) is convex, the binary constraints on $\mathbf{X}$ keep the problem combinatorial. The following lemma introduces the identity that expresses it in a form affine with respect to $\mathbf{X}$ without introducing any relaxation.

Lemma 2. *For any given matrix $\mathbf{X}$ and a symmetric positive semi-definite matrix $\mathbf{M}$, we have*

$$\mathrm{tr}(\mathbf{X}^T\mathbf{M}\mathbf{X}) = \max_{\mathbf{Q}} \Big(2\cdot\mathrm{tr}(\mathbf{Q}^T\mathbf{M}^{1/2}\mathbf{X}) - \mathrm{tr}(\mathbf{Q}^T\mathbf{Q})\Big),$$

and the maximum with respect to $\mathbf{Q}$ is achieved at the unique point $\mathbf{Q}^* = \mathbf{M}^{1/2}\mathbf{X}$ (for the proof, see Appendix A).

Algorithm 1 The DASH SSL algorithm; $\mathbf{B} \leftarrow -\mathbf{L}_{u\ell}\mathbf{Y}$ gives the one-hop variant.

Require: $\mathbf{L}$, $\mathbf{L}_{uu}$, labels $\mathbf{Y}$, sizes $\mathbf{m}$, tolerance ϵ
Ensure: One-hot labeling $\mathbf{X} \in \{0,1\}^{n\times k}$
1: Solve $\mathbf{L}\mathbf{U} = \mathbf{F}$ as in (3); set $\mathbf{B} \leftarrow \mathbf{U}_{\mathcal{U}}$, scaled to $\|\mathbf{L}_{u\ell}\mathbf{Y}\|_F$
2: $s \leftarrow 1.01\,\lambda_{\max}(\mathbf{L}_{uu})$; $\mathbf{M} \leftarrow s\,\mathbf{I}_n - \mathbf{L}_{uu}$
3: Initialize $\mathbf{X}^{(0)}$ feasible, with $\mathbf{X}^{(0)}\mathbf{1}_k = \mathbf{1}_n$ and $(\mathbf{X}^{(0)})^T\mathbf{1}_n = \mathbf{m}$
4: **for** $t = 0, 1, 2, \ldots$ **do**
5: $\quad\mathbf{Q}^{(t+1)} \leftarrow \mathbf{M}^{1/2}\mathbf{X}^{(t)}$
6: $\quad\mathbf{X}^{(t+1)} \leftarrow \arg\max_{\mathbf{X}}\ \operatorname{tr}\big(\mathbf{X}^T(\mathbf{M}^{1/2}\mathbf{Q}^{(t+1)} + \mathbf{B})\big)\quad$ s.t. $\mathbf{X} \in [0,1]^{n\times k}$, $\mathbf{X}\mathbf{1}_k = \mathbf{1}_n$, $\mathbf{X}^T\mathbf{1}_n = \mathbf{m}$
7: $\quad$**if** $\big|F(\mathbf{X}^{(t+1)}) - F(\mathbf{X}^{(t)})\big| \big/ \big|F(\mathbf{X}^{(t)})\big| < \epsilon$ **then break**
8: **end for**
9: **return** $\mathbf{X}^{(t+1)}$

Leveraging Lemma 2 to replace the quadratic term in (6) transforms the maximization over $\mathbf{X}$ into the joint maximization

$$\begin{aligned} \max_{\mathbf{X},\mathbf{Q}} \quad & \operatorname{tr}(\mathbf{Q}^T\mathbf{M}^{1/2}\mathbf{X}) - \tfrac{1}{2}\operatorname{tr}(\mathbf{Q}^T\mathbf{Q}) + \operatorname{tr}(\mathbf{X}^T\mathbf{B}) \\ \text{s.t.} \quad & \mathbf{X} \in \{0,1\}^{n\times k}, \quad \mathbf{X}\,\mathbf{1}_k = \mathbf{1}_n, \quad \mathbf{X}^T\mathbf{1}_n = \mathbf{m}, \end{aligned} \tag{7}$$

involving both $\mathbf{X}$ and an auxiliary matrix variable $\mathbf{Q}$. The crucial benefit is that the objective in (7) is linear in $\mathbf{X}$ when $\mathbf{Q}$ is held constant, a structure ideally suited to an alternating algorithm. Fixing $\mathbf{X}^{(t)}$, the objective is an unconstrained quadratic in $\mathbf{Q}$ whose maximizer follows in closed form from Lemma 2 as $\mathbf{Q}^{(t+1)} = \mathbf{M}^{1/2}\mathbf{X}^{(t)}$; fixing $\mathbf{Q}^{(t+1)}$ in turn, the optimization with respect to $\mathbf{X}$ becomes the integer linear program

$$\begin{aligned} \mathbf{X}^{(t+1)} = \arg\max_{\mathbf{X}} \quad & \operatorname{tr}\Big(\mathbf{X}^T(\mathbf{M}^{1/2}\mathbf{Q}^{(t+1)} + \mathbf{B})\Big) \\ \text{s.t.} \quad & \mathbf{X} \in \{0,1\}^{n\times k}, \quad \mathbf{X}\,\mathbf{1}_k = \mathbf{1}_n, \quad \mathbf{X}^T\mathbf{1}_n = \mathbf{m}. \end{aligned} \tag{8}$$

A common strategy for such a problem is to relax the integrality constraint $X_{ij} \in \{0,1\}$ to the box $0 \le X_{ij} \le 1$. Although a relaxation of that kind can in general lead to fractional solutions, and so to a loss of tightness, the structure of the constraints in (8) keeps the solution integral, which is a direct result of the total unimodularity of the constraint matrix, described in the definition and theorem below [20, 24].

Definition 3. A matrix $\mathbf{D}$ is *totally unimodular* (TU) if every square submatrix has a determinant equal to $-1, 0$, or 1.

Theorem 4. *If* $\mathbf{D}$ *is TU and* $\mathbf{b}$ *is an integral vector, then the polyhedron* $\{\mathbf{x} : \mathbf{D}\,\mathbf{x} \le \mathbf{b},\ 0 \le \mathbf{x} \le 1\}$ *contains only integral vertices, and consequently any linear program over this system has an integral optimal solution (for the proof, see Appendix B).*

In Appendix B we vectorize (8), write its row-sum and column-sum constraints as a single system $\mathbf{D}\mathbf{x} \le \mathbf{b}$ with integral $\mathbf{b}$, and verify that $\mathbf{D}$ is TU, so that Theorem 4 applies and every vertex of the feasible polyhedron is integral. No tightness is lost, and the integer program for $\mathbf{X}$ may therefore be solved in the continuous form

$$\begin{aligned} \max_{\mathbf{X}\in[0,1]^{n\times k}} \quad & \operatorname{tr}\big((\mathbf{M}^{1/2}\mathbf{Q}^{(t+1)} + \mathbf{B})^T\mathbf{X}\big) \\ \text{s.t.} \quad & \mathbf{X}\,\mathbf{1}_k = \mathbf{1}_n, \qquad \mathbf{X}^T\mathbf{1}_n = \mathbf{m}, \end{aligned} \tag{9}$$

which is handled efficiently by standard solvers such as CVX [11] or CVXPY [7], or via Lagrangian duality methods [23]; an optimal vertex, which simplex-type and network-flow methods return directly, is integral. The complete procedure is summarized in Algorithm 1.

3 Computational Complexity and Convergence

The per-iteration cost of DASH SSL is $O(k^2 n + \mathrm{Cost}_{\mathrm{LP}}(n,k))$, where the first term is the sparse product $\mathbf{M}\mathbf{X}^{(t)}$ that forms the cost matrix and the second is the transportation program of (9), and

the memory footprint is linear in the number of vertices; Appendix C derives both. The number of iterations to stationarity, T, reflects how far supervision has to travel, since the cost operator $\mathbf{M} = s\mathbf{I} - \mathbf{L}_{uu}$ has the sparsity pattern of the graph. An unrolled schedule has a depth chosen in advance and independent of n, so that its cost is linear in the number of vertices, which is one of the concrete motivations for Section 4.

3.1 Convergence Analysis

We now establish that the iterates of DASH SSL produce a non-decreasing objective value, that this sequence of values is bounded and therefore convergent, and that any limit point is a KKT stationary point of (6). We first state the first-order necessary condition for a local maximizer of a smooth function over a closed convex set [2].

Proposition 5. *Let $f : \mathbb{R}^{n\times k} \to \mathbb{R}$ be continuously differentiable, and let $\mathcal{C} \subseteq \mathbb{R}^{n\times k}$ be a closed, convex set. If $\mathbf{X}^*$ is a local maximizer of f over $\mathcal{C}$, then $\operatorname{tr}\big(\nabla_{\mathbf{X}} f(\mathbf{X}^*)^T\,(\mathbf{Z} - \mathbf{X}^*)\big) \le 0$ for all $\mathbf{Z} \in \mathcal{C}$ (for the proof, see Appendix A).*

DASH SSL is an alternating procedure applied to the equivalent formulation of $F(\mathbf{X})$ derived from Lemma 2, namely the maximization of

$$G(\mathbf{X}, \mathbf{Q}) = \operatorname{tr}(\mathbf{Q}^T \mathbf{M}^{1/2} \mathbf{X}) - \tfrac{1}{2}\operatorname{tr}(\mathbf{Q}^T\mathbf{Q}) + \operatorname{tr}(\mathbf{X}^T\mathbf{B}) \tag{10}$$

over $\mathbf{X} \in \mathcal{C}$ and $\mathbf{Q} \in \mathbb{R}^{n\times k}$. The following theorem collects the three guarantees that this structure provides.

Theorem 6. *Let $\{\mathbf{X}^{(t)}\}$ be the sequence generated by DASH SSL from any feasible $\mathbf{X}^{(0)}$, and let f be the objective of* (6)*. Then*

- *(a) the objective is non-decreasing, that is, $f(\mathbf{X}^{(t+1)}) \ge f(\mathbf{X}^{(t)})$ for every t;*
- *(b) the sequence $\{f(\mathbf{X}^{(t)})\}$ is bounded above and therefore converges, and when ties in the $\mathbf{X}$-update are broken in favor of the current iterate, the iterates, being integral points of a finite feasible set, reach a fixed point after finitely many iterations;*
- *(c) every fixed point $\mathbf{X}^{(\infty)}$ satisfies $\operatorname{tr}(\nabla_{\mathbf{X}} f(\mathbf{X}^{(\infty)})^T(\mathbf{Z} - \mathbf{X}^{(\infty)})) \le 0$ for all $\mathbf{Z} \in \mathcal{C}$, which is the KKT stationarity condition of* (6) *(for the proof, see Appendix A).*

4 The DASH-Net Architecture

4.1 The Role of the Starting Point

The guarantees of Subsection 3.1 hold from any feasible start. The objective f in (6) is *convex*, and a convex function maximized over a polytope attains its optimum at one of its vertices, of which the feasible set here has exponentially many. Maximizing f over this set is therefore a non-convex problem, so the initialization also matters, as it does for any method on a problem of this kind, and we treat the starting point as part of the design.

When labeled vertices are abundant the supervision term steers the ascent on its own, whereas in the extreme low-label regime it carries far less information, which is why the choice of supervision term in Section 2 matters so much. With the one-hop term $\mathbf{B}_{\text{lap}}$ the objective is numerically dominated by the label-agnostic balanced-cut term, while the global term $\mathbf{B}_{\text{poi}}$ of (3) makes the supervision visible at every vertex and so draws the ascent toward the labels. Appendix G sets the fixed iteration beside the unrolled schedule of Subsection 4.2 on the same graphs.

One response is to design a better $\mathbf{X}^{(0)}$, and Appendix D records a training-free construction, based on personalized PageRank [22] followed by a balanced entropic optimal-transport plan [6]. We go further and learn the trajectory itself, which is the construction of the next subsection.

4.2 Unrolling the DASH Iteration

We therefore replace the hand-fixed iteration by a learned one, following the algorithm-unrolling paradigm, in which each iteration of DASH SSL becomes one layer of a network and the quantities set

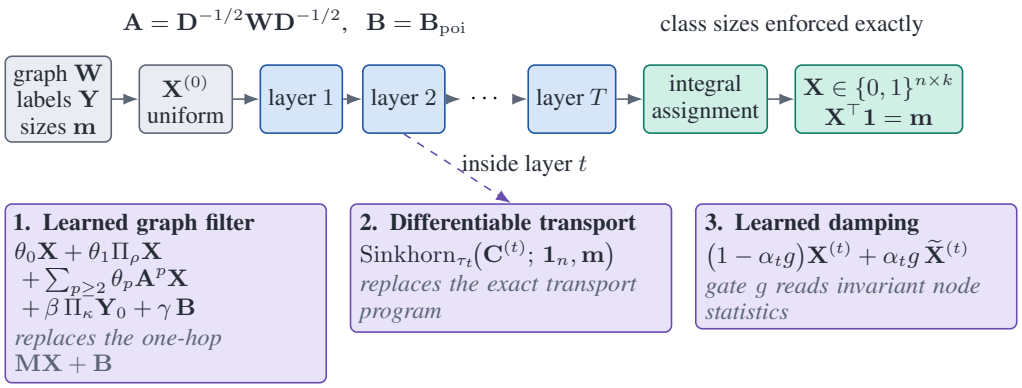


Figure 1: DASH-Net unrolls the DASH iteration into a network.

by hand become parameters fit to data. Written in terms of its cost matrix, each iteration forms $\mathbf{C}^{(t)} = \mathbf{M}\mathbf{X}^{(t)} + \mathbf{B}$, the gradient of f at the current labeling, and solves $\mathbf{X}^{(t+1)} = \arg\max_{\mathbf{X}\in\mathcal{C}} \operatorname{tr}(\mathbf{X}^T\mathbf{C}^{(t)})$, and **DASH-Net** generalizes each of the three ingredients, as illustrated in Figure 1. The first is **the cost operator**, in which $\mathbf{M} = s\mathbf{I} - \mathbf{L}_{uu}$ is a degree-one polynomial in $\mathbf{L}_{uu}$ and therefore carries information one hop per layer. DASH-Net replaces it by the learned filter of order P, $\mathbf{C}^{(t)} = \theta_0^{(t)}\mathbf{X}^{(t)} + \theta_1^{(t)}\Pi_{\rho_t}\mathbf{X}^{(t)} + \sum_{p=2}^{P}\theta_p^{(t)}\mathbf{A}^p\mathbf{X}^{(t)} + \beta_t\Pi_{\kappa_t}\mathbf{Y}_0 + \gamma_t\mathbf{B}$, in which $\mathbf{A}$ is the symmetrically normalized adjacency, Π_ρ is the personalized-PageRank diffusion with restart probability ρ, truncated to a fixed number of steps, $\mathbf{Y}_0$ carries the observed labels on the labeled vertices and zeros elsewhere, and ρ_t, κ_t, β_t and γ_t are learned per layer, so that information crosses the graph within a single layer and the global field $\mathbf{B}$ brings supervision to every vertex, which addresses the emptiness of the one-hop term directly. The second is **the assignment step**, where the exact transportation program is not differentiable and is replaced in each layer by a log-domain Sinkhorn plan $\widetilde{\mathbf{X}}^{(t)}$ at learned temperature τ_t, the standard entropic approximation, which recovers the vertex solution as $\tau_t \to 0$. The third is **the schedule**, where the hand-chosen iteration count and the implicit unit step become a learned damping α_t together with a per-vertex gate g, the layer returning $(1-\alpha_t g)\mathbf{X}^{(t)} + \alpha_t g\,\widetilde{\mathbf{X}}^{(t)}$, so that the rate at which the iterates sharpen is fit to the graph.

Two structural properties distinguish this model from a generic network applied to the same data. First, **class-permutation equivariance**, since the filter coefficients $\theta_p^{(t)}$ are scalars, not $k\times k$ matrices, so permuting the class labels permutes the output identically, while the gate, the only nonlinearity that sees class information, consumes permutation-*invariant* per-vertex statistics. Second, **each layer keeps the structure of a DASH iteration**, forming a cost matrix from the current labeling and the supervision and passing it through a balanced transport step, so that what is learned is how to run the iteration and not a map from vertex features to labels. Critically, **the exact class-size constraint is retained**, in that every layer forms a balanced transport plan with column marginal $\mathbf{m}$, and at inference a final integral assignment returns a labeling satisfying $\mathbf{X}\mathbf{1}_k = \mathbf{1}_n$ and $\mathbf{X}^T\mathbf{1}_n = \mathbf{m}$ exactly. Only the trajectory taken to reach a feasible point is learned, and the model is deliberately small, comprising $T(P+7)$ scalars and a gate network of 97 parameters, 169 in all at the depth $T = 8$ and filter order $P = 2$ used for every graph and label rate, too few to memorize a graph of the sizes considered here. Neither T nor P is a tuning parameter, since both are capacity limits fixed in advance. The depth bounds the number of steps, while the learned damping and gate decide how far each layer moves the iterate, a layer whose damping tends to zero leaving it unchanged, and the order bounds only the local polynomial part of the filter, whose coefficients may vanish, the global reach coming from the personalized-PageRank terms. The training protocol is stated in Appendix D.

5 Numerical Results

5.1 Datasets, Baselines and Experimental Setup

Our main evaluation is carried out on standard graph benchmarks in which the graph is *given* by the data and not constructed from learned features, namely the citation networks Cora, CiteSeer and

Table 1: Citation benchmarks.

(a) Cora

# Labels per class	1	2	3	4	5	RunTime (s)
Volume MBO (2018) [15]	46.00 ± 8.82	51.02 ± 7.35	57.12 ± 6.44	57.75 ± 5.08	58.86 ± 7.11	0.07
Poisson Learning (2020) [4]	38.66 ± 26.74	31.41 ± 21.85	23.20 ± 12.49	25.48 ± 20.97	20.54 ± 14.70	12.59
Correct & Smooth (2021) [14]	48.67 ± 9.35	54.22 ± 8.24	57.36 ± 4.96	58.15 ± 3.89	59.34 ± 4.79	0.32
p-Laplace (2022) [9]	51.40 ± 9.17	56.68 ± 6.57	59.36 ± 3.73	60.07 ± 5.21	60.51 ± 4.65	0.43
Procrustes-SSL (2023) [12]	48.45 ± 7.68	51.09 ± 9.14	52.56 ± 6.60	54.80 ± 7.50	50.14 ± 6.42	0.14
CutSSL (2024) [13]	55.57 ± 8.82	62.99 ± 8.31	67.34 ± 6.25	70.11 ± 3.36	71.40 ± 4.15	1.91
CombCut (2026) [21]	16.29 ± 7.35	19.01 ± 6.90	15.84 ± 7.70	17.36 ± 7.82	18.04 ± 8.02	2.78
GEM (2026) [18]	38.64 ± 7.79	44.92 ± 8.01	49.25 ± 8.02	56.05 ± 5.44	60.07 ± 5.94	—
DASH-Net (Ours)	$\mathbf{59.58 \pm 9.12}$	$\mathbf{66.18 \pm 6.60}$	$\mathbf{70.48 \pm 3.61}$	$\mathbf{71.99 \pm 2.50}$	$\mathbf{72.99 \pm 2.43}$	0.11

(b) CiteSeer

# Labels per class	1	2	3	4	5	RunTime (s)
Volume MBO (2018) [15]	29.18 ± 7.98	37.11 ± 5.21	38.00 ± 4.50	38.57 ± 6.21	42.76 ± 2.68	0.08
Poisson Learning (2020) [4]	20.65 ± 6.48	17.56 ± 2.77	18.46 ± 2.97	18.23 ± 2.99	19.28 ± 1.56	15.88
Correct & Smooth (2021) [14]	30.61 ± 6.90	34.68 ± 4.81	37.01 ± 4.30	38.33 ± 5.44	41.63 ± 13.44	0.33
p-Laplace (2022) [9]	29.62 ± 7.21	36.10 ± 5.04	38.11 ± 4.21	39.54 ± 5.62	42.87 ± 3.03	2.07
Procrustes-SSL (2023) [12]	31.63 ± 9.07	38.85 ± 7.21	41.46 ± 4.86	38.85 ± 6.43	43.11 ± 3.01	1.15
CutSSL (2024) [13]	35.34 ± 9.83	39.28 ± 7.99	41.73 ± 7.19	45.64 ± 6.23	47.73 ± 3.90	2.89
CombCut (2026) [21]	17.49 ± 4.21	18.99 ± 1.86	19.30 ± 2.33	19.36 ± 1.76	18.98 ± 3.09	4.20
GEM (2026) [18]	27.72 ± 5.48	38.98 ± 5.80	40.51 ± 5.91	$\mathbf{46.24 \pm 5.73}$	47.35 ± 4.69	—
DASH-Net (Ours)	$\mathbf{35.57 \pm 8.70}$	$\mathbf{42.76 \pm 6.93}$	$\mathbf{45.32 \pm 4.97}$	45.22 ± 5.83	$\mathbf{47.94 \pm 2.69}$	0.12

(c) PubMed

# Labels per class	1	2	3	4	5	RunTime (s)
Volume MBO (2018) [15]	50.64 ± 13.57	59.83 ± 11.00	60.36 ± 12.06	63.32 ± 8.18	65.54 ± 4.24	0.33
Poisson Learning (2020) [4]	38.67 ± 3.85	39.71 ± 0.29	38.73 ± 4.09	39.88 ± 0.36	40.68 ± 2.20	9.39
Correct & Smooth (2021) [14]	55.31 ± 10.88	62.38 ± 8.48	63.36 ± 7.19	63.09 ± 6.80	66.45 ± 4.97	0.24
p-Laplace (2022) [9]	56.60 ± 10.58	62.95 ± 8.84	63.03 ± 8.16	64.86 ± 5.90	65.07 ± 5.76	0.15
Procrustes-SSL (2023) [12]	47.18 ± 14.52	55.59 ± 10.30	60.17 ± 7.94	57.20 ± 11.06	54.13 ± 11.79	2.68
CutSSL (2024) [13]	$\mathbf{61.78 \pm 6.54}$	61.19 ± 5.77	62.87 ± 1.04	63.15 ± 1.87	63.65 ± 3.02	4.82
CombCut (2026) [21]	38.92 ± 4.16	37.93 ± 5.71	35.71 ± 7.46	34.08 ± 8.69	34.93 ± 8.16	10.90
GEM (2026) [18]	49.97 ± 7.25	52.38 ± 8.60	60.05 ± 7.45	61.60 ± 6.67	64.74 ± 6.04	—
DASH-Net (Ours)	59.53 ± 10.38	$\mathbf{66.50 \pm 7.04}$	$\mathbf{69.01 \pm 3.83}$	$\mathbf{65.94 \pm 7.91}$	$\mathbf{69.44 \pm 6.29}$	0.68

PubMed [28]. We evaluate on all three, as these are the benchmarks against which this literature has consistently measured itself [13, 14, 18, 21]. They span a useful range of scale and difficulty, from 2,708 vertices with $k = 7$ classes on Cora, through 3,327 vertices with $k = 6$ classes on CiteSeer, to 19,717 with $k = 3$ on PubMed, and, the adjacency structure being intrinsic, they isolate the optimizer from the quality of any feature-extraction pipeline. Appendix H reports image benchmarks, those most often repeated in this literature, built by nearest-neighbour search over learned representations. For each graph we use the adjacency structure supplied with the dataset, symmetrized where required, the class-size vector $\mathbf{m}$ is set from the true class priors, and degree-zero vertices are removed identically for every method; Appendix E explains why that preprocessing is necessary and records the remaining details.

One of the contributions of this work is that the evaluation is directed at the extreme low-label regime, using from one to five labeled samples per class. It is where the harmonic extension degenerates and where the supervision term is nearly empty, and it is for those very reasons the regime in which a graph-based method has to be shown to work. Performance is measured by the average top-1 classification accuracy (%) over 20 independent trials, with standard deviations reported to show the impact of random label selection. Every method except GEM, which is run through its own released pipeline as Appendix F describes, sees *identical* label draws on a given dataset and label rate, so that the differences among them are not attributable to the choice of labeled set. To guarantee the fairness of the comparison, every competing method is evaluated at the configuration its own paper prescribes.

We compare our framework against eight graph-based SSL methods, spanning the state-of-the-art and benchmark methods of this literature, namely Volume-MBO (2018) [15], Poisson Learning (2020) [4], Correct & Smooth (2021) [14], p-Laplace (2022) [9], Procrustes-SSL (2023) [12], CutSSL (2024) [13], CombCut (2026) [21] and GEM (2026) [18], which between them cover the propagation, spectral, relaxation-based and entropy-minimization families. Appendix F describes what each of them does, together with the weakness that our formulation is designed to remove.

5.2 RESULTS

Table 1 reports accuracy on the citation benchmarks for one to five labels per class. DASH-Net outperforms the state-of-the-art and benchmark methods on these graphs. It leads on Cora at every rate, by 4.01, 3.19, 3.14, 1.88 and 1.59 points over the strongest baseline in each column, on CiteSeer by 0.23, 3.48, 3.59 and 0.21 points at one, two, three and five labels per class, and on PubMed by 3.55, 5.65, 1.08 and 2.99 points from two labels per class upward. On CiteSeer with four labels per class and on PubMed with one, it is on a par with the leading method, within one standard deviation of it. The inference cost tells the same story, in that DASH-Net labels each of these graphs in a fraction of a second.

Two components of DASH-Net bear on these results, the supervision term, whose effect on these graphs Appendix I isolates, and the learned schedule, which Appendix G sets beside the fixed iteration of Algorithm 1 on the same graphs and label draws. The two are complementary, since the supervision term decides where the ascent is drawn while the schedule governs how reliably it arrives.

A third property of the table is not visible in the accuracy columns at all. Every labeling DASH-Net returns meets the prescribed class sizes exactly, since each layer transports mass under the class-size marginal and the final integral assignment respects it. Of the baselines, only CutSSL, CombCut and Volume-MBO build the prescribed class sizes into their formulation, and the others read a label off a continuous score at each vertex, so the proportions they produce are whatever those scores happen to give. DASH-Net obtains this guarantee between seven and thirty-five times faster than CutSSL and CombCut, labeling Cora in 0.11 seconds, CiteSeer in 0.12 and PubMed in 0.68, beside 1.91, 2.89 and 4.82 for CutSSL and 2.78, 4.20 and 10.90 for CombCut. The cheaper propagation and threshold-dynamics rules each reach a lower accuracy than DASH-Net in all cells, so the additional time, under a second on each graph, is what buys the accuracy Table 1 reports.

6 LIMITATIONS

Our problem is discrete and non-convex, a class in which no polynomial-time algorithm is known to guarantee a global minimizer [3]. We establish monotone ascent, and only convergence of the objective and stationarity of the limit points can be proved.

7 CONCLUSION

This paper introduced *DASH SSL*, a hyperparameter-free framework for graph-based semi-supervised learning that operates directly on discrete label assignments. Its core is a reformulation of the combinatorial graph-cut problem into a tractable objective, solved by an alternating algorithm whose label update is a linear program with a totally unimodular constraint set, so that the solution is integral and the class sizes hold exactly without any rounding or projection. We proved that the objective values increase monotonically, that the sequence is bounded and convergent, and that its limit points are KKT stationary, and the unrolled network keeps the exact class sizes of every labeling it returns while fitting its schedule to the graph. Evaluations on Cora, CiteSeer and PubMed showed state-of-the-art accuracy in the extreme low-label regime, with each graph labeled in a fraction of a second, several times faster than CutSSL and CombCut, and the appendices carry the study onto further benchmarks.

REPRODUCIBILITY STATEMENT

The problem formulation and the assumptions on which our theory rests are stated in Section 2, and complete proofs of Lemmas 1 and 2, of the total unimodularity of the constraint matrix, and of the monotonicity, boundedness and stationarity claims of Theorem 6 are given in Appendices A and B. The solver is stated as Algorithm 1 in Subsection 2.1, the architecture of the unrolled network in Subsection 4.2, and its training protocol in Appendix D. Appendix E records the experimental setup in the detail needed to repeat it, including the preprocessing applied identically to every method, the number of trials, the shared label draws and the hardware on which the reported timings were measured, while Appendix H gives the corresponding graph-construction details for the image benchmarks. Anonymous source code accompanies this submission as supplementary material. It

contains our implementation of the solver and of the unrolled network, the scripts that build every graph used in the paper and that run each experiment and emit the reported tables, and the exact label permutations on which our method and every paired baseline were evaluated, so that the paired trials can be reproduced draw for draw.

AI USE STATEMENT

AI-assisted editing was used for language editing and proofreading of the main paper and supplementary materials. The authors take full responsibility for the final content of this paper.

REFERENCES

[1] Rie K. Ando and Tong Zhang. Learning on graph with Laplacian regularization. *Advances in neural information processing systems*, 19, 2006.

[2] Dimitri P. Bertsekas, Angelia Nedić, and Asuman E. Ozdaglar. *Convex analysis and optimization*. Athena Scientific, 2003.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[4] Jeff Calder, Brendan Cook, Matthew Thorpe, and Dejan Slepcev. Poisson learning: Graph based semi-supervised learning at very low label rates. In *International conference on machine learning*, pp. 1306–1316. PMLR, 2020.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

[6] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

[7] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[8] Ahmed El Alaoui, Xiang Cheng, Aaditya Ramdas, Martin J Wainwright, and Michael I Jordan. Asymptotic behavior of ℓ_p-based Laplacian regularization in semi-supervised learning. In *Conference on Learning Theory*, pp. 879–906. PMLR, 2016.

[9] Mauricio Flores, Jeff Calder, and Gilad Lerman. Analysis and algorithms for ℓ_p-based semi-supervised learning on graphs. *Applied and Computational Harmonic Analysis*, 60:77–122, 2022.

[10] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

[11] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `https://cvxr.com/cvx`, March 2014.

[12] Chester Holtz, Pengwen Chen, Alexander Cloninger, Chung-Kuan Cheng, and Gal Mishne. Semi-supervised Laplace learning on Stiefel manifolds. *arXiv preprint arXiv:2308.00142*, 2023.

[13] Chester Holtz, Pengwen Chen, Zhengchao Wan, Chung-Kuan Cheng, and Gal Mishne. Continuous partitioning for graph-based semi-supervised learning. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[14] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R. Benson. Combining label propagation and simple models out-performs graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2021.

[15] Matt Jacobs, Ekaterina Merkurjev, and Selim Esedoḡlu. Auction dynamics: A volume constrained MBO scheme. *Journal of Computational Physics*, 354:288–310, 2018.

[16] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[18] Yi Luo, Xu Sun, Guangchun Luo, Ke Qin, and Aiguo Chen. Graph entropy minimization for semi-supervised node classification. *Neurocomputing*, 667:132359, 2026. doi: 10.1016/j.neucom.2025.132359.

[19] Boaz Nadler, Nathan Srebro, and Xueyuan Zhou. Semi-supervised learning with the graph Laplacian: The limit of infinite unlabelled data. *Advances in neural information processing systems*, 22:1330–1338, 2009.

[20] George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization*. John Wiley & Sons, 1999.

[21] Mohammad Mahdi Omati, Yasin Salajeghe, Mahshad Moradi, and Arash Amini. Exact combinatorial multi-class graph cuts for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 37117–37124, 2026. doi: 10.1609/aaai.v40i43.41041.

[22] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999.

[23] Astha Saini, Petre Stoica, Prabhu Babu, and Aakash Arora. Min-max framework for majorization-minimization algorithms in signal processing applications: An overview. *Foundations and Trends® in Signal Processing*, 18(4):310–389, 2024.

[24] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[25] Petre Stoica and Prabhu Babu. SPICE and LIKES: Two hyperparameter-free methods for sparse-parameter estimation. *Signal Processing*, 92(7):1580–1590, 2012.

[26] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[27] Xin Yang, Haoying Fu, Hongyuan Zha, and Jesse Barlow. Semi-supervised nonlinear dimensionality reduction. In *Proceedings of the 23rd international conference on Machine learning*, pp. 1065–1072, 2006.

[28] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pp. 40–48, 2016.

[29] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.

[30] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd international conference on Machine learning*, pp. 1036–1043, 2005.

[31] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

Supplementary Material

The appendices accompanying this paper are organized as follows. Appendix A proves the two lemmas of Subsection 2.1, the first-order condition of Proposition 5 and the convergence theorem of Subsection 3.1, while Appendix B establishes the integrality theorem together with the total unimodularity of the constraint matrix of the assignment subproblem. Appendix C derives the computational complexity of the solver, and Appendix D gives the initialization and training protocol of the unrolled network. Appendix E records the experimental setup and Appendix F describes the compared methods. Appendix G isolates the effect of initialization, Appendix H reports the image benchmarks, and Appendix I separates out the contribution of the supervision term.

A Proofs

A.1 Proof of Lemma 1

Lemma 1. *Let* $\mathbf{X} \in \{0,1\}^{n\times k}$ *satisfy* $\mathbf{X}\,\mathbf{1}_k = \mathbf{1}_n$. *Then* $\|\mathbf{X}\|_F^2 = n$.

Proof. By definition, $\|\mathbf{X}\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{k} X_{ij}^2$. Since $X_{ij} \in \{0,1\}$, it follows that $X_{ij}^2 = X_{ij}$, and thus $\|\mathbf{X}\|_F^2 = \sum_{i,j} X_{ij}$. The constraint $\mathbf{X}\mathbf{1}_k = \mathbf{1}_n$ implies that each row sums to one, which leads to $\sum_{i,j} X_{ij} = n$. ■

A.2 Proof of Lemma 2

Lemma 2. *For any given matrix* $\mathbf{X}$ *and a symmetric positive semi-definite matrix* $\mathbf{M}$, *we have* $\operatorname{tr}(\mathbf{X}^T\mathbf{M}\mathbf{X}) = \max_{\mathbf{Q}}(2\operatorname{tr}(\mathbf{Q}^T\mathbf{M}^{1/2}\mathbf{X}) - \operatorname{tr}(\mathbf{Q}^T\mathbf{Q}))$, *and the maximum is attained at the unique point* $\mathbf{Q}^* = \mathbf{M}^{1/2}\mathbf{X}$.

Proof. Let $\mathbf{X}$ be a fixed matrix. We identify the maximizer of $g(\mathbf{Q}) = 2\operatorname{tr}(\mathbf{Q}^T\mathbf{M}^{1/2}\mathbf{X}) - \operatorname{tr}(\mathbf{Q}^T\mathbf{Q})$ by completing the square. We first rearrange the objective function as

$$\begin{aligned} g(\mathbf{Q}) &= -\Big(\operatorname{tr}(\mathbf{Q}^T\mathbf{Q}) - 2\operatorname{tr}(\mathbf{Q}^T\mathbf{M}^{1/2}\mathbf{X})\Big) \\ &= -\Big(\operatorname{tr}(\mathbf{Q}^T\mathbf{Q}) - 2\operatorname{tr}(\mathbf{Q}^T\mathbf{M}^{1/2}\mathbf{X}) + \operatorname{tr}(\mathbf{X}^T\mathbf{M}\mathbf{X})\Big) + \operatorname{tr}(\mathbf{X}^T\mathbf{M}\mathbf{X}), \end{aligned} \tag{S1}$$

where we have added and subtracted the term $\operatorname{tr}(\mathbf{X}^T\mathbf{M}\mathbf{X})$. The terms grouped in the parenthesis can be written as the trace of a squared term, which corresponds to the squared Frobenius norm,

$$\operatorname{tr}\Big((\mathbf{Q} - \mathbf{M}^{1/2}\mathbf{X})^T(\mathbf{Q} - \mathbf{M}^{1/2}\mathbf{X})\Big) = \|\mathbf{Q} - \mathbf{M}^{1/2}\mathbf{X}\|_F^2, \tag{S2}$$

and substituting this expression into (S1) yields

$$g(\mathbf{Q}) = -\|\mathbf{Q} - \mathbf{M}^{1/2}\mathbf{X}\|_F^2 + \operatorname{tr}(\mathbf{X}^T\mathbf{M}\mathbf{X}). \tag{S3}$$

Because $\mathbf{X}$ is fixed, the term $\operatorname{tr}(\mathbf{X}^T\mathbf{M}\mathbf{X})$ is constant, and $g(\mathbf{Q})$ is therefore maximized when the squared Frobenius norm term is minimized. This minimum is achieved if and only if $\mathbf{Q} = \mathbf{M}^{1/2}\mathbf{X}$. By setting $\mathbf{Q}^* = \mathbf{M}^{1/2}\mathbf{X}$, the maximum value obtained is $g(\mathbf{Q}^*) = \operatorname{tr}(\mathbf{X}^T\mathbf{M}\mathbf{X})$, which completes the proof. ■

A.3 Proof of Proposition 5

Proposition 5. *Let* f *be continuously differentiable and* $\mathcal{C}$ *closed and convex. If* $\mathbf{X}^*$ *is a local maximizer of* f *over* $\mathcal{C}$, *then* $\operatorname{tr}(\nabla_{\mathbf{X}} f(\mathbf{X}^*)^T(\mathbf{Z} - \mathbf{X}^*)) \le 0$ *for all* $\mathbf{Z} \in \mathcal{C}$.

Proof. Fix $\mathbf{Z} \in \mathcal{C}$ and set $\mathbf{X}_\eta = \mathbf{X}^* + \eta(\mathbf{Z} - \mathbf{X}^*)$ for $\eta \in (0,1]$. Since $\mathcal{C}$ is convex, $\mathbf{X}_\eta \in \mathcal{C}$ for every such η, and since $\mathbf{X}^*$ is a local maximizer there is an $\bar\eta \in (0,1]$ with $f(\mathbf{X}_\eta) \le f(\mathbf{X}^*)$ for all $\eta \in (0,\bar\eta)$. Consequently

$$\frac{f(\mathbf{X}^* + \eta(\mathbf{Z} - \mathbf{X}^*)) - f(\mathbf{X}^*)}{\eta} \;\le\; 0, \qquad \eta \in (0,\bar\eta).$$

The left-hand side tends to the directional derivative $\operatorname{tr}(\nabla_{\mathbf{X}} f(\mathbf{X}^*)^T(\mathbf{Z}-\mathbf{X}^*))$ as $\eta \downarrow 0$, because f is continuously differentiable, and a limit of non-positive quantities is non-positive. As $\mathbf{Z} \in \mathcal{C}$ was arbitrary, the stated condition holds throughout $\mathcal{C}$. ■

A.4 Proof of Theorem 6

Part (a) of the theorem follows from the two maximization steps of the alternating scheme together with the identity $f(\mathbf{X}) = G(\mathbf{X}, \mathbf{M}^{1/2}\mathbf{X})$; part (b) from the compactness of $\mathcal{C}$, the integrality that Theorem 4 supplies at every iteration and the rule that breaks ties in favor of the current iterate; and part (c) from Proposition 5 applied to the linear program that defines the $\mathbf{X}$-update, whose gradient $\nabla_{\mathbf{X}} f(\mathbf{X}) = \mathbf{M}\mathbf{X} + \mathbf{B}$ is exactly the cost matrix of that program.

Proof. We prove the three parts in turn.

(a) Monotonicity. At iteration t, the algorithm performs two maximization steps on the joint objective G of (10). The first is the update $\mathbf{Q}^{(t+1)} = \arg\max_{\mathbf{Q}} G(\mathbf{X}^{(t)}, \mathbf{Q})$, which gives $G(\mathbf{X}^{(t)}, \mathbf{Q}^{(t+1)}) \geq G(\mathbf{X}^{(t)}, \mathbf{Q}^{(t)})$. The second is the update $\mathbf{X}^{(t+1)} = \arg\max_{\mathbf{X}\in\mathcal{C}} G(\mathbf{X}, \mathbf{Q}^{(t+1)})$, which gives $G(\mathbf{X}^{(t+1)}, \mathbf{Q}^{(t+1)}) \geq G(\mathbf{X}^{(t)}, \mathbf{Q}^{(t+1)})$. Combining the two inequalities yields $G(\mathbf{X}^{(t+1)}, \mathbf{Q}^{(t+1)}) \geq G(\mathbf{X}^{(t)}, \mathbf{Q}^{(t)})$. By Lemma 2, substituting the optimal $\mathbf{Q}$ back into G recovers the original objective, that is, $f(\mathbf{X}) = G(\mathbf{X}, \mathbf{M}^{1/2}\mathbf{X})$. We therefore obtain the ascent property

$$\begin{aligned} f(\mathbf{X}^{(t+1)}) &= G(\mathbf{X}^{(t+1)}, \mathbf{Q}^{(t+2)}) \\ &\geq G(\mathbf{X}^{(t+1)}, \mathbf{Q}^{(t+1)}) \\ &\geq G(\mathbf{X}^{(t)}, \mathbf{Q}^{(t+1)}) = f(\mathbf{X}^{(t)}), \end{aligned} \tag{S4}$$

so that the sequence of objective values $\{f(\mathbf{X}^{(t)})\}$ is non-decreasing.

(b) Boundedness and finite termination. The constraint set $\mathcal{C}$ is compact and f is continuous, so f is bounded above on $\mathcal{C}$. A non-decreasing real sequence that is bounded above converges, and hence $\{f(\mathbf{X}^{(t)})\}$ converges. Furthermore, because the constraint matrix defining the feasible region for $\mathbf{X}$ is totally unimodular by the argument of Appendix B, the linear program solved at each step has an integral optimal vertex, which a simplex-type or network-flow method returns, so that $\mathbf{X}^{(t+1)} \in \{0,1\}^{n\times k}$. The set of feasible integral matrices is finite, so the non-decreasing sequence $\{f(\mathbf{X}^{(t)})\}$ can increase strictly only finitely many times. Once $f(\mathbf{X}^{(t+1)}) = f(\mathbf{X}^{(t)})$, both inequalities of part (a) hold with equality, so that $G(\mathbf{X}^{(t)}, \mathbf{Q}^{(t+1)}) = G(\mathbf{X}^{(t+1)}, \mathbf{Q}^{(t+1)})$ and $\mathbf{X}^{(t)}$ is itself a maximizer of the $\mathbf{X}$-update; with ties broken in favor of the current iterate, $\mathbf{X}^{(t+1)} = \mathbf{X}^{(t)}$, and the iteration has reached a fixed point after finitely many steps.

(c) Stationarity. Let the algorithm reach a fixed point $\mathbf{X}^{(\infty)}$, which by part (b) is guaranteed to happen in a finite number of steps, so that $\mathbf{X}^{(\infty)} = \mathbf{X}^{(t+1)} = \mathbf{X}^{(t)}$. Substituting the $\mathbf{Q}$-update into the $\mathbf{X}$-update, the rule for $\mathbf{X}$ reads

$$\mathbf{X}^{(t+1)} = \arg\max_{\mathbf{X}\in\mathcal{C}} \operatorname{tr}\left(\mathbf{X}^T(\mathbf{M}\mathbf{X}^{(t)} + \mathbf{B})\right),$$

which at the fixed point becomes $\mathbf{X}^{(\infty)} = \arg\max_{\mathbf{X}\in\mathcal{C}} \operatorname{tr}(\mathbf{X}^T(\mathbf{M}\mathbf{X}^{(\infty)} + \mathbf{B}))$. This is a linear program over the convex set $\mathcal{C}$, and by Proposition 5 its first-order optimality condition at the solution $\mathbf{X}^{(\infty)}$ is

$$\operatorname{tr}\big((\mathbf{M}\mathbf{X}^{(\infty)} + \mathbf{B})^T(\mathbf{Z} - \mathbf{X}^{(\infty)})\big) \leq 0, \quad \forall \mathbf{Z} \in \mathcal{C}.$$

The gradient of the objective is $\nabla_{\mathbf{X}} f(\mathbf{X}) = \mathbf{M}\mathbf{X} + \mathbf{B}$, so that the condition above is exactly $\operatorname{tr}(\nabla_{\mathbf{X}} f(\mathbf{X}^{(\infty)})^T(\mathbf{Z} - \mathbf{X}^{(\infty)})) \leq 0$ for all $\mathbf{Z} \in \mathcal{C}$, which is the KKT stationarity condition for the problem of maximizing f over $\mathcal{C}$. Since f is convex, its maximum over the polytope $\mathcal{C}$ is attained at a vertex, and the vertices of $\mathcal{C}$ are exactly the feasible points of (6) by Theorem 4, so this is the stationarity notion appropriate to (6). Any fixed point of the iteration is therefore a KKT stationary point. ■

B TOTAL UNIMODULARITY OF THE ASSIGNMENT SUBPROBLEM

We first prove the integrality theorem stated in Subsection 2.1, and then verify its hypothesis for the subproblem (8).

Theorem 4. *If* $\mathbf{D}$ *is TU and* $\mathbf{b}$ *is an integral vector, then the polyhedron* $P = \{\mathbf{x} : \mathbf{D}\mathbf{x} \leq \mathbf{b},\ 0 \leq \mathbf{x} \leq 1\}$ *contains only integral vertices, and consequently any linear program over this system has an integral optimal solution.*

Proof. Write the full system as $\widetilde{\mathbf{D}}\mathbf{x} \leq \widetilde{\mathbf{b}}$ with $\widetilde{\mathbf{D}} = [\mathbf{D}^T,\ \mathbf{I}^T,\ -\mathbf{I}^T]^T$ and $\widetilde{\mathbf{b}} = [\mathbf{b}^T,\ \mathbf{1}^T,\ \mathbf{0}^T]^T$, so that $P = \{\mathbf{x} : \widetilde{\mathbf{D}}\mathbf{x} \leq \widetilde{\mathbf{b}}\}$. Appending the rows of an identity matrix and of its negation to a totally unimodular matrix preserves total unimodularity, since any square submatrix of $\widetilde{\mathbf{D}}$ either lies in $\mathbf{D}$, or has a row containing a single non-zero entry of ± 1, along which its determinant expands to a determinant of a smaller submatrix of the same kind; $\widetilde{\mathbf{D}}$ is therefore TU, and $\widetilde{\mathbf{b}}$ is integral because $\mathbf{b}$ is.

Let $\mathbf{x}^*$ be a vertex of P, and let N denote the dimension of $\mathbf{x}$. By the definition of a vertex, the rows of $\widetilde{\mathbf{D}}$ that are active at $\mathbf{x}^*$ contain N linearly independent rows; collecting them into a nonsingular matrix $\widetilde{\mathbf{D}}_B \in \mathbb{R}^{N \times N}$ with the corresponding entries $\widetilde{\mathbf{b}}_B$, the vertex is the unique solution of $\widetilde{\mathbf{D}}_B\mathbf{x} = \widetilde{\mathbf{b}}_B$. By Cramer's rule its i-th coordinate is

$$x_i^* \ = \ \frac{\det\left(\widetilde{\mathbf{D}}_B^{(i)}\right)}{\det\left(\widetilde{\mathbf{D}}_B\right)},$$

where $\widetilde{\mathbf{D}}_B^{(i)}$ is $\widetilde{\mathbf{D}}_B$ with its i-th column replaced by $\widetilde{\mathbf{b}}_B$. Total unimodularity forces $\det(\widetilde{\mathbf{D}}_B) \in \{-1, +1\}$, as the submatrix is nonsingular, while $\det(\widetilde{\mathbf{D}}_B^{(i)})$ is an integer because every entry of $\widetilde{\mathbf{D}}_B^{(i)}$ is an integer. Hence $x_i^* \in \mathbb{Z}$ for every i, and with $0 \leq \mathbf{x}^* \leq 1$ this gives $\mathbf{x}^* \in \{0, 1\}^N$.

Finally, P is bounded by the box constraints and is therefore a polytope, so a linear program over P, whenever it is feasible, attains its optimum at a vertex of P. Every such vertex is integral by the argument above, which proves the claim. ■

To apply Theorem 4 to the subproblem (8), we vectorize the assignment matrix $\mathbf{X} \in \mathbb{R}^{n \times k}$ into a vector $\mathbf{x} = \mathrm{vec}(\mathbf{X}) \in \mathbb{R}^{nk}$. This allows the affine constraints of (8) to be expressed as a system of linear inequalities, $\mathbf{D}\,\mathbf{x} \leq \mathbf{b}$.

The row-sum constraint $\mathbf{X}\,\mathbf{1}_k = \mathbf{1}_n$ is equivalent to $(\mathbf{I}_n \otimes \mathbf{1}_k^T)\,\mathbf{x} = \mathbf{1}_n$, which can be represented as the pair of inequalities

$$(\mathbf{I}_n \otimes \mathbf{1}_k^T)\,\mathbf{x} \leq \mathbf{1}_n, \quad -(\mathbf{I}_n \otimes \mathbf{1}_k^T)\,\mathbf{x} \leq -\mathbf{1}_n.$$

Similarly, the column-sum constraint $\mathbf{X}^T\mathbf{1}_n = \mathbf{m}$ translates to the vectorized form $(\mathbf{1}_n^T \otimes \mathbf{I}_k)\,\mathbf{x} = \mathbf{m}$, which is enforced by the inequalities

$$(\mathbf{1}_n^T \otimes \mathbf{I}_k)\,\mathbf{x} \leq \mathbf{m}, \quad -(\mathbf{1}_n^T \otimes \mathbf{I}_k)\,\mathbf{x} \leq -\mathbf{m}.$$

These constraints are combined into a single system in which the matrix $\mathbf{D}$ and the vector $\mathbf{b}$ are given by

$$\mathbf{D} \ = \ \begin{bmatrix} \mathbf{I}_n \otimes \mathbf{1}_k^\top \\ -(\mathbf{I}_n \otimes \mathbf{1}_k^\top) \\ \mathbf{1}_n^\top \otimes \mathbf{I}_k \\ -(\mathbf{1}_n^\top \otimes \mathbf{I}_k) \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{1}_n \\ -\mathbf{1}_n \\ \mathbf{m} \\ -\mathbf{m} \end{bmatrix}.$$

The linear programming relaxation of the $\mathbf{X}$ subproblem can thus be written as

$$\begin{aligned} \max_{\mathbf{x} \in \mathbb{R}^{nk}} \quad & (\mathrm{vec}(\mathbf{M}^{1/2}\mathbf{Q}^{(t+1)} + \mathbf{B}))^T\mathbf{x} \\ \text{s.t.} \quad & \mathbf{D}\,\mathbf{x} \leq \mathbf{b}, \qquad 0 \leq \mathbf{x} \leq 1. \end{aligned} \tag{S5}$$

The matrix $\mathbf{D}$ stacks $\mathbf{I}_n \otimes \mathbf{1}_k^T$ and $\mathbf{1}_n^T \otimes \mathbf{I}_k$ together with their negations. The first two blocks together form the vertex–edge incidence matrix of the complete bipartite graph between the n unlabeled

vertices and the k classes, since the column of each pair (i, j) carries exactly one unit entry in each block, and the incidence matrix of a bipartite graph is totally unimodular [24]. Appending the negation of each row preserves the property, because a square submatrix that contains a row together with its negation is singular, and otherwise equals, up to the signs of its rows, a square submatrix of the incidence matrix; the full matrix $\mathbf{D}$ is therefore TU. As the vector $\mathbf{b}$ contains only integer entries, Theorem 4 applies, which guarantees that all vertices of the feasible polyhedron are integral, so that an optimal vertex $\mathbf{x}^*$ satisfies the binary constraint $\mathbf{x}^* \in \{0, 1\}^{nk}$.

C Computational Complexity

At each iteration the primary tasks are the update of the auxiliary variable and the solution of the linear program for $\mathbf{X}$. The auxiliary variable is never formed explicitly, so no matrix square root is computed, since the cost matrix of that program is $\mathbf{C}^{(t+1)} = \mathbf{M}^{1/2}\mathbf{Q}^{(t+1)} + \mathbf{B} = \mathbf{M}\mathbf{X}^{(t)} + \mathbf{B}$, whose dominant operation is the multiplication of the sparse $n \times n$ matrix $\mathbf{M}$ by the $n \times k$ matrix $\mathbf{X}^{(t)}$. Since every vertex has on the order of k' neighbours, as in a k'-nearest-neighbour graph, $\mathbf{M}$ has $\mathrm{nnz}(\mathbf{M}) = O(k'n)$ non-zero entries, so that this multiplication costs $O(k'nk)$, which simplifies to $O(k^2n)$ when k' is on the order of k. The second step is a variant of the transportation problem; while the worst-case complexity of a general linear program is polynomial, specialized algorithms such as the network simplex or efficient interior-point methods solve this class of problem very quickly in practice [3], and we denote its exact cost by $\mathrm{Cost}_{\mathrm{LP}}(n, k)$. The per-iteration cost is therefore $O(k^2n + \mathrm{Cost}_{\mathrm{LP}}(n, k))$, the memory footprint is $O(k'n + nk)$ and thus linear in the number of vertices, and the total complexity is $O(T(k^2n + \mathrm{Cost}_{\mathrm{LP}}(n, k)))$, where T is the number of iterations to stationarity.

The iteration count T is set by the data and is not fixed in advance. The cost operator $\mathbf{M} = s\mathbf{I} - \mathbf{L}_{uu}$ has the sparsity pattern of the graph, so that after t iterations a vertex can have been influenced only by vertices within t hops, and supervision has to traverse a distance on the order of the diameter of the graph before the assignment stabilizes. An unrolled schedule has a depth chosen in advance and independent of n, so that its cost is linear in the number of vertices, which is one of the concrete motivations for Section 4.

D Initialization and Training Protocol of DASH-Net

Before describing how the unrolled schedule is fit, we record the training-free seed referred to in Subsection 4.1. It diffuses the labels over the graph by personalized PageRank [22] and converts the resulting class affinities into a feasible soft assignment with a balanced entropic optimal-transport plan [6], which respects the class-size marginal by construction.

DASH-Net is trained *per dataset*, so that for each benchmark the unrolled schedule is fit on the graph of that benchmark, with the same training configuration at every rate from one to five labels per class. Training is episodic, in that each round samples a subgraph, reveals between one and three labels per class on it, runs the T unrolled layers, and takes a cross-entropy loss on a random sample of the vertices of that subgraph whose labels were not revealed, updating the filter coefficients $\boldsymbol{\theta}^{(t)}$, the restart probabilities ρ_t and κ_t, the supervision weights β_t and γ_t, the temperatures τ_t, the damping α_t and the gate network. What is fit is a schedule and not a labeling. The 169 parameters are shared by every vertex and every class, so they carry no per-vertex state and have no means of storing the labels of a graph, and what they encode are graph-level quantities, namely how far supervision should diffuse and how quickly the plan should concentrate at each layer, of the kind that propagation methods fix by hand, such as a diffusion time or a propagation coefficient. Nothing in the parameterization ties the schedule to the graph on which it was fit, so it applies without change to any other graph. At test time the network sees only the one to five labels per class of the shared draws, the same labeled sets as every paired method in the comparison.

E Experimental Protocol

This appendix records the details of the experimental setup to which Section 5 refers.

E.1 Preprocessing

We remove degree-zero vertices from every graph before evaluation, and CiteSeer has 48 such vertices while Cora and PubMed have none. This step matters for correctness. A vertex of degree zero is not merely unreachable by label propagation, since the normalized Laplacian raises the degree to a negative power, so that an isolated vertex contributes a division by zero and the resulting non-finite entries propagate through the linear solves until a single constant class is returned for the entire graph, which measures a numerical failure and not the propagation rule under study. Removal is standard practice on these benchmarks and is applied identically to every method compared. After it, Cora has 78 connected components and CiteSeer 390, which is why the sources in (3) are centered within each component, while PubMed and every image graph are connected.

E.2 Trials, Label Draws and Hardware

Every cell of every table is an average over 20 independent trials at the stated label rate, and every method other than GEM sees *identical* label draws on a given dataset and rate, so that the paired differences reported in the main text are not attributable to the choice of labeled set. The label permutations are released with the accompanying code, so that the paired trials can be reproduced draw for draw. All reported inference timings were measured on a server with a dual-socket Intel Xeon E5-2695 v3 CPU, on which our implementation requires at most 5 GB of memory, so that it can be run on modestly provisioned machines. Training of the unrolled network, which is carried out once per dataset, and of the methods in the comparison that train a network of their own, was performed on an NVIDIA RTX 3090 GPU, and the timings in the tables are per labeling at inference.

F Compared Methods

This appendix describes the eight baselines of Subsection 5.1, listed from oldest to newest, each with the weakness in the extreme low-label regime that our formulation is designed to remove.

Volume-MBO (2018) [15] descends on a threshold-dynamics approximation of an interface energy, not on the supervised objective of (4), thresholding a graph diffusion through an auction under volume constraints, and at one label per class that diffusion has almost nothing to spread.

Poisson Learning (2020) [4] replaces label values at the labeled vertices by sources and sinks and solves the resulting graph Poisson equation. Its sources are centered on the global mean label, which leaves the net mass of each connected component unconstrained on a fragmented graph such as Cora or CiteSeer, and the labeling is still read off the largest score at each vertex, so the class proportions it returns are whatever those scores give.

Correct & Smooth (2021) [14] spreads the residuals of a simple base classifier over the graph and then smooths the corrected scores. Its accuracy is inherited from the base predictor it corrects, which is exactly the component that has almost nothing to work with when one label per class is observed, and the propagation it applies afterwards carries no class-size requirement of any kind.

p-Laplace (2022) [9] is a non-linear label propagation method built on the p-Laplacian operator. Raising p damps the spikes of the harmonic extension, but the exponent is a parameter that has to be selected, and the method still returns a continuous score from which a label must be read off, with no control over the class proportions.

Procrustes-SSL (2023) [12] aligns a Laplacian eigenmap to the labeled vertices by an orthogonal Procrustes transform, formulating SSL on the Stiefel manifold. Orthogonality is a weaker requirement than one-hot assignment, so the method solves a relaxation, in the sense of losing tightness, of the labeling problem, and the spectral embedding it starts from fixes in advance how much of the graph the method can see, through a subspace dimension that is itself a choice. It is run from an implementation that follows its paper, and on the fragmented citation graphs its spectral step discards the whole null space of the Laplacian, one indicator per connected component, which extends its construction to graphs that are not connected.

CutSSL (2024) [13] relaxes the binary constraint of the cardinality-constrained minimum cut to the box $[0, 1]$, in the sense of losing tightness, and perturbs the objective by a multiple of the degree matrix until its minimizers become integral. Integrality therefore rests on the size of that shift, the

Table S1: Initialization study.

(a) Cora

# Labels per class	1	2	3	4	5	RunTime (s)
DASH SSL (Alg. 1)	29.64 ± 8.90	31.66 ± 9.69	42.70 ± 9.34	43.12 ± 9.02	45.40 ± 9.08	0.15
DASH-Net (Ours)	$\mathbf{59.58 \pm 9.12}$	$\mathbf{66.18 \pm 6.60}$	$\mathbf{70.48 \pm 3.61}$	$\mathbf{71.99 \pm 2.50}$	$\mathbf{72.99 \pm 2.43}$	0.11

(b) CiteSeer

# Labels per class	1	2	3	4	5	RunTime (s)
DASH SSL (Alg. 1)	26.55 ± 8.69	32.11 ± 10.03	31.19 ± 6.48	33.15 ± 5.93	33.79 ± 6.45	0.27
DASH-Net (Ours)	$\mathbf{35.57 \pm 8.70}$	$\mathbf{42.76 \pm 6.93}$	$\mathbf{45.32 \pm 4.97}$	$\mathbf{45.22 \pm 5.83}$	$\mathbf{47.94 \pm 2.69}$	0.12

(c) PubMed

# Labels per class	1	2	3	4	5	RunTime (s)
DASH SSL (Alg. 1)	43.14 ± 7.76	51.16 ± 8.65	53.11 ± 11.03	50.39 ± 9.81	51.80 ± 10.34	0.93
DASH-Net (Ours)	$\mathbf{59.53 \pm 10.38}$	$\mathbf{66.50 \pm 7.04}$	$\mathbf{69.01 \pm 3.83}$	$\mathbf{65.94 \pm 7.91}$	$\mathbf{69.44 \pm 6.29}$	0.68

primary hyperparameter of the method, and the value adopted in practice falls short of the condition under which the minimizers are guaranteed to be integral.

CombCut (2026) [21] leaves the objective unperturbed and descends on it with projections onto the transportation polytope. The perturbation nonetheless reaches the answer, since the descent is started from the solution of the perturbed program above, so that the parameter governing that program acts on the final labeling through the initialization.

GEM (2026) [18] trains a one-hop classifier by minimizing the entropy of predictions over edges, drawing supervision from uncategorized vertices. Entropy minimization rewards confident predictions whether or not they are correct, which is least safe precisely when supervision is scarcest, and the class proportions are never constrained at any stage. It is evaluated through its own released pipeline, which draws its own labeled sets, selects its training epoch on a labeled validation set of 500 vertices and scores $1{,}000$ held-out vertices, so its row is not paired with the shared label draws used for the remaining rows.

Volume-MBO is reported wherever it completed the full set of 20 paired trials, which it does on all three citation benchmarks and on Extended Yale B. Two gaps remain. On SVHN, the largest graph in this study at $n = 99{,}289$, the method did not return within a ten-hour budget, and it is therefore absent from that block of Table S2; on Fashion-MNIST it did not complete the twenty trials at one label per class, and that cell alone carries a dash while the remaining four rates are reported. It is worth mentioning that the method is also omitted from the citation-network experiments of CutSSL, owing to runtime errors on those graphs [13].

G Initialization Study

Table S1 sets the fixed iteration of Algorithm 1 beside DASH-Net. Both work on the same graph with the same supervision term and the same class-size vector, and both return an integral labeling that meets those class sizes exactly. The fixed iteration ascends on f from an uninformative feasible seed, while DASH-Net replaces its one-hop operator by a learned diffusion and follows a schedule whose coefficients and temperatures were fit by the protocol of Appendix D. On Cora that difference is worth between 27.6 and 34.5 points depending on the label rate; on PubMed between 15.3 and 17.6; and on CiteSeer between 9.0 and 14.2.

This is the empirical content of the argument in Subsection 4.1. Maximizing a convex function over a polytope is a non-convex problem, so the initialization also matters, and learning the trajectory is what recovers that difference, while every labeling DASH-Net returns still meets the prescribed class sizes exactly.

H IMAGE BENCHMARKS

This appendix reports the same evaluation on benchmarks whose graph is not intrinsic to the data but is *constructed* by nearest-neighbour search over learned representations. The setting is included for continuity with prior work on graph-based SSL, which has largely been benchmarked this way.

H.1 GRAPH CONSTRUCTION

The experiments use Fashion-MNIST [26] and SVHN. For Fashion-MNIST, feature representations were extracted using a variational autoencoder [16] with fully connected layers of sizes $(784, 400, 30)$, trained for 100 epochs, giving $n = 70{,}000$ vertices. For SVHN, representations were obtained by pretraining a SimCLR model [5] with self-supervised contrastive learning on all images without labels, using the encoder representation and not the projection head, which gives $n = 99{,}289$ vertices with $k = 10$ classes. Extended Yale B [10] adds a third regime, with $n = 2{,}414$ images of $k = 38$ subjects in a fixed frontal pose under 64 illumination directions, so that its difficulty is illumination, not identity. Raw pixels are a poor metric there, since neighbours link by illumination and give a 10-nearest-neighbour label agreement of 41.6%, and we therefore encode each image by its unit image-gradient field followed by whitened PCA to 75 dimensions. Under the Lambertian model a change of illumination rescales intensities, and the gradient *direction* is invariant to any monotone rescaling while its magnitude is not, so that discarding the magnitude removes the nuisance the database is built around and raises the 10-nearest-neighbour agreement to 96.7%. The encoding itself uses no labels. Its PCA dimension sits on a plateau of the 10-nearest-neighbour agreement, 96.7% at 75 dimensions against 96.5% at 100 and 95.7% at 125, and the resulting graph is shared by every method in the comparison.

The three benchmarks cover distinct regimes. Fashion-MNIST is the image benchmark most often repeated in this literature, the graph of SVHN is poorly conditioned, with a spectral gap of only $\lambda_2 = 3.4 \times 10^{-4}$, which stresses iterative solvers in a way the other benchmarks do not, and Extended Yale B isolates variation in illumination.

For all three datasets, graphs were constructed using a k'-nearest neighbour approach with $k' = 10$, and edge weights were computed using the Gaussian kernel

$$w_{ij} = \exp\big(-4\|v_i - v_j\|^2 / d_{k'}(v_i)^2\big),$$

where v_i is the latent feature vector for image i, and $d_{k'}(v_i)$ is the latent-space distance to its k'-th nearest neighbour. The resulting weight matrix $\mathbf{W}$ was symmetrized via $\mathbf{W} \leftarrow \frac{1}{2}(\mathbf{W} + \mathbf{W}^T)$.

Because these graphs are built from representations that were themselves learned on the same images, the constructed neighbourhoods already encode a great deal of class information before any label propagation takes place. Accuracies are correspondingly high and the spread between methods is compressed relative to the intrinsic graphs of Section 5.

H.2 RESULTS

Table S2 reports the image benchmarks. Against the strongest baseline in each column, DASH-Net leads at every rate on Fashion-MNIST and at the three sparsest rates on SVHN, the regime this paper is directed at, and at the two densest rates of SVHN it is on a par with the leading method, within one standard deviation of it. On Extended Yale B the gradient encoding leaves the graph so clean that Poisson learning, CutSSL and DASH-Net all exceed 99.5% at one label per class and remain within two tenths of a point of one another at every rate, with DASH-Net the most accurate at one and three labels per class. On every image graph DASH-Net returns a labeling that meets the class sizes exactly, and on SVHN, the largest graph of this study, it does so in 7.47 seconds per trial, against 68.10 for Poisson learning and 129.74 for CutSSL. Appendix I isolates the contribution of the supervision term on these graphs.

I ISOLATING THE SUPERVISION TERM

The framework of Subsection 2.1 depends on $\mathbf{B}$ only through the linearity of $\mathrm{tr}(\mathbf{X}^T\mathbf{B})$, so the supervision term can be changed without touching the auxiliary-variable reformulation, the transportation

Table S2: Image benchmarks.

(a) FashionMNIST-VAE

# Labels per class	1	2	3	4	5	RunTime (s)
Volume MBO (2018) [15]	—	60.57 ± 5.60	65.12 ± 3.23	69.29 ± 3.12	70.08 ± 1.67	3.11
Poisson Learning (2020) [4]	62.65 ± 4.09	66.60 ± 4.73	70.90 ± 1.93	72.59 ± 3.01	72.75 ± 2.04	21.14
Correct & Smooth (2021) [14]	53.11 ± 3.87	58.86 ± 4.48	63.73 ± 2.38	66.68 ± 2.61	67.73 ± 1.73	0.42
p-Laplace (2022) [9]	52.49 ± 3.31	57.64 ± 4.77	62.15 ± 2.57	64.98 ± 2.57	66.21 ± 1.67	4.71
Procrustes-SSL (2023) [12]	55.28 ± 3.86	63.24 ± 5.35	65.22 ± 3.39	67.89 ± 3.28	68.06 ± 2.80	3.55
CutSSL (2024) [13]	60.85 ± 4.26	66.83 ± 5.29	71.12 ± 2.70	72.27 ± 3.15	73.11 ± 2.52	101.26
CombCut (2026) [21]	61.76 ± 3.60	66.03 ± 4.47	70.50 ± 2.11	71.95 ± 2.78	72.54 ± 1.80	23.86
DASH-Net (Ours)	$\mathbf{62.85 \pm 4.97}$	$\mathbf{67.47 \pm 5.17}$	$\mathbf{71.82 \pm 2.15}$	$\mathbf{72.95 \pm 3.22}$	$\mathbf{73.41 \pm 2.08}$	3.81

(b) SVHN-SimCLR

# Labels per class	1	2	3	4	5	RunTime (s)
Poisson Learning (2020) [4]	42.83 ± 4.55	55.35 ± 7.34	58.34 ± 6.51	62.41 ± 4.65	64.52 ± 5.65	68.10
Correct & Smooth (2021) [14]	35.69 ± 3.78	49.19 ± 5.72	56.10 ± 4.60	62.72 ± 4.73	64.53 ± 2.65	0.50
p-Laplace (2022) [9]	37.80 ± 3.53	49.23 ± 4.72	55.31 ± 4.99	62.70 ± 4.40	65.49 ± 2.94	7.71
Procrustes-SSL (2023) [12]	40.01 ± 5.14	47.53 ± 4.47	52.02 ± 6.32	55.52 ± 4.98	55.80 ± 4.63	9.12
CutSSL (2024) [13]	36.40 ± 2.66	49.39 ± 6.12	58.13 ± 5.67	64.03 ± 5.75	68.57 ± 4.86	129.74
CombCut (2026) [21]	42.11 ± 4.22	55.05 ± 6.72	63.08 ± 6.45	$\mathbf{68.48 \pm 5.56}$	$\mathbf{71.25 \pm 4.73}$	90.86
DASH-Net (Ours)	$\mathbf{44.21 \pm 4.56}$	$\mathbf{58.28 \pm 6.95}$	$\mathbf{63.88 \pm 6.72}$	65.10 ± 5.29	67.48 ± 6.03	7.47

(c) Extended Yale B

# Labels per class	1	2	3	4	5	RunTime (s)
Volume MBO (2018) [15]	92.90 ± 1.95	95.97 ± 2.31	97.59 ± 1.16	98.58 ± 0.80	99.06 ± 0.59	1.46
Poisson Learning (2020) [4]	99.55 ± 0.28	99.69 ± 0.07	99.66 ± 0.07	99.68 ± 0.06	99.67 ± 0.06	0.26
Correct & Smooth (2021) [14]	97.22 ± 1.14	98.93 ± 0.37	99.27 ± 0.36	99.42 ± 0.20	99.60 ± 0.12	0.31
p-Laplace (2022) [9]	95.08 ± 1.02	96.50 ± 0.78	96.89 ± 0.52	97.31 ± 0.55	97.55 ± 0.57	0.49
Procrustes-SSL (2023) [12]	95.58 ± 0.92	97.89 ± 1.17	98.91 ± 0.69	99.26 ± 0.57	99.50 ± 0.29	0.05
CutSSL (2024) [13]	99.58 ± 0.19	$\mathbf{99.72 \pm 0.07}$	99.72 ± 0.06	$\mathbf{99.74 \pm 0.04}$	$\mathbf{99.75 \pm 0.06}$	2.00
CombCut (2026) [21]	98.43 ± 0.51	97.10 ± 0.61	94.41 ± 1.05	90.52 ± 0.91	86.52 ± 1.30	1.88
DASH-Net (Ours)	$\mathbf{99.72 \pm 0.09}$	99.66 ± 0.05	$\mathbf{99.75 \pm 0.08}$	99.69 ± 0.10	99.73 ± 0.11	0.51

Table S3: Supervision term, citation graphs.

(a) Cora

# Labels per class	1	2	3	4	5	RunTime (s)
DASH SSL, $\mathbf{B}_{\text{lap}}$ (Alg. 1)	22.30 ± 2.99	22.47 ± 2.99	23.77 ± 3.53	25.18 ± 3.03	26.02 ± 3.76	0.13
DASH SSL, $\mathbf{B}_{\text{poi}}$ (Alg. 1)	29.64 ± 8.90	31.66 ± 9.69	42.70 ± 9.34	43.12 ± 9.02	45.40 ± 9.08	0.15
DASH-Net, without $\mathbf{B}_{\text{poi}}$	58.61 ± 7.68	63.69 ± 8.47	68.14 ± 5.83	70.52 ± 3.52	71.80 ± 2.72	0.06
DASH-Net, with $\mathbf{B}_{\text{poi}}$ (Ours)	$\mathbf{59.58 \pm 9.12}$	$\mathbf{66.18 \pm 6.60}$	$\mathbf{70.48 \pm 3.61}$	$\mathbf{71.99 \pm 2.50}$	$\mathbf{72.99 \pm 2.43}$	0.11

(b) CiteSeer

# Labels per class	1	2	3	4	5	RunTime (s)
DASH SSL, $\mathbf{B}_{\text{lap}}$ (Alg. 1)	21.35 ± 2.99	22.10 ± 3.14	22.52 ± 2.96	20.48 ± 2.09	21.87 ± 2.09	0.29
DASH SSL, $\mathbf{B}_{\text{poi}}$ (Alg. 1)	26.55 ± 8.69	32.11 ± 10.03	31.19 ± 6.48	33.15 ± 5.93	33.79 ± 6.45	0.27
DASH-Net, without $\mathbf{B}_{\text{poi}}$	34.81 ± 8.84	40.61 ± 6.51	44.20 ± 6.42	44.82 ± 5.29	47.46 ± 3.17	0.06
DASH-Net, with $\mathbf{B}_{\text{poi}}$ (Ours)	$\mathbf{35.57 \pm 8.70}$	$\mathbf{42.76 \pm 6.93}$	$\mathbf{45.32 \pm 4.97}$	$\mathbf{45.22 \pm 5.83}$	$\mathbf{47.94 \pm 2.69}$	0.12

(c) PubMed

# Labels per class	1	2	3	4	5	RunTime (s)
DASH SSL, $\mathbf{B}_{\text{lap}}$ (Alg. 1)	39.79 ± 4.36	40.03 ± 5.11	41.11 ± 5.52	40.55 ± 4.16	41.25 ± 5.53	0.60
DASH SSL, $\mathbf{B}_{\text{poi}}$ (Alg. 1)	43.14 ± 7.76	51.16 ± 8.65	53.11 ± 11.03	50.39 ± 9.81	51.80 ± 10.34	0.93
DASH-Net, without $\mathbf{B}_{\text{poi}}$	53.60 ± 15.31	63.36 ± 10.28	63.98 ± 8.84	$\mathbf{66.17 \pm 6.64}$	67.19 ± 5.66	0.09
DASH-Net, with $\mathbf{B}_{\text{poi}}$ (Ours)	$\mathbf{59.53 \pm 10.38}$	$\mathbf{66.50 \pm 7.04}$	$\mathbf{69.01 \pm 3.83}$	65.94 ± 7.91	$\mathbf{69.44 \pm 6.29}$	0.68

subproblem, the total unimodularity argument, or the convergence guarantees. This appendix measures what that choice is worth, holding everything else fixed, namely the same architecture, the same constraint set, the same graph and the same label draws. The two rows of the fixed iteration differ only in $\mathbf{B}_{\text{lap}} = -\mathbf{L}_{u\ell}\mathbf{Y}$ against $\mathbf{B}_{\text{poi}}$ of (3), and the two DASH-Net rows only in whether $\mathbf{B}_{\text{poi}}$ enters the cost alongside the learned diffusion of the labels.

Tables S3 and S4 report the citation and image graphs in turn. For the fixed iteration, the global term raises the accuracy at every rate on all five graphs. With the one-hop term $\mathbf{B}_{\text{lap}}$ the iteration scores 13.65 and 11.38 at one label per class on Fashion-MNIST and SVHN, at or near chance in

Table S4: Supervision term, image graphs.

(a) FashionMNIST-VAE

# Labels per class	1	2	3	4	5	RunTime (s)
DASH SSL, $\mathbf{B}_{\mathrm{lap}}$ (Alg. 1)	13.65 ± 2.33	13.72 ± 2.15	14.13 ± 3.00	15.01 ± 2.87	18.47 ± 2.96	8.67
DASH SSL, $\mathbf{B}_{\mathrm{poi}}$ (Alg. 1)	$\underline{62.32 \pm 4.50}$	$\underline{66.56 \pm 4.73}$	$\underline{70.82 \pm 1.89}$	$\underline{72.54 \pm 3.04}$	$\underline{72.69 \pm 2.03}$	5.18
DASH-Net, without $\mathbf{B}_{\mathrm{poi}}$	61.00 ± 5.36	65.85 ± 4.83	70.42 ± 2.64	71.86 ± 2.67	72.69 ± 1.71	0.23
DASH-Net, with $\mathbf{B}_{\mathrm{poi}}$ (Ours)	$\mathbf{62.85 \pm 4.97}$	$\mathbf{67.47 \pm 5.17}$	$\mathbf{71.82 \pm 2.15}$	$\mathbf{72.95 \pm 3.22}$	$\mathbf{73.41 \pm 2.08}$	3.81

(b) SVHN-SimCLR

# Labels per class	1	2	3	4	5	RunTime (s)
DASH SSL, $\mathbf{B}_{\mathrm{lap}}$ (Alg. 1)	11.38 ± 1.10	12.74 ± 1.36	14.27 ± 1.31	15.03 ± 1.51	15.52 ± 1.11	16.78
DASH SSL, $\mathbf{B}_{\mathrm{poi}}$ (Alg. 1)	25.92 ± 4.83	32.84 ± 8.72	45.84 ± 7.13	54.24 ± 8.77	58.76 ± 7.92	14.88
DASH-Net, without $\mathbf{B}_{\mathrm{poi}}$	$\underline{42.77 \pm 4.94}$	$\mathbf{59.32 \pm 6.96}$	$\mathbf{67.94 \pm 6.13}$	$\mathbf{74.29 \pm 4.65}$	$\mathbf{77.83 \pm 3.25}$	0.33
DASH-Net, with $\mathbf{B}_{\mathrm{poi}}$ (Ours)	$\mathbf{44.21 \pm 4.56}$	$\underline{58.28 \pm 6.95}$	$\underline{63.88 \pm 6.72}$	$\underline{65.10 \pm 5.29}$	$\underline{67.48 \pm 6.03}$	7.47

both cases, and with $\mathbf{B}_{\mathrm{poi}}$ the same solver, on the same graphs, under the same constraints and with no learning of any kind, scores 62.32 and 25.92, the first on a par with the strongest baselines of Table S2. On the citation graphs the gain reaches 19.38 points, on Cora at five labels per class. The unrolled schedule then lifts the fixed iteration further on every graph, and most on the citation graphs and on SVHN, where the one-hop operator of Algorithm 1 has the farthest to carry the labels, so that the two components are complementary.

For the unrolled network, adding $\mathbf{B}_{\mathrm{poi}}$ raises the accuracy at every rate on Cora, from $+0.97$ to $+2.49$, on CiteSeer, from $+0.40$ to $+2.15$, and on Fashion-MNIST, from $+0.72$ to $+1.85$, and on PubMed at every rate except four labels per class, where the two variants are on a par. On Fashion-MNIST the advantage decays with the label rate, from $+1.85$ to $+0.72$, which is what the argument of Section 2 predicts, since supervision spread locally from ℓ is nearly empty when ℓ is tiny and progressively more informative as ℓ grows. On the citation graphs no such decay is visible over this range, which is consistent with ℓ remaining negligible relative to n at all five rates.

On SVHN the global term leads at one label per class, by $+1.44$, and from two labels per class upward the network driven by the learned diffusion of the labels alone is the stronger of the two, reaching 77.83 at five labels per class, above every baseline of Table S2 at those rates. The framework takes either term without modification, since the choice of $\mathbf{B}$ is orthogonal to the solver, and every comparison with the baselines in this paper uses a single configuration, the one with $\mathbf{B}_{\mathrm{poi}}$, which is the stronger at the sparsest rate.