LEARNING LINEARIZED ASSIGNMENT FLOWS FOR IMAGE LABELING

ALEXANDER ZEILMANN, STEFANIA PETRA, CHRISTOPH SCHNÖRR

ABSTRACT. We introduce a novel algorithm for estimating optimal parameters of linearized assignment flows for image labeling. An exact formula is derived for the parameter gradient of any loss function that is constrained by the linear system of ODEs determining the linearized assignment flow. We show how to efficiently evaluate this formula using a Krylov subspace and a low-rank approximation. This enables us to perform parameter learning by Riemannian gradient descent in the parameter space, without the need to backpropagate errors or to solve an adjoint equation, in less than 10 seconds for a 512×512 image using just about 0.5 GB memory. Experiments demonstrate that our method performs as good as highly-tuned machine learning software using automatic differentiation. Unlike methods employing automatic differentiation, our approach yields a low-dimensional representation of internal parameters and their dynamics which helps to understand how networks work and perform that realize assignment flows and generalizations thereof.

CONTENTS

Key words and phrases. assignment flows, image labeling, parameter learning, exponential integration, low-rank approximation.

1. Introduction

1.1. **Overview, Motivation.** Learning the parameters of large neural networks from training data constitutes a basic problem in imaging science, machine learning and other fields. The prevailing approach utilizes gradient descent or approximations thereof based on automatic differentiation [BPRS18] and corresponding software tools, like PyTorch [PGM+19] and TensorFlow[AAB+16]. This kind of software support has been spurring research in imaging science and machine learning dramatically. However, merely relying on numerical schemes and their automatic differentiation tends to thwart attempts to shed light on the often-criticized black-box behavior of deep networks and to better understand the internal representation and function of parameters and their adaptive dynamics.

In this paper, we explore a different route. Adopting the *linearized assignment flow* approach introduced by [ZSPS20], we focus on a corresponding large system of linear ODEs of the form

$$\dot{V} = A(\Omega)V + B, \tag{1.1}$$

and study a geometric approach to learning the regularization parameters Ω by Riemannian gradient descent of a loss function

$$\Omega \mapsto \mathcal{L}(V(T;\Omega)) \tag{1.2}$$

constrained by the dynamical system (1.1). Here, the matrix $V \in \mathbb{R}^{|I|\times c}$ represents a tangent vector of the so-called assignment manifold, $|I|$ is the number of nodes $i \in I$ of the underlying graph, and c is the number of labels (classes) that have to be assigned to data observed at nodes $i \in I$. Specifically,

- we derive a formula for the Euclidean parameter gradient $\partial_\Omega \mathcal{L}(V(T;\Omega))$ in closed form;
- we show that a low-rank representation of this gradient can be used to efficiently and accurately approximate evaluate this closed form gradient; neither backpropagation, nor automatic differentiation or solving adjoint equations are required;
- we highlight that the resulting parameter estimation algorithm can be implemented without any specialized software support with modest computational resources;
- we demonstrate that optimal parameters for a 512×512 image can be learned in less than 10 seconds with a memory requirement that does not exceed ≈ 0.5 GB.

The significance of our work reported in this paper arises in a broader context. The linearized assignment flow approach also comprises the equation

$$W(T) = \operatorname{Exp}_{\mathbb{1}_\mathcal{W}}(V(T)) \tag{1.3}$$

that yields the labeling in terms of almost integral assignment vectors $W_i \in \mathbb{R}^c_+,\ i \in I$ that form the rows of the matrix W, depending on the solution $V(t)$ of (1.1) for a sufficiently large time $t = T$. Both equations (1.3) and (1.1) together constitute a linearization of the full nonlinear assignment flow [ÅPSS17]

$$\dot{W} = R_W S(W) \tag{1.4}$$

at the barycenter $\mathbb{1}_\mathcal{W}$ of the assignment manifold. Choosing an arbitrary sequence of time intervals (step sizes) $h_1, h_2, \ldots$ and setting

$$W^{(0)} = \mathbb{1}_\mathcal{W}, \qquad B = \Pi_0 S(W^{(0)}), \qquad W^{(k)} = W(h_k), \quad k \in \mathbb{N}, \tag{1.5}$$

a *sequence* of linearized assignment flows

$$W^{(k+1)} = \operatorname{Exp}_{\mathbb{1}_\mathcal{W}}(V^{(k)}), \tag{1.6a}$$

$$V^{(k+1)} = V^{(k)} + V\big(h_k; \Omega^{(k)}, W^{(k)}\big), \quad k = 0, 1, 2, \ldots \tag{1.6b}$$

can be computed in order to approximate (1.4) more closely, where $V\big(h_k; \Omega, W^{(k)}\big)$ solves the corresponding updated ODE (1.1) of the form

$$\dot{V} = A(\Omega^{(k)}; W^{(k)})V + \Pi_0 S(W^{(k)}). \tag{1.6c}$$

The time-discrete equations (1.6) reveal two basic ingredients of deep networks (or neural ODEs) which the full assignment flow (1.4) embodies in a continuous-time manner: coupling a pointwise nonlinearity (1.6a) and diffusion (1.6b),(1.6c) enhances the expressivity of network models for data analysis.

The key point motivating the work reported in this paper is that our results apply to learning the parameters Ω^k in each step of the iterative scheme (1.6). We expect that the gradient, and its low-dimensional subspace representations, will help the further study of how each ingredient of (1.6) impacts the predictive power of assignment flows. Furthermore, 'deep' extensions of (1.4) and (1.6) are equally feasible within the *same* mathematical framework (cf. Section 5).

1.2. **Related Work.** *Assignment flows* were introduced by [ÅPSS17]. For a survey of prior and recent related work, we refer to [Sch20]. *Linearized* assignment flows were introduced by [ZSPS20] as part of a comprehensive study of numerical schemes for the geometric integration of the assignment flow equation (1.4).

While the bulk of these schemes rely on established theory and algorithms for the numerical integration of ODEs that evolve in a Euclidean space [HNW08], due to the observation that addition in the tangent space in connection with the exponential map constitutes a Lie group action on the assignment manifold (cf. [IMKNZ00]), a representation of the solution to the linear ODE (1.1) in closed form is given by the Duhamel (or variation-of-constants) formula [Tes12]. Corresponding extensions to *nonlinear* ODEs rely on exponential integration [HOS09, HO10]. The scheme (1.6) combines a corresponding iterative scheme and the tangent-space based parametrization (1.3) of the linearized assignment flow.

A key computational step of the latter class of methods requires to evaluate an analytical matrix-valued function, like the matrix exponential and similar functions [Hig08, Section 10]. While basic methods [MVL03] only work for problem of small and medium size, dedicated methods using Krylov subspaces [HL97, AMH11] and established numerical linear algebra [Saa92, Saa03] can be applied to larger problems. The algorithm that results from our approach employs such methods.

Machine learning requires to compute gradients of loss functions that take solutions of ODEs as argument. This defines an enormous computational task and explains why automatic differentiation and corresponding software tools are almost exclusively applied. Alternative dedicated recent methods like [KKRS21] have to focus on a special problem structure, viz. the action of the differential of the matrix exponential on a rank-one matrix. Our closed-form formula for the parameter gradient also involves the differential of a matrix exponential. Yet, we wish to evaluate the gradient itself rather than its action on another matrix. The special problem structure that we can exploit is the Kronecker sum of matrices. Accordingly, our approach is based on the recent corresponding work [BS17] and an additional subsequent low-rank approximation.

1.3. **Contribution, Organization.** We derive a closed-form expression of the gradient of any C^1 loss function of the form (1.2) that depends on the solution $V(t)$ of the linear system of ODEs (1.1) at some arbitrary but fixed time $t = T$. In addition, we develop a numerical method that enables to evaluate the gradient efficiently for the common large sizes of image labeling problems. We apply the method to optimal parameter estimation by Riemannian gradient descent and validate our approach by a series of proof-of-concept experiments. This includes a comparison with automatic differentiation applied to two numerical schemes for integrating the linearizes assignment flow: geometric explicit Euler and exponential integration. It turns out that our method is as accurate and efficient as the highly optimized automatic differentiation software. We point out that to our knowledge, automatic differentiation has not been applied to exponential integration, so far.

This paper elaborates the conference version [ZPS21] in that all parameter dependencies of the loss function, constrained by the linearized assignment flow, are taken into account (cf. diagram (3.16)), and a complete proof of the corresponding main result (Theorem 3.8) is provided. The space complexity of various gradient approximations are specified in a series of Remarks. The approach is validated numerically and more comprehensively by comparing to automatic differentiation and by examining the influence of all parameters.

The plan for this paper is as follows. Section 2 summarizes the assignment flow approach, the linearized assignment flow and exponential integration for integrating the latter flow. Section 3 details the derivation of the exact gradient of any loss function of the flow with respect to the weight parameters that regularize the flow. Furthermore, a low-rank approximation of the gradient is developed for evaluating the gradient efficiently. We also sketch how automatic derivation is applied to two numerical schemes in order to solve the parameter estimation problem in alternative ways. Numerical experiments are reported in Section 4 for comparing the methods and for inspecting quantitatively the gradient approximation and properties of the estimated parameter patches. We conclude in Section 5 and point out further directions of research.

2. Preliminaries

2.1. **Basic Notation.** We set $[n] = \{1, 2, \ldots, n\}$ for $n \in \mathbb{N}$. The cardinality of a finite set S is denoted by S, e.g. $|[n]| = n$. $\mathbb{R}^n_+$ denotes the positive orthant and $\mathbb{R}^n_>$ its interior. $\mathbb{1} = (1, 1, \ldots, 1)^\top$ has dimension depending on the context that we specify sometimes by a subscript, e.g. $\mathbb{1}_n \in \mathbb{R}^n$. Similarly, we set $0_n = (0, 0, \ldots, 0)^\top \in \mathbb{R}^n$. $\{e_i \colon i \in [n]\}$ is the canonical basis of $\mathbb{R}^n$ and $I_n = (e_1, \ldots, e_n) \in \mathbb{R}^{n\times n}$ the identity matrix.

The support of a vector $x \in \mathbb{R}^n$ is denoted by $\operatorname{supp}(x) = \{i \in [n] \colon x_i \neq 0\}$. $\Delta_n = \{p \in \mathbb{R}^n_+ \colon \langle \mathbb{1}_n, p\rangle = 1\}$ is the probability simplex whose points represent discrete distributions on $[n]$. Distributions with full support $[n]$ form the relative interior $\mathring{\Delta}_n = \Delta_n \cap \mathbb{R}^n_>$. $\langle \cdot, \cdot \rangle$ is the Euclidean inner product of vectors and matrices. In the latter case, this reads $\langle A, B\rangle = \operatorname{tr}(A^\top B)$ with the trace $\operatorname{tr}(A) = \sum_i A_{ii}$. The induced Frobenius norm is denoted by $\|A\| = \sqrt{\langle A, A, \rangle}$, and other matrix norms like the spectral norm $\|A\|_2$ are indicated by subscripts. The mapping $\operatorname{Diag} \colon \mathbb{R}^n \to \mathbb{R}^{n\times n}$ sends a vector x to the diagonal matrix $\operatorname{Diag}(x)$ with entries x. $A \otimes B$ denotes the Kronecker *product* of matrices A and B [VL00] and $\oplus$ the Kronecker *sum*

$$A \oplus B = A \oplus I_n + I_m \oplus B \in \mathbb{R}^{mn\times mn}, \qquad A \in \mathbb{R}^{m\times m}, \quad B \in \mathbb{R}^{n\times n}. \tag{2.1}$$

We have

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD) \tag{2.2}$$

for matrices of compatible dimensions. The operator vec_r turns a matrix into the vector by stacking the row vectors. It satisfies

$$\operatorname{vec}_r(ABC) = (A \otimes C^\top) \operatorname{vec}_r(B). \tag{2.3}$$

The Kronecker product $v \otimes w \in \mathbb{R}^{mn}$ of two vectors $v \in \mathbb{R}^m$ and $w \in \mathbb{R}^n$ is defined by viewing the vectors as matrices with only one column and applying the definition of Kronecker products for matrices. We have

$$v \otimes w = \operatorname{vec}_r(vw^\top). \tag{2.4}$$

The matrix exponential of a square matrix A is given by [Hig08, Ch. 10]

$$\operatorname{expm}(A) = \sum_{k\geq 0} \frac{A^k}{k!}. \tag{2.5}$$

$L(\mathcal{E}_1, \mathcal{E}_2)$ denotes the space of all linear bounded mappings from $\mathcal{E}_1$ to $\mathcal{E}_2$.

2.2. **Assignment Flow.** Let $G = (I, E)$ be a given undirected graph with vertices $i \in I$ indexing data

$$\mathcal{F}_I = \{f_i \colon i \in I\} \subset \mathcal{F} \tag{2.6}$$

given in a metric space $(\mathcal{F}, d)$. The edge set E specifies neighborhoods

$$\mathcal{N}_i = \{k \in I \colon ik = ki \in E\} \cup \{i\}, \quad i \in I. \tag{2.7}$$

Along with $\mathcal{F}_I$, *prototypical data (labels)* $\mathcal{L}_J = \{l_j \in \mathcal{F} \colon j \in J\}$ are given that represent classes $j = 1, \ldots, |J|$. *Supervised image labeling* denotes the task to assign precisely one prototype l_j to each datum f_i at every vertex i in a coherent way, depending on the label assignments in the neighborhoods $\mathcal{N}_i$. These assignments at i are represented by probability vectors

$$W_i \in \mathcal{S} := (\mathring{\Delta}_{|J|}, g_{FR}), \quad i \in I \tag{2.8}$$

on $\mathring{\Delta}_{|J|}$ that becomes a Riemannian manifold denoted by $\mathcal{S}$ when endowed with the Fisher-Rao metric g_{FR}. Collecting all assignment vectors as *rows* defines the strictly positive row-stochastic *assignment matrix*

$$W = (W_1, \ldots, W_{|I|})^\top \in \mathcal{W} = \mathcal{S} \times \cdots \times \mathcal{S} \subset \mathbb{R}^{|I| \times |J|}, \tag{2.9}$$

that we regard as point on the product *assignment manifold* $\mathcal{W}$. Image labeling is accomplished by geometrically integrating the *assignment flow* $W(t)$ solving

$$\dot{W} = R_W\big(S(W)\big), \qquad W(0) = \mathbb{1}_{\mathcal{W}} := \frac{1}{|J|} \mathbb{1}_{|I|} \mathbb{1}_{|J|}^\top \qquad \text{(barycenter)}, \tag{2.10}$$

that provably converges towards a binary matrix [ZZS20], i.e., $\lim_{t \to \infty} W_i(t) = e_{j(i)}$, for every $i \in I$ and some $j(i) \in J$, which yields the label assignment $l_{j(i)} \to f_i$. In practice, geometric integration is terminated when $W(t)$ is ε-close to an integral point using the entropy criterion from [ÅPSS17], followed by trivial rounding [ZZS20].

We specify the right-hand side of (2.10) — see (2.15) and (2.18) below — and refer to [ÅPSS17, Sch20] for more details and the background. With the tangent space

$$T_0 = T_p \mathcal{S} = \{v \in \mathbb{R}^{|J|} \colon \langle \mathbb{1}, v \rangle = 0\}, \qquad \forall p \in \mathcal{S}, \tag{2.11}$$

that does not depend on the base point $p \in \mathcal{S}$, we define

$$\Pi_0 \colon \mathbb{R}^{|J|} \to T_0, \qquad z \mapsto I_{|J|} - \frac{1}{|J|} \mathbb{1}_{|J|} \mathbb{1}_{|J|}^\top, \tag{2.12a}$$

$$R_p \colon \mathbb{R}^{|J|} \to T_0, \qquad z \mapsto R_p(z) = \big(\operatorname{Diag}(p) - pp^\top\big) z, \tag{2.12b}$$

$$\operatorname{Exp} \colon \mathcal{S} \times T_0 \to \mathcal{S}, \quad (p, v) \mapsto \operatorname{Exp}_p(v) = \frac{e^{\frac{v}{p}}}{\langle p, e^{\frac{v}{p}} \rangle} p, \tag{2.12c}$$

$$\operatorname{Exp}^{-1} \colon \mathcal{S} \times \mathcal{S} \to T_0, \quad (p, q) \mapsto \operatorname{Exp}_p^{-1}(q) = R_p \log \frac{q}{p}, \tag{2.12d}$$

$$\exp \colon \mathcal{S} \times \mathbb{R}^{|J|} \to \mathcal{S}, \quad (p, z) \mapsto \exp_p(z) = \operatorname{Exp}_p \circ R_p(z) = \frac{p e^z}{\langle p, e^z \rangle}, \tag{2.12e}$$

where multiplication, division, exponentiation $e^{(\cdot)}$ and $\log(\cdot)$ apply *component-wise* to the vectors. Corresponding maps

$$R_W, \qquad \operatorname{Exp}_W, \qquad \exp_W \tag{2.13}$$

in connection with the product manifold (2.9) are defined analogously, and likewise the tangent space

$$\mathcal{T}_0 = T_0 \times \cdots \times T_0 = T_W \mathcal{W}, \qquad \forall W \in \mathcal{W} \tag{2.14}$$

and the extension of the orthogonal projection (2.12a) onto $\mathcal{T}_0$, again denoted by Π_0.

For example, regarding (2.10), with $W \in \mathcal{W}$ and $S(W) \in \mathcal{W}$ (or more generally $S \in \mathbb{R}^{|I|\times|J|}$), we have

$$R_W S(W) = \big(R_{W_1} S_1(W), \ldots, R_{W_{|I|}} S_{|I|}(W)\big)^\top = \mathrm{vec}_r^{-1}\left(\mathrm{Diag}(R_W)\,\mathrm{vec}_r\left(S(W)\right)\right) \tag{2.15a}$$

with

$$\mathrm{Diag}(R_W) := \begin{pmatrix} R_{W_1} & 0 & \cdots & 0 \\ 0 & R_{W_2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & & R_{W_{|I|}} \end{pmatrix}. \tag{2.15b}$$

Given data $\mathcal{F}_I$ are taken into account as distance vectors

$$D_i = \big(d(f_i, l_1), \ldots, d(f_i, l_{|J|})\big)^\top, \quad i \in I \tag{2.16}$$

and mapped to $\mathcal{W}$ by

$$L(W) = \exp_W(-\tfrac{1}{\rho} D) \in \mathcal{W}, \quad L_i(W_i) = \exp_{W_i}(-\tfrac{1}{\rho} D_i) = \frac{W_i e^{-\frac{1}{\rho} D_i}}{\langle W_i, e^{-\frac{1}{\rho} D_i}\rangle}, \tag{2.17}$$

where $\rho > 0$ is a user parameter for normalizing the scale of the data. These *likelihood vectors* represent data terms in conventional variational approaches: Each individual flow $\dot{W}_i = R_{W_i} L_i(W_i)$, $W_i(0) = \mathbb{1}_{\mathcal{S}}$ converges to $e_{j(i)}$ with $j(i) = \arg\min_{j\in J} D_{ij}$ and in this sense maximizes the local data likelihoods.

The vector field defining the assignment flow (2.10) arises through *coupling* flows for individual pixels through *geometric averaging* within the neighborhoods $\mathcal{N}_i$, $i \in I$, conforming to the underlying Fisher-Rao geometry

$$S(W) = \begin{pmatrix} \vdots \\ S_i(W)^\top \\ \vdots \end{pmatrix} = \mathcal{G}^\Omega\big(L(W)\big) \in \mathcal{W}, \tag{2.18a}$$

$$S_i(W) = \mathcal{G}_i^\Omega\big(L(W)\big) = \mathrm{Exp}_{W_i}\Big(\sum_{k\in\mathcal{N}_i} \omega_{ik} \mathrm{Exp}_{W_i}^{-1}\left(L_k(W_k)\right)\Big), \quad i \in I. \tag{2.18b}$$

The *similarity vectors* $S_i(W)$ are parametrized by strictly positive *weight patches* $(\omega_{ik})_{k\in\mathcal{N}_i}$, centered at $i \in I$ and indexed by local neighborhoods $\mathcal{N}_i \subset I$, that in turn define the *weight parameter matrix*

$$\Omega = (\Omega_i)_{i\in I} \in \mathbb{R}_+^{|I|\times|I|}, \qquad \Omega_i|_{\mathcal{N}_i} = (\omega_{ik})_{k\in\mathcal{N}_i} \in \mathring{\Delta}_{|\mathcal{N}_i|}, \qquad \sum_{k\in\mathcal{N}_i} \omega_{ik} = 1,\ \forall i \in I \tag{2.19}$$

that comprises all *regularization parameters* satisfying the latter linear constraints. Flattening these weight patches defines row vectors $\Omega_i|_{\mathcal{N}_i}$, $i \in I$ and, by complementing with 0, entries of the *sparse* row vectors Ω_i of the matrix Ω. Note that the positivity assumption $\omega_{ik} > 0$ is reflected by the membership $\Omega_i|_{\mathcal{N}_i} \in \mathring{\Delta}_{|\mathcal{N}_i|}$. Throughout this paper, we assume equal neighborhoods

$$|\mathcal{N}| := |\mathcal{N}_i|, \quad \forall i \in I \tag{2.20}$$

and therefore simply write $\Omega_i|_{\mathcal{N}} = \Omega_i|_{\mathcal{N}_i}$.

Estimating these parameters from data using the linearized assignment flow, to be introduced next, is the subject of this paper.

2.3. **Linearized Assignment Flow.** The *linearized assignment flow*, introduced by [ZSPS20], approximates (2.10) by

$$\dot{W} = R_W\Big(S(W_0) + dS_{W_0} R_{W_0} \log \frac{W}{W_0}\Big), \quad W(0) = W_0 \in \mathcal{W} \tag{2.21}$$

around any point W_0. In what follows, we only consider the barycenter

$$W_0 = \mathbb{1}_{\mathcal{W}} \tag{2.22}$$

which is the initial point of (2.10). The differential equation (2.21) is still *nonlinear* but can be parametrized by a *linear* ODE on the tangent space

$$W(t) = \operatorname{Exp}_{W_0}\big(V(t)\big), \tag{2.23a}$$

$$\dot{V} = R_{W_0}\big(S(W_0) + dS_{W_0} V\big) =: B_{W_0} + A(\Omega)V, \quad V(0) = 0, \tag{2.23b}$$

where matrix $A(\Omega)$ linearly depends on the parameters Ω of (2.18) and whose action on V is explicitly given by [ZSPS20, Prop. 4.4]

$$A(\Omega)V = R_{W_0} dS_{W_0} V = R_{S(W_0)} \Omega V \overset{(2.15)}{=} \operatorname{vec}_r^{-1}\big(\operatorname{Diag}(R_{S(W_0)})\operatorname{vec}_r(\Omega V)\big) \tag{2.24a}$$

$$= \begin{pmatrix} R_{S_1(W_0)} \sum_{k\in\mathcal{N}_1} \omega_{1k} V_k, \ldots, R_{S_{|I|}(W_0)} \sum_{k\in\mathcal{N}_{|I|}} \omega_{|I|k} V_k \end{pmatrix}^\top, \tag{2.24b}$$

where $\operatorname{Diag}(R_{S(W_0)})$ is defined by (2.15b) and we took into account (2.22). The linear ODE (2.23b) admits a closed-form solution which in turn enables a different numerical approach (Section 2.4) and a novel approach to parameter learning (Section 3).

2.4. **Exponential Integration.** The solution to (2.23b) is given by a high-dimensional integral (Duhamel's formula) whose value in closed form is given by

$$V(t;\Omega) = t\varphi\big(tA(\Omega)\big)B_{W_0}, \qquad \varphi(x) = \frac{e^x - 1}{x} = \sum_{k=0}^{\infty} \frac{x^k}{(k+1)!}, \tag{2.25}$$

where the entire function φ is extended to matrix arguments in the standard way [Hig08]. As the matrix A is already very large even for medium-sized images, however, it is not feasible in practice to compute $\varphi(tA)$. Exponential integration [HL97, NW12], therefore, was used by [ZSPS20] for approximately evaluating (2.25), as sketched next.

Applying the row-stacking operator (2.3) to both sides of (2.23b) and (2.25), respectively, yields with

$$v = \operatorname{vec}_r(V) \tag{2.26}$$

the ODE (2.23b) in the form

$$\dot{v} = b + A^J(\Omega)v, \qquad v(0) = 0, \qquad b = b(\Omega) = \operatorname{vec}_r(B_{W_0}) \in \mathbb{R}^n, \tag{2.27a}$$

$$A^J(\Omega) = \big(A^J_{ik}(\Omega)\big)_{i,k\in I} \in \mathbb{R}^{n\times n}, \qquad A^J_{ik}(\Omega) = \begin{cases} \omega_{ik} R_{S_i(W_0)}, & k \in \mathcal{N}_i, \\ 0, & k \notin \mathcal{N}_i. \end{cases} \tag{2.27b}$$

$$v(t;\Omega) = t\varphi\big(tA^J(\Omega)\big)b, \qquad n := \dim v(t;\Omega) = |I||J|, \tag{2.27c}$$

where $A^J(\Omega)$ results from

$$\operatorname{vec}_r\big(A(\Omega)V\big) \overset{(2.24)}{=} \operatorname{Diag}(R_{S(W_0)})\operatorname{vec}_r(\Omega V) = \operatorname{Diag}(R_{S(W_0)})(\Omega \otimes I_{|J|})v \tag{2.28a}$$

$$= A^J(\Omega)v. \tag{2.28b}$$

Using the Arnoldi iteration [Saa03] with initial vector $q_1 = b/\|b\|$, an orthonormal basis $Q_m = (q_1, \dots, q_m) \in \mathbb{R}^{n \times m}$ of the Krylov space $\mathcal{K}_m(A^J, b)$ of dimension m is determined. As will be validated in Section 4, choosing $m \leq 10$ yields sufficiently accurate approximations of the actions of the matrix exponential expm and the φ operator on a vector, respectively, that are given by

$$\operatorname{expm}\big(tA^J(\Omega)\big)b \approx \|b\| Q_m \operatorname{expm}(tH_m)e_1, \qquad H_m = Q_m^\top A^J(\Omega) Q_m, \tag{2.29a}$$

$$t\varphi\big(tA^J(\Omega)\big)b \approx t\|b\| Q_m \varphi(tH_m)e_1. \tag{2.29b}$$

The expression $\varphi(tH_m)e_1$ results from computing the left-hand side of the relation [Hig08, Section 10.7.4]

$$\operatorname{expm}\begin{pmatrix} tH_m & e_1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} \operatorname{expm}(tH_m) & \varphi(tH_m)e_1 \\ 0 & 1 \end{pmatrix} \tag{2.30}$$

and extracting the upper-right vector. Since H_m is a small matrix, any standard method [MVL03] can be used for computing the matrix exponential on the left-hand side.

3. Parameter Estimation

Section 3.1 details our approach for learning optimal weight parameters for a given image and ground truth labeling: Riemannian gradient descent is performed with respect to a loss function that depends on the solution of the linearized assignment flow. A closed form expression of this gradient is derived in Section 3.2 along with a low-rank approximation in Section 3.3 that can be computed efficiently. As an alternative and baseline, we outline in Section 3.4 two gradient approximations based on numerical schemes for integrating the linearized assignment flow and automatic differentiation.

3.1. **Learning Procedure.** Let

$$P_\Omega = \{\Omega \in \mathbb{R}_+^{|I| \times |I|} \colon \Omega \text{ satisfies (2.19)}\} \tag{3.1}$$

denote the space of weight parameter matrices that parametrize the similarity mapping (2.18). Due to (2.19) and (2.20), the restrictions $\Omega_i|_{\mathcal{N}}$ are strictly positive probability vectors, as are the assignment vectors W_i defined by (2.8). Therefore, similar to $W_i \in \mathcal{S}$, we consider each $\Omega_i|_{\mathcal{N}}$ as point on a corresponding manifold $(\Delta_{|\mathcal{N}|}, g_{FR})$ equipped with the Fisher-Rao metric and — in this sense — regard P_Ω in (3.1) as corresponding product manifold.

Let $W^* \in \mathcal{W}$ denote the ground truth labeling for a given image, and let $V^* \in \mathcal{T}_0$ be a tangent vector such that

$$\lim_{s \to \infty} \operatorname{Exp}_{\mathbb{1}_\mathcal{W}}(sV^*) = W^*. \tag{3.2}$$

Our objective is to determine Ω such that, for some specified time $T > 0$, the vector

$$V_T(\Omega) := V(T; \Omega), \tag{3.3}$$

given by (2.25) and corresponding to the linearized assignment flow, approximates the *direction* of V^* and hence

$$\lim_{s \to \infty} \operatorname{Exp}_{\mathbb{1}_\mathcal{W}}\big(sV_T(\Omega)\big) = W^*. \tag{3.4}$$

A suitable distance function that is sensitive to the direction of these vectors, but not to their size, is given by

$$f_\mathcal{L} \colon \mathcal{T}_0 \to \mathbb{R}, \qquad V \mapsto 1 - \frac{\langle V^*, V \rangle}{\|V^*\| \|V\|}. \tag{3.5}$$

In addition, we consider a regularizer

$$\mathcal{R} \colon P_\Omega \to \mathbb{R}, \qquad \Omega \mapsto \frac{\tau}{2} \sum_{i \in I} \|t_i(\Omega)\|^2, \qquad t_i(\Omega) = \exp_{\mathbb{1}_\Omega}^{-1}(\Omega_i|_\mathcal{N}), \qquad \tau > 0 \tag{3.6}$$

and define the loss function

$$\mathcal{L}\colon P_\Omega \to \mathbb{R}, \qquad \mathcal{L}(\Omega) = f_{\mathcal{L}}\big(V_T(\Omega)\big) + \mathcal{R}(\Omega), \tag{3.7}$$

with $V_T(\Omega)$ from (3.3). Ω is determined by the Riemannian gradient descent sequence

$$\Omega^{(k+1)} = \exp_{\Omega^{(k)}}\big(-h\nabla\mathcal{L}(\Omega^{(k)})\big), \quad k \geq 0, \qquad \Omega_i^{(0)}|_{\mathcal{N}} = \mathbb{1}_{|\mathcal{N}|}, \quad i \in I \tag{3.8}$$

with step size $h > 0$. Here

$$\nabla\mathcal{L}(\Omega) = R_\Omega \partial\mathcal{L}(\Omega) \tag{3.9}$$

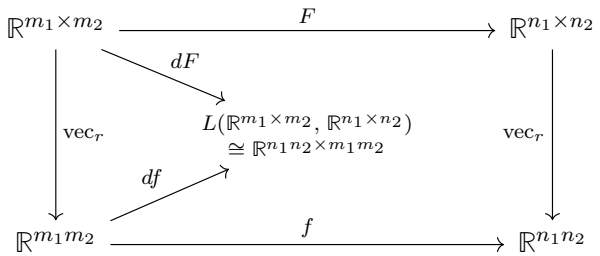
is the Riemannian gradient with respect to the Fisher-Rao metric. R_Ω is given by (2.13) and (2.12b) and effectively applies to the restrictions $\Omega_i|_{\mathcal{N}}$ of the row vectors with all remaining components equal to 0. It remains to compute the Euclidean gradient $\partial\mathcal{L}(\Omega)$ of the loss function 3.7 which is presented in the subsequent Section 3.2.

3.2. **Loss Function Gradient.** We derive in Section 3.2.2 a closed form expression for the loss function gradient (Theorem 3.8), after introducing some basic calculus rules for representing and computing differentials of matrix-valued mappings in Section 3.2.1.

3.2.1. *Matrix Differentials.* Let $F\colon \mathbb{R}^{m_1\times m_2} \to \mathbb{R}^{n_1\times n_2}$ be a smooth mapping. Using the canonical identification $T\mathcal{E} \cong \mathcal{E}$ of the tangent spaces of any Euclidean space $\mathcal{E}$ with $\mathcal{E}$ itself, we both represent and compute the differential

$$dF\colon \mathbb{R}^{m_1\times m_2} \to L(\mathbb{R}^{m_1\times m_2}, \mathbb{R}^{n_1\times n_2}) \tag{3.10}$$

in terms of a vector-valued mapping f, which is defined by F according to the diagram



(3.11)

Specifically, based on the equation

$$\mathrm{vec}_r\big(F(X)\big) = f\big(\mathrm{vec}_r(X)\big), \quad \forall X \in \mathbb{R}^{m_1\times m_2}, \tag{3.12}$$

we set

$$\mathrm{vec}_r\big(dF(X)Y\big) = df\big(\mathrm{vec}_r(X)\big)\,\mathrm{vec}_r(Y), \qquad \forall X, Y \in \mathbb{R}^{m_1\times m_2} \tag{3.13}$$

and hence *define* and compute the differential (3.10) as matrix-valued mapping

$$dF := df \circ \mathrm{vec}_r\,. \tag{3.14}$$

The corresponding linear actions on $Y \in \mathbb{R}^{m_1\times m_2}$ and $\mathrm{vec}_r(Y) \in \mathbb{R}^{m_1m_2}$, respectively, are given by (3.13). We state an auxiliary result required in the next subsection, which also provides a first concrete instance of the general relation (3.13).

Lemma 3.1 (differential of the matrix exponential)**.** *If* $F = \mathrm{expm}\colon \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$, *then* (3.13) *reads*

$$\mathrm{vec}_r\big(d\,\mathrm{expm}(X)Y\big) = \big(\mathrm{expm}(X)\otimes I_n\big)\varphi(-X\oplus X^\top)\,\mathrm{vec}_r(Y), \quad Y \in \mathbb{R}^{n\times n}, \tag{3.15}$$

with φ *given by* (2.25)*.*

Proof. The result follows from [Hig08, Thm. 10.13] where columnwise vectorization is used, after rearranging so as to conform to the row-stacking mapping vec_r used in this paper. □

3.2.2. *Closed-Form Gradient Expression.* We separate the computation of $\mathcal{L}(\Omega)$ and the gradient $\partial\mathcal{L}(\Omega)$ into several operations as illustrated by the following diagram that refers to quantities in (2.27) and (2.28) related to the linearized assignment flow, after vectorization.

$$\begin{array}{ccccc} \Omega & \xrightarrow{(1)} & S(W_0) = \exp_{\mathbb{1}_{\mathcal{W}}}\left(-\frac{1}{\rho}\Omega D\right) & \xrightarrow{(2)} & b(\Omega) = \mathrm{vec}_r(R_{W_0}S(W_0)) \\ & \searrow^{(3)} & \downarrow & & \downarrow{\scriptstyle (4)} \\ & & A^J(\Omega) = \mathrm{Diag}(R_{S(W_0)})(\Omega \otimes I_{|J|}) & \xrightarrow{(4)} & v_T(\Omega) = T\varphi\left(TA^J(\Omega)\right) b(\Omega) \\ & \overset{(5)}{\searrow} & & & \downarrow \\ & & \mathcal{R}(\Omega) & \longrightarrow & \mathcal{L}(\Omega) = f_{\mathcal{L}}(v_T(\Omega)) + \mathcal{R}(\Omega) \end{array} \tag{3.16}$$

In what follows, we traverse this diagram from top-left to bottom-right and collect each partial result by a corresponding lemma or proposition. Theorem 3.8 assembles all results and provides a closed form expression of the loss function gradient $\partial\mathcal{L}(\Omega)$. To enhance readability, the proofs of most Lemmata are listed in Appendix A.1.

We focus on mapping (1) in diagram (3.16).

Lemma 3.2. *The differential of the function*

$$f_1 \colon \mathbb{R}^{|I|\times|I|} \to \mathbb{R}^{|I|\times|J|}, \qquad \Omega \mapsto f_1(\Omega) := S(W_0) = \exp_{\mathbb{1}_{\mathcal{W}}}\Big(-\frac{1}{\rho}\Omega D\Big), \qquad D \in \mathbb{R}^{|I|\times|J|} \tag{3.17}$$

and its transpose are given by

$$df_1(\Omega)Y = -\frac{1}{\rho}R_{f_1(\Omega)}(YD), \qquad \forall Y \in \mathbb{R}^{|I|\times|I|}, \tag{3.18a}$$

$$df_1(\Omega)^\top Z = -\frac{1}{\rho}R_{f_1(\Omega)}(Z)D^\top, \qquad \forall Z \in \mathbb{R}^{|I|\times|J|}, \tag{3.18b}$$

with $R_{f_1(\Omega)}$ defined by (2.15).

Proof: see Appendix A.1.

We consider mapping (2) of diagram (3.16), taking into account mapping (1) and notation (3.17).

Lemma 3.3. *The differential of the function*

$$f_2 \colon \mathbb{R}^{|I|\times|I|} \to \mathbb{R}^{|I|^2}, \qquad \Omega \mapsto f_2(\Omega) := b(\Omega) = \mathrm{vec}_r\left(R_{W_0}f_1(\Omega)\right) \tag{3.19}$$

and its transpose are given by

$$df_2(\Omega)Y = \mathrm{vec}_r\left(R_{W_0}df_1(\Omega)Y\right), \qquad \forall Y \in \mathbb{R}^{|I|\times|I|} \tag{3.20a}$$

$$df_2(\Omega)^\top Z = df_1(\Omega)^\top(R_{W_0}Z), \qquad \forall Z \in \mathbb{R}^{|I|\times|I|}. \tag{3.20b}$$

Proof: see Appendix A.1.

We note that $df_2(\Omega)^\top$ should act on a vector $\mathrm{vec}_r(Z) \in \mathbb{R}^{|I|^2}$. We prefer the more compact and equivalent non-vectorized expression (3.20b).

We turn to mapping (3) of diagram (3.16) and use (3.16).

Lemma 3.4. *The differential of the mapping*

$$f_3 \colon \mathbb{R}^{|I|\times|I|} \to \mathbb{R}^{n\times n}, \qquad \Omega \mapsto f_3(\Omega) := A^J(\Omega) = \mathrm{Diag}(R_{f_1(\Omega)})(\Omega \otimes I_{|J|}), \qquad n = |I||J| \tag{3.21}$$

is given by

$$df_3(\Omega)Y = \mathrm{Diag}(dR_{f_1(\Omega)}Y)(\Omega \otimes I_{|J|}) + \mathrm{Diag}(R_{f_1(\Omega)})(Y \otimes I_{|J|}), \qquad \forall Y \in \mathbb{R}^{|I|\times|I|}. \tag{3.22a}$$

Here, $\mathrm{Diag}(dR_{f_1(\Omega)}Y) \in \mathbb{R}^{n\times n}$ *is defined by* (2.15b) *and* $|I|$ *block matrices of size* $|J| \times |J|$ *on the diagonal of the form*

$$dR_{f_{1i}(\Omega)}Y = \mathrm{Diag}\left(df_{1i}(\Omega)Y\right) - \left(df_{1i}(\Omega)Y\right)f_{1i}(\Omega)^\top - f_{1i}(\Omega)\left(df_{1i}(\Omega)Y\right)^\top, \quad i \in I, \tag{3.22b}$$

where $df_{1i}(\Omega)Y$ *is given by*

$$(dR_{f_{1i}(\Omega)}Y)S_i = \big((dR_{f_1(\Omega)}Y)S\big)_i, \quad i \in I \tag{3.22c}$$

for any $S = (\dots, S_i, \dots)^\top \in \mathbb{R}^{|I|\times|J|}$ *and by* (3.18a).

Proof: see Appendix A.1.

We focus on the differential of the vector-valued mapping $v_T(\Omega) \in \mathbb{R}^n$ of (3.16) with n given by (2.27c). We utilize the fact that analogous to (2.30), the vector

$$v_T(\Omega) = T\varphi(TA^J(\Omega))b(\Omega) = (I_n, 0_n)\,\mathrm{expm}\left(\mathcal{A}(\Omega)\right)e_{n+1} \tag{3.23a}$$

can be extracted from the last column of the matrix

$$\mathrm{expm}\left(\mathcal{A}(\Omega)\right) = \begin{pmatrix} \mathrm{expm}\left(TA^J(\Omega)\right) & v_T(\Omega) \\ 0_n^\top & 1 \end{pmatrix}, \qquad \mathcal{A}(\Omega) = \begin{pmatrix} TA^J(\Omega) & Tb(\Omega) \\ 0_n^\top & 0 \end{pmatrix}. \tag{3.23b}$$

By means of relation (3.12), we associate a vector-valued function $f_{\mathcal{A}}$ with the matrix-valued mapping $\mathcal{A}$ through

$$\mathrm{vec}_r\left(\mathcal{A}(\Omega)\right) = f_{\mathcal{A}}\big(\mathrm{vec}_r(\Omega)\big) \tag{3.24}$$

and record for later that, for any matrix $Y \in \mathbb{R}^{|I|\times|I|}$, equation (3.13) implies

$$\mathrm{vec}_r\left(d\mathcal{A}(\Omega)Y\right) = df_{\mathcal{A}}\big(\mathrm{vec}_r(\Omega)\big)\,\mathrm{vec}_r(Y). \tag{3.25}$$

Lemma 3.5. *The differential of the mapping* $\mathcal{A}$ *in* (3.23b) *is given by*

$$d\mathcal{A}(\Omega)Y = T\begin{pmatrix} df_3(\Omega) & df_2(\Omega) \\ 0_n^\top & 0 \end{pmatrix}\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes Y\right), \qquad \forall Y \in \mathbb{R}^{|I|\times|I|}. \tag{3.26}$$

Proof. Equation (3.26) is immediate due to

$$d\mathcal{A}(\Omega) = \begin{pmatrix} TdA^J(\Omega)Y & Tdb(\Omega)Y \\ 0_n^\top & 0 \end{pmatrix} \tag{3.27}$$

and Lemmata 3.3 and 3.4. □

Now we are in the position to specify the differential of the solution to the linearized assignment flow with respect to the regularizing weight parameters.

Proposition 3.6. *Let*

$$f_4(\Omega) := v_T(\Omega) := v(T;\Omega) \tag{3.28}$$

denote the solution (2.27c) *in vectorized form to the ODE* (2.23b)*. Then the differential is given according to the convention* (3.14) *by*

$$df_4(\Omega)Y = T\Big(d\big(\varphi\big(TA^J(\Omega)\big)b(\Omega)\big) + \varphi\big(TA^J(\Omega)\big)df_2(\Omega)\Big)Y \tag{3.29a}$$

where

$$d\big(\varphi\big(TA^J(\Omega)\big)b(\Omega)\big)Y = \Big(\big(\operatorname{expm}(TA^J(\Omega)), v_T(\Omega)\big) \otimes e_{n+1}^\top\Big)\varphi\big(-\mathcal{A}(\Omega)\oplus\mathcal{A}(\Omega)^\top\big)\cdot \tag{3.29b}$$

$$\cdot\, df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)\operatorname{vec}_r(Y) + T\varphi\big(TA^J(\Omega)\big)df_2(\Omega)Y, \quad \forall Y\in\mathbb{R}^{|I|\times|I|}, \tag{3.29c}$$

where $A^J(\Omega)$ is given by (2.27b), $\mathcal{A}(\Omega)$ by (3.23b), $df_{\mathcal{A}}$ by (3.25) and Lemma 3.5, and df_2 by Lemma 3.3.

Proof. Equation (3.29a) follows directly from equation (2.27c) and Lemma 3.3 makes explicit the second summand on the right-hand side. It remains to compute the first summand. Using (3.23) and the chain rule, we have for any $Y\in\mathbb{R}^{|I|\times|I|}$,

$$d\big(T\varphi(TA^J(\Omega))b(\Omega)\big)Y = (I_n,0_n)d\operatorname{expm}\big(\mathcal{A}(\Omega)\big)\big(d\mathcal{A}(\Omega)Y\big)e_{n+1}. \tag{3.30a}$$

Applying vec_r to both sides which does not change the vector on the left-hand side, yields by (2.3)

$$d\big(T\varphi(TA^J(\Omega))b(\Omega)\big)Y = \big((I_n,0_n)\otimes e_{n+1}^\top\big)\operatorname{vec}_r\big(d\operatorname{expm}\big(\mathcal{A}(\Omega)\big)(d\mathcal{A}(\Omega)Y)\big). \tag{3.30b}$$

Applying Lemma 3.1 and (3.25), we obtain

$$d\big(T\varphi(TA^J(\Omega))b(\Omega)\big)Y = \big((I_n,0_n)\otimes e_{n+1}^\top\big)\big(\operatorname{expm}\big(\mathcal{A}(\Omega)\big)\otimes I_{n+1}\big)\varphi\big(-\mathcal{A}(\Omega)\oplus\mathcal{A}(\Omega)^\top\big) \tag{3.30c}$$

$$\cdot\, df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)\operatorname{vec}_r(Y) \tag{3.30d}$$

and using (2.2) and (3.23b)

$$= \Big(\big(\operatorname{expm}(TA^J(\Omega)), v_T(\Omega)\big)\otimes e_{n+1}^\top\Big)\varphi\big(-\mathcal{A}(\Omega)\oplus\mathcal{A}(\Omega)^\top\big) \tag{3.30e}$$

$$\cdot\, df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)\operatorname{vec}_r(Y). \qquad \square$$

We finally consider the regularizing mapping $\mathcal{R}(\Omega)$, defined by (3.6) and corresponding to arc (5) in diagram (3.16). Here, we have to take into account the constraints (2.19) imposed on Ω. Accordingly, we define the corresponding set of tangent matrices

$$\mathcal{Y}_\Omega = \big\{Y\in\mathbb{R}^{|I|\times|I|}\colon \langle \mathbb{1}_{\mathcal{N}}, Y_i|_{\mathcal{N}}\rangle = 0,\ \forall i\in I\big\}. \tag{3.31}$$

Lemma 3.7. *The differential of the mapping $\mathcal{R}$ in (3.6) is given by*

$$d\mathcal{R}(\Omega)Y = \tau\sum_{i\in I}\Big\langle t_i(\Omega), \Pi_0\Big(\frac{Y_i}{\Omega_i}\Big)\Big|_{\mathcal{N}}\Big\rangle, \qquad \forall Y\in\mathcal{Y}_\Omega. \tag{3.32}$$

Proof: see Appendix A.1.

Putting all results together, we state the main result of this section.

Theorem 3.8 (loss function gradient). *Let*

$$\mathcal{L}(\Omega) = f_{\mathcal{L}}\big(v_T(\Omega)\big) + \mathcal{R}(\Omega) \tag{3.33}$$

be a continuously differentiable loss function, where $v_T(\Omega)$ given by (2.27c) is the vectorized solution to the linearized assignment flow (2.23b) at time $t=T$. Then its gradient $\partial\mathcal{L}(\Omega)$ is given by

$$\langle\partial\mathcal{L}(\Omega), Y\rangle = d\mathcal{L}(\Omega)Y, \qquad \forall Y\in\mathcal{Y}_\Omega \tag{3.34a}$$

with

$$d\mathcal{L}(\Omega)Y = \big\langle\partial f_{\mathcal{L}}\big(v_T(\Omega)\big), df_4(\Omega)Y\big\rangle + d\mathcal{R}(\Omega)Y \tag{3.34b}$$

and $df_4(\Omega)$ given by (3.29), and with $d\mathcal{R}(\Omega)Y$ given by Lemma 3.7.

Proof. The claim (3.34) follows from applying the definition of the gradient in (3.34a) and evaluating the right-hand side using the chain rule and Proposition 3.6, to obtain (3.34b). □

3.3. **Gradient Approximation.** In this section, we discuss the complexity of the evaluation of the loss function gradient $\partial\mathcal{L}(\Omega)$ as given by (3.34), and we develop a low-rank approximation (3.47) that is computationally feasible and efficient.

3.3.1. *Motivation.* We reconsider the gradient $\partial\mathcal{L}$ given by (3.34). The gradient involves the term $df_4(\Omega)Y$, given by (3.29), which comprises three summands. We focus on the computationally expensive first summand,

$$df_4(\Omega)Y = \Big(\big(\operatorname{expm}(TA^J(\Omega)), v_T(\Omega)\big) \otimes e_{n+1}^\top\Big)\varphi\big(-\mathcal{A}(\Omega)\oplus\mathcal{A}(\Omega)^\top\big)\cdot \tag{3.35a}$$

$$\cdot\, df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)\operatorname{vec}_r(Y) + \cdots \tag{3.35b}$$

$$=: C(\Omega)\operatorname{vec}_r(Y) + \cdots . \tag{3.35c}$$

In order to evaluate the corresponding component of $\partial\mathcal{L}(\Omega)$ based on (3.34b), the matrix $C(\Omega)$ is transposed and multiplied with $\partial f_{\mathcal{L}}(v_T(\Omega))$,

$$C(\Omega)^\top\partial f_{\mathcal{L}}(v_T(\Omega)) = df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)^\top\varphi\big(-\mathcal{A}(\Omega)^\top\oplus\mathcal{A}(\Omega)\big)\cdot \tag{3.36a}$$

$$\cdot\,\Big(\big(\operatorname{expm}(TA^J(\Omega)), v_T(\Omega)\big)^\top \otimes e_{n+1}\Big)\partial f_{\mathcal{L}}(v_T(\Omega)) \tag{3.36b}$$

$$= df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)^\top\varphi\big(-\mathcal{A}(\Omega)^\top\oplus\mathcal{A}(\Omega)\big)\cdot \tag{3.36c}$$

$$\cdot\,\Big(\big(\operatorname{expm}(TA^J(\Omega)), v_T(\Omega)\big)^\top \otimes e_{n+1}\Big)\big(\partial f_{\mathcal{L}}(v_T(\Omega))\otimes(1)\big) \tag{3.36d}$$

$$\overset{(2.2)}{=} df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)^\top\varphi\big(-\mathcal{A}(\Omega)^\top\oplus\mathcal{A}(\Omega)\big)\cdot \tag{3.36e}$$

$$\cdot\,\Big(\big(\operatorname{expm}(TA^J(\Omega)), v_T(\Omega)\big)^\top\partial f_{\mathcal{L}}(v_T(\Omega))\otimes e_{n+1}\Big). \tag{3.36f}$$

Thus, the matrix-valued function φ defined by (2.25) has to be evaluates at a Kronecker sum of matrices and then multiplied by a vector. The structure of this expression has the general form

$$f(M_1\oplus M_2)(b_1\otimes b_2), \qquad M_1, M_2\in\mathbb{R}^{k\times k}, \quad b_1, b_2\in\mathbb{R}^k, \tag{3.37a}$$

where

$$M_1 = -\mathcal{A}(\Omega)^\top, \qquad M_2 = \mathcal{A}(\Omega), \qquad k = n+1 = |I||J|+1, \tag{3.37b}$$

$$b_1 = \big(\operatorname{expm}(TA^J(\Omega)), v_T(\Omega)\big)^\top\partial f_{\mathcal{L}}\big(v_T(\Omega)\big), \qquad b_2 = e_{n+1}, \tag{3.37c}$$

$$f = \varphi. \tag{3.37d}$$

We discuss two ways to compute (3.37):

Direct computation: Compute the Kronecker sum $M_1\oplus M_2$, evaluate the matrix function φ and multiply the vector $b_1\otimes b_2$. This approach has space and time complexity of at least $\mathcal{O}(k^4)$, with k given by (3.37b). The complexity might be even higher depending on how the function f is evaluated.

Krylov subspace approximation: Use the Krylov space $\mathcal{K}_m(M_1\oplus M_2, b_1\otimes b_2)$ for approximating (3.37), as explained in Section 2.4. This approach has space complexity $\mathcal{O}(k^2m^2)$ and time complexity $\mathcal{O}(k^2(m+1))$ [Saa11, p. 132].

Remark 3.9 (space complexity)**.** Consider an image with 512×512 pixels ($|I| = 262\,144$), $|J| = 10$ labels (i.e. $k = |I||J|+1 = 2\,621\,441$) and using 8 bytes per number. Then the direct computation requires to store more than 10^{14} terabytes of data. The Krylov subspace approximation (with $m = 10$) is significantly

cheaper, but still requires to store more than 5000 terabytes. Hence both methods are computationally infeasible especially in view of the fact that (3.37) has to be recomputed in every step of the gradient descent procedure (3.8).

3.3.2. *An Approximation by Benzi and Simoncini.* To reduce the memory footprint, we employ an approximation for computing (3.37), first discussed by Benzi and Simoncini [BS17], and refine it using a new additional approximation in Section 3.3.3. In the following, the notation from Benzi and Simoncini is slightly adapted to our definition (2.1) of the Kronecker sum that differs from the authors' definition of the Kronecker sum ($A \oplus B = B \otimes I + I \otimes A$).

The approach uses the Arnoldi iteration [Saa03] to determine orthonormal bases P_m, Q_m and the corresponding Hessenberg matrices T_1 and T_2 of the two Krylov subspaces $\mathcal{K}(M_1, b_1)$, $\mathcal{K}(M_2, b_2)$. The matrices are connected by a standard relation of Krylov subspaces [Hig08, Section 13.2.1],

$$M_1 P_m = P_m T_1 + t_1 p_{m+1} e_m^\top, \tag{3.38a}$$

$$M_2 Q_m = Q_m T_2 + t_2 q_{m+1} e_m^\top, \tag{3.38b}$$

where $t_1 \in \mathbb{R}$, $p_{m+1} \in \mathbb{R}^n$ (resp. $t_2 \in \mathbb{R}$, $q_{m+1} \in \mathbb{R}^n$) refer to the entries of the Hessenberg matrices and the orthonormal bases in the next step of the Arnoldi iteration. With these formulas we deduce

$$(M_1 \oplus M_2)(P_m \otimes Q_m) \overset{(2.1)}{=} (M_1 P_m \otimes Q_m) + (P_m \otimes M_2 Q_m) \tag{3.39a}$$

$$\overset{(3.38)}{=} (P_m T_1 + t_1 p_{m+1} e_m^\top \otimes Q_m) + (P_m \otimes Q_m T_2 + P_m \otimes t_2 q_{m+1} e_m^\top) \tag{3.39b}$$

$$= (Q_m \otimes P_m)(T_1 \oplus T_2) + (t_1 p_{m+1} e_m^\top \otimes Q_m) + (P_m \otimes t_2 q_{m+1} e_m^\top). \tag{3.39c}$$

Ignoring the last two summands and multiplying by $(P_m \otimes Q_m)^\top$ yields the approximation

$$(M_1 \oplus M_2) \approx (P_m \otimes Q_m)(T_1 \oplus T_2)(P_m \otimes Q_m)^\top, \tag{3.40}$$

which after applying f and multiplying $b_1 \otimes b_2$ leads to the approximation

$$f(M_1 \oplus M_2)(b_1 \otimes b_2) \approx (P_m \otimes Q_m) f(T_1 \oplus T_2)(P_m \otimes Q_m)^\top (b_1 \otimes b_2) \tag{3.41}$$

of the expression (3.37) as proposed by Benzi and Simoncini. We note that, due to the orthonormality of the bases P_m and Q_m and their relation to the vectors b_1, b_2 that generate the subspaces $\mathcal{K}(M_1, b_1)$, $\mathcal{K}(M_2, b_2)$, the approximation simplifies to

$$f(M_1 \oplus M_2)(b_1 \otimes b_2) \approx \|b_1\| \|b_2\| (P_m \otimes Q_m) f(T_1 \oplus T_2) e_1 \tag{3.42a}$$

$$= \|b_1\| \|b_2\| \operatorname{vec}_r \left(P_m \ \operatorname{vec}_r^{-1} \left(f(T_1 \oplus T_2) e_1 \right) Q_m^\top \right), \tag{3.42b}$$

where $e_1 \in \mathbb{R}^{m^2}$ denotes the first unit vector.

Remark 3.10 (complexity of the approximation (3.42b))**.** Computing and storing the matrices P_m, Q_m, T_1 and T_2 has space complexity $\mathcal{O}(2km^2)$ and a time complexity of $\mathcal{O}(2k(m+1))$ [Saa11, p. 132]. Storing the matrices $T_1 \oplus T_2$ and $f(T_1 \oplus T_2)$ has complexity $\mathcal{O}(m^4)$. Finally, multiplying the three matrices $P_m \in \mathbb{R}^{k \times m}$, $\operatorname{vec}_r^{-1}(f(T_1 \oplus T_2)e_1) \in \mathbb{R}^{m \times m}$ and $Q_m^\top \in \mathbb{R}^{m \times k}$ has time complexity $\mathcal{O}(k^2 m + k m^2)$ and space complexity $\mathcal{O}(k^2 + km)$.

Ignoring negligible terms (recall $m \ll k$), the entire approximation has computational complexity $\mathcal{O}(k^2 m)$ and storage complexity $\mathcal{O}(k^2)$. Compared to the Krylov subspace approximation of (3.37) discussed in the preceding section, this is a reduction of space complexity by a factor m^2.

Consider as in Remark 3.9 an image with 512×512 pixels ($|I| = 262\,144$) and $|J| = 10$ labels. Then the approximation (3.42b) requires to store a bit more than 50 terabytes. While this is a huge improvement compared to the 5000 terabytes from the Krylov approximation (see Remark 3.9), using this approximation is still

computationally infeasible. This motivates why we introduce below an additional low-rank approximation that yields a computationally feasible and efficient gradient approximation.

3.3.3. *Low-Rank Approximation.* We consider again the approximation (3.42b)

$$f(M_1 \oplus M_2)(b_1 \otimes b_2) \approx \|b_1\|\|b_2\| \operatorname{vec}_r \Big(P_m \ \operatorname{vec}_r^{-1}\big(f(T_1 \oplus T_2)e_1\big) Q_m^\top \Big) \tag{3.43}$$

and decompose the matrix $\operatorname{vec}_r^{-1}\big(f(T_1 \oplus T_2)e_1\big) \in \mathbb{R}^{m\times m}$ using the singular value decomposition (SVD)

$$\operatorname{vec}_r^{-1}\big(f(T_1 \oplus T_2)e_1\big) = \sum_{i\in[m]} \sigma_i y_i \otimes z_i^\top, \tag{3.44}$$

with $y_i, z_i \in \mathbb{R}^m$ and the singular values $\sigma_i \in \mathbb{R},\ i \in [m]$. As m is generally quite small, computing the SVD is neither computationally nor storage-wise expensive. We accordingly rewrite the approximation in the form

$$\|b_1\|\|b_2\| \operatorname{vec}_r \Big(P_m \ \operatorname{vec}_r^{-1}\big(f(T_1 \oplus T_2)e_1\big) Q_m^\top \Big) \tag{3.45a}$$

$$= \|b_1\|\|b_2\| \operatorname{vec}_r \Big(P_m \Big(\sum_{i\in[m]} \sigma_i y_i \otimes z_i^\top \Big) Q_m^\top \Big) \tag{3.45b}$$

$$= \|b_1\|\|b_2\| \sum_{i\in[m]} \sigma_i (P_m y_i) \otimes (Q_m z_i). \tag{3.45c}$$

Remark 3.11 (space complexity)**.** While the factorized form (3.45c) is equal to the approximation (3.42b), it requires only a fraction of the storage space: The intermediate results require storing m singular values and k numbers for each $P_m y_i$ and $Q_m z_i$, and the final approximation has an additional storage requirement of $\mathcal{O}(2km)$. In total $\mathcal{O}(4km)$ numbers need to be stored.

For a 512×512 pixels image with 10 labels (see Remark 3.9), storing this approximation requires at most a gigabyte of memory.

In practice, this can be further improved: Numerical experiments show that the singular values decline very rapidly, such that just the first singular value can be used to obtain the gradient approximation

$$f(M_1 \oplus M_2)(b_1 \otimes b_2) \approx \|b_1\|\|b_2\| \sigma_1 (P_m y_1) \otimes (Q_m z_1). \tag{3.46}$$

Numerical results in Section 4 demonstrate that this approximation is sufficiently accurate.

Remark 3.12 (space complexity)**.** The term $\|b_1\|\|b_2\|\sigma_1(P_m y_1) \otimes (Q_m z_1)$ requires to store $\mathcal{O}(2k)$ numbers, i.e. about twice as much storage space as the original image. In total, we need to store $\mathcal{O}(2k+2km)$ numbers. The required storage for the running example (see Remark 3.9) now adds up to less than 500 megabytes.

We conclude this section by returning to our problem using the notation (3.37) and state the proposed *low-rank approximation of the loss function gradient*. By (3.34), (3.36), (3.37) and (3.46), we have

$$\partial \mathcal{L}(\Omega) \approx c(\Omega) \cdot \operatorname{vec}_r^{-1}\Big(df_{\mathcal{A}}\big(\operatorname{vec}_r(\Omega)\big)^\top \big(\sigma_1 (P_m y_1) \otimes (Q_m z_1)\big) \Big) \tag{3.47a}$$

where

$$c(\Omega) = \Big\| \big(\operatorname{expm}(T A^J(\Omega)), v_T(\Omega) \big)^\top \partial f_{\mathcal{L}}\big(v_T(\Omega)\big) \Big\|, \tag{3.47b}$$

$$v_T(\Omega) = v(T;\Omega) \qquad \text{(cf. (2.27c))} \tag{3.47c}$$

$$\sigma_1 y_1 \otimes z_1^\top \approx \operatorname{vec}_r^{-1}\big(\varphi(T_1 \otimes T_2)e_1\big). \qquad \text{(top singular value and vectors)} \tag{3.47d}$$

Here, the matrices P_m, Q_m, T_1, T_2 result from the Arnoldi iteration, cf. (3.38), that returns the two Krylov subspaces used to approximate the matrix vector product $\varphi(-\mathcal{A}(\Omega)^\top \oplus \mathcal{A}(\Omega))b_1$, with b_1 given by (3.37c).

3.4. **Computing the Gradient using Automatic Differentiation.** An entirely different approach to computing the gradient $\partial\mathcal{L}(\Omega)$ of the loss function (3.7) is to not use an approximation of the exact gradient given in closed form by (3.8), but to replace the solution $v_T(\Omega)$ to the linearized assignment flow in (3.34b) by an approximation determined by a numerical integration scheme and to compute the exact gradient therefrom. Thus, one replaces an *differentiate-then-approximate* approach by an *approximate-then-differentiate* alternative. We numerically compare these two approaches in Section 4.

We sketch the latter alternative. Consider again the loss function (3.7) evaluated at the linearized assignment flow integrated up to time T

$$\mathcal{L}(\Omega) = f_{\mathcal{L}}\big(v_T(\Omega)\big). \tag{3.48}$$

Gradient approximations determined by automatic differentiation depend on what numerical is scheme is used. We pick out two basic choices out of a broad range of proper schemes studied in [ZSPS20]. In both cases, we implemented the loss function $f_{\mathcal{L}}$ in PyTorch together with the functions $\Omega \mapsto A^J(\Omega)$ and $\Omega \mapsto b(\Omega)$ given by are (2.27). Now two approximations can be distinguished depending on how the mappings $(A^J(\Omega), b(\Omega)) \mapsto v_T(\Omega) = v(T;\Omega)$ are implemented.

Automatic Differentiation based on the explicit Euler scheme: We partition the interval $[0,T]$ into T/h subintervals with some step size $h > 0$ and use the iterative scheme

$$v^{(k+1)} = v^{(k+1)} + h\big(A^J(\Omega)v^{(k)} + b(\Omega)\big), \qquad v^{(0)} = 0, \tag{3.49}$$

in order to approximate $v_T(\Omega) \approx v^{(T/h)}$ the solution to the linearized assignment flow ODE (2.27a). As the computations only involve basic linear algebra, PyTorch is able to compute the gradient using automatic differentiation.

Automatic Differentiation based on exponential integration: The second approximation utilizes the numerical integration scheme developed in Section 2.4. Again, only basic operations of linear algebra are involved so that PyTorch can compute the gradient using automatic differentiation. The more special matrix exponential (2.30) is computed by PyTorch using a Taylor polynomial approximation [BBC19].

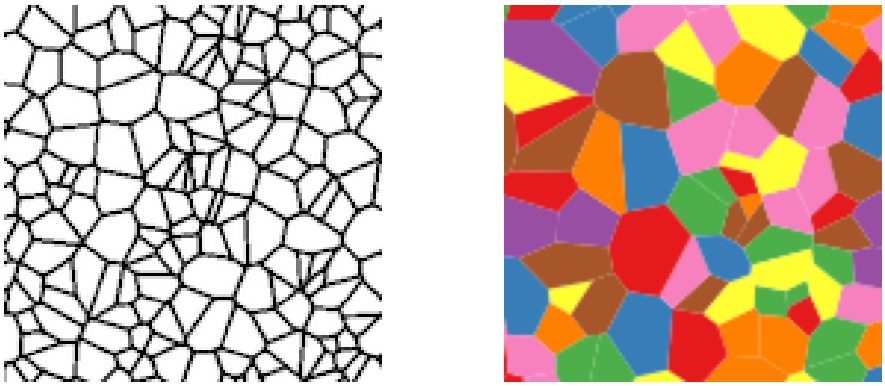
Both approaches determine an approximation of the Euclidean gradient $\partial\mathcal{L}(\Omega)$ which we subsequently convert into an approximation of the Riemannian gradient using Equation (3.9).

4. Experiments

We report and discuss results of a series of experiments devoted to the evaluation of the methods used to approximate the loss function gradient and to *estimate optimal parameters* according to Section 3.1. For this purpose, we used the two scenarios depicted in Figure 4.1.

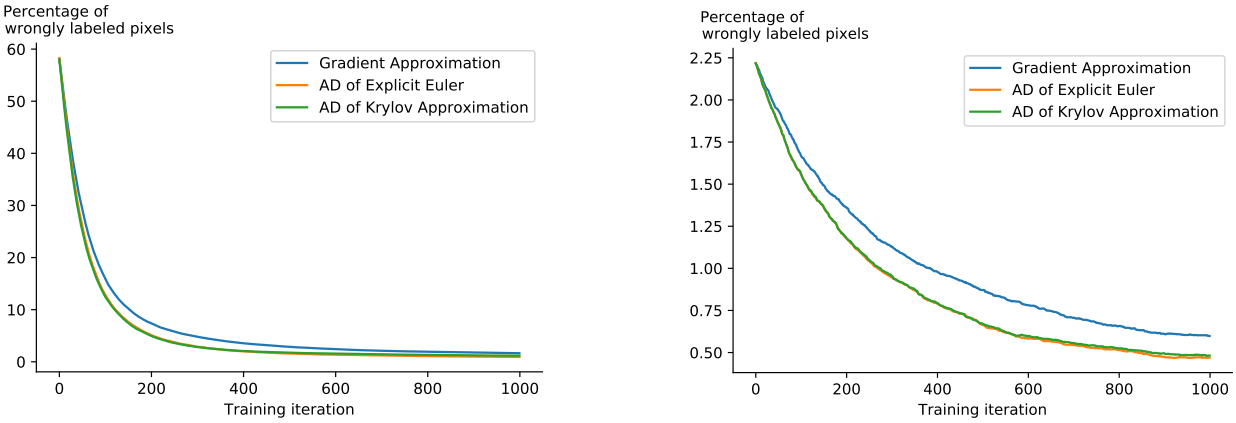
Figure 4.2 shows a comparison of parameter learning using the low-rank approximation (3.47) and automatic differentiation based on explicit Euler integration and Krylov subspace approximation, respectively, as described in Section 3.4. Generally speaking, the difference between the three algorithms is negligible compared to the influence of other hyperparameters like the step size (learning rate) h used for gradient descent, the time T used for integrating the linearized assignment flow or the size $|\mathcal{N}|$ of weight parameter patches (cf. (2.20)), respectively. This is illustrated in another way by Figure 4.3 which displays pixelwise differences of the gradient approximations. Overall, these results validate the closed form formulas in Section 3.2 and the subsequent low-rank approximation in Section 3.3.

Figure 4.4 demonstrates the influence of different hyperparameters (Krylov subspace dimension m, rank of the low-rank gradient approximation) on parameter learning. The influence of the time T of integrating the linearized assignment flow and of noise is analyzed in Figure 4.5. See the figure captions for further discussion. These experiments show that quite *low-dimensional* representations suffice for representing the



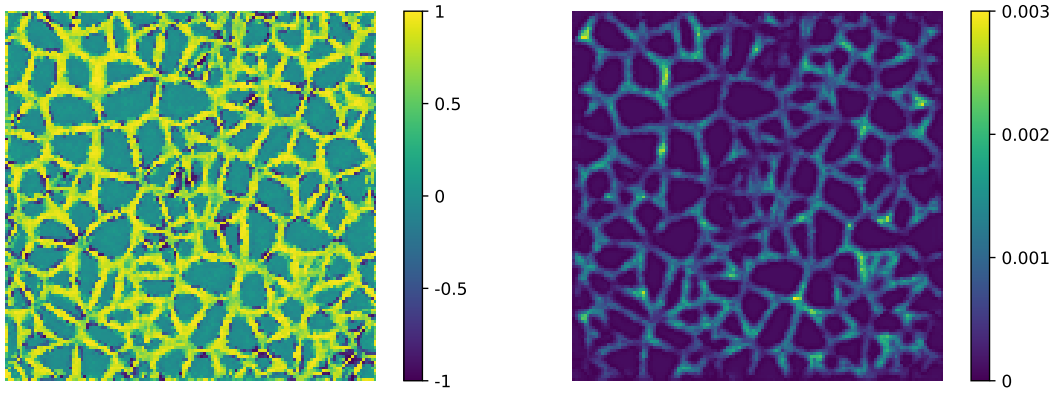
(a) Random Voronoi line structure (b) Random colored Voronoi regions

FIGURE 4.1. **Randomized scenarios for training and testing.** Two sample images of two scenarios with randomly generated images that were used to evaluate regularization parameter estimation and prediction. **(a)** Random line structure whose accurate labeling requires to adapt weight parameters. **(b)** Random Voronoi cells to be labeled according to their color (■, ■, ■, ■, ■, ■, ■, ■). In both cases, additive Gaussian noise was added as specified below.



(a) Noisy Image (b) Noiseless image

FIGURE 4.2. **Comparing gradient approximation and automatic differentiation.** Both figures show the effect of parameter learning in terms of the labeling error during the training procedure (3.8), for a noisy version of the image from Figure 4.1b (panel **(a)**) and for the noise-free image (panel **(b)**). Note the different scales of the two ordinates. For a fair comparison, the Krylov dimension $m = 10$ and the step size h of the explicit Euler scheme were chosen such that the entire parameter learning procedures took approximately the same runtime for the three different algorithms. As exemplarily shown here by both figures, we generally observed very similar results for all three algorithms which validates the closed form formulas in Section 3.2 and the subsequent subspace approximation in Section 3.3.



(a) Gradient directions (b) Norm of gradients

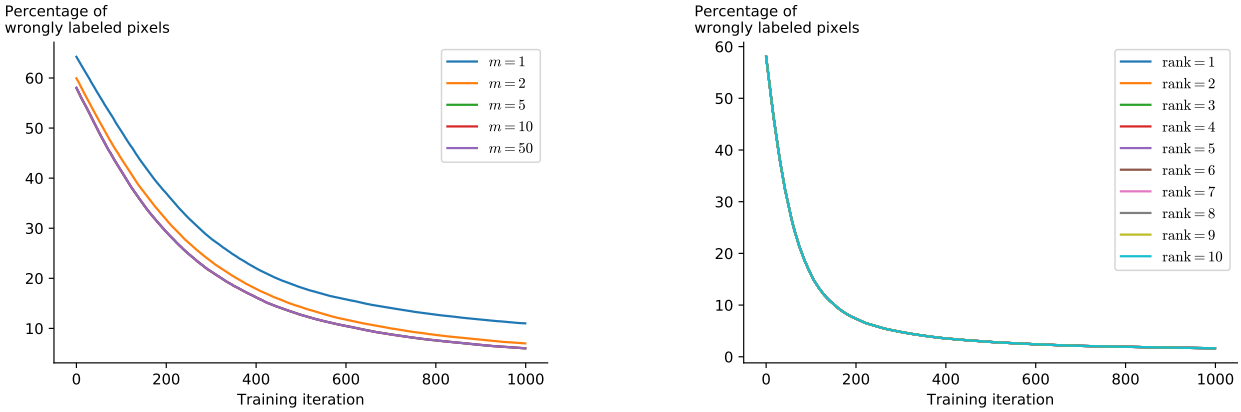
FIGURE 4.3. **Checking the gradient approximation at each pixel.** As the exact gradient is too expensive to compute, we used the gradient computed by automatic differentiation of the explicit Euler with a very small step size h as a proxy for the exact gradient. We then compared our gradient approximation (3.47) with this gradient at every pixel of the image from Figure 4.1a at the beginning of the iteration. **(a)** A value of 1 means that the gradients point in the same direction, 0 means they are orthogonal and -1 means that they point in opposite directions. We see that, especially for not yet correctly labeled pixels at the cell boundaries, the gradient directions mostly agree. In the interior of the cells, the gradients disagree more which is insignificant, however, since the norm of the gradients are close 0. **(b)** The norm of the gradients are shown at each pixel. Non-vanishing norms display where parameter learning (adaption) occurs. Since the initial weight parameter patches are uniform, no adaption happens here, corresponding to zero norms of gradients, because such parameters are optimal in homogeneous regions for noise removal.

information required for optimal regularization of dynamic image labeling. We point out that such insights cannot be gained from automatic differentiation.

Besides parameter learning, parameter *prediction* for unseen test data define another important task. In this connection, the *regularity (smoothness)* of optimal weight parameter patches is an important property since smooth patches are easier to predict both in theory and in practice. Our experiments illustrate, see Figure 4.6 and the caption, that prescribed labelings can be achieved with weight patches that are located in a *bounded* set around the uniform weight parameter patch $\mathbb{1}_{|\mathcal{N}|}$. The latter maximally smooth patch is the starting point $\Omega_i^{(0)}|_{\mathcal{N}} = \mathbb{1}_{|\mathcal{N}|}$ of the learning procedure (3.8) which determines at each pixel the closest patch such that integrating the linearized assignment flow approximates the desired labeling. As a consequence, employing additional regularization through a regularizer $\mathcal{R}$ in (3.7) is not really required.

5. CONCLUSION

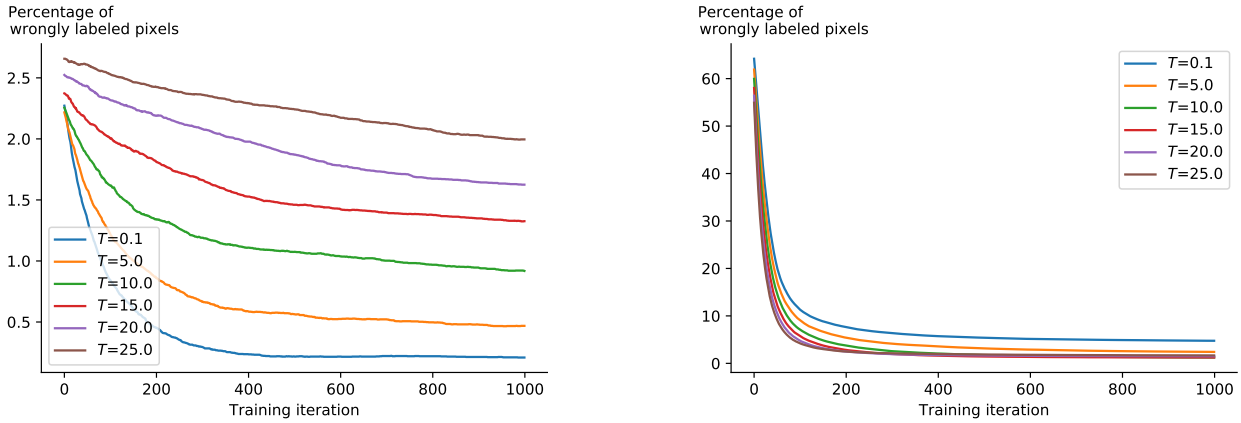
We presented a novel approach for learning the parameters of the linearized assignment flow for image labeling. Based on the exact formula of the parameter gradient of a loss function subject to the ODE-constraint, an approximation of the gradient was derived using exponential integration, a Krylov subspaces and low-rank approximation, that is efficient regarding both runtime and memory. Regarding runtime and accuracy of gradient approximation, experiments demonstrate that our research implementation is on par with highly tuned-machine learning toolboxes. Unlike the latter, however, our approach additionally returns the essential information for image labeling in terms of a low-dimensional parameter subspace.



(a) Influence of the Krylov subspace dimension

(b) Influence of the rank of the approximation

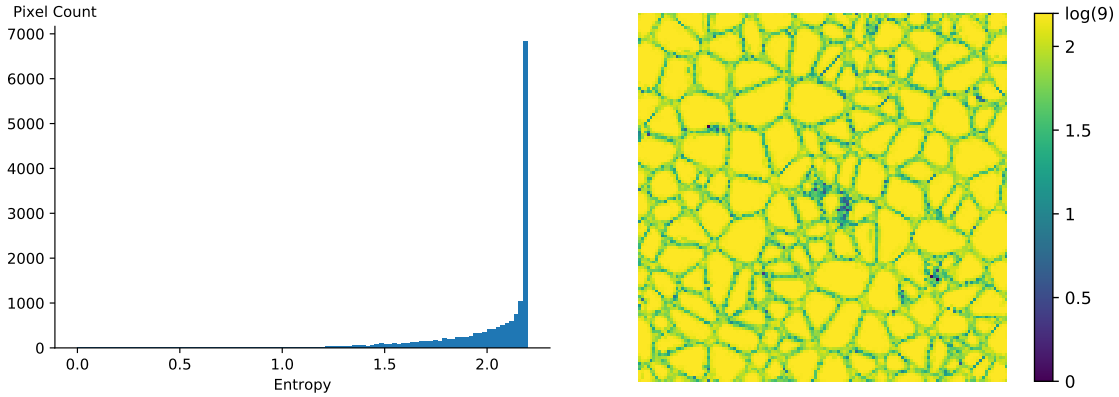
FIGURE 4.4. **Influence of the Krylov subspace dimension and the rank.** The setup as for Figure 4.2 was used to demonstrate the influence of the Krylov subspace dimension and the low-rank approximation on parameter learning. **(a)** In general, we observed that Krylov dimensions of 5 to 10 are sufficient for most experiments, here exemplarily shown for the image depicted by Figure 4.1a. Larger Krylov dimensions only increased the computation time without any noticeable improvement of accuracy. **(b)** Loss curves for different low-rank approximations coincide. This illustrates that just selecting the top singular value and vectors in (3.47) suffices for parameter learning.



(a) Training without noise

(b) Training with noise

FIGURE 4.5. **Influence of image noise and the integration time T.** The setup as for Figure 4.2 was used. Notice the different scales of the ordinates for noise-free data (panel **(a)**) and for noisy data (panel **(b)**). **(a)** In the absence of noise, *local* parameter adaption suffices, which can be accomplished with a short integration time T. Using larger values of T first averages out local image structure which then has to be restored, since parameter learning starts at uniform 'uninformed' weight patches. **(b)** Larger integration times T are beneficial for noise removal during parameter adaption. Surprisingly, a smaller number of iterations is needed for removing the bulk of erroneous labels, than in the case of noise-free data.



(a) Parameter entropy histogram

(b) Entropy of parameter patches

FIGURE 4.6. **Parameter entropy and regularization.** Both panels illustrate the entropies of the weight parameter patches (2.19) after learning. **(a)** The histogram of weight patch entropies contains a large peak that corresponds to uniform weight patches that are optimal in homogeneous regions. For almost all of the remaining pixels, however, the entropies are not smaller than 1, which indicates that the learned parameter patches are smooth. This is a favorable property for parameter prediction. **(b)** Entropies of weight parameter patches are shown at each pixel. In agreement with (a), almost all entropies are fairly large. This illustrates that parameter learning only performs minimal changes of uniform weight patches in order to approach the prespecified labeling.

Our future work will study generalizations of the linearized assignment flow. Since this can be done within the overall mathematical framework of the assignment flow approach, the result presented in this paper are applicable. We briefly indicate this for the continuous-time ODE (1.1) that we write down here again with an index 0,

$$\dot{V}_0 = A_0(\Omega_0)V_0 + B_0. \tag{5.1}$$

Recall that B_0, given by B_{W_0} of (2.23b), represents the input data (2.16) via the mappings (2.17) and (2.18). Now suppose the data are represented in another way and denoted by B_1. Then consider the additional system

$$\dot{V}_1 = A_1(\Omega_1)V_1 + B_1 + V_0(T)L, \tag{5.2}$$

where the solution $V_0(T_0)$ to (5.1) at time $t = T_0$, possibly transformed to a tangent subspace by a linear mapping L, modifies the data term B_1 of (5.2). Applying (2.25) to (5.1) at time $t = T_0$ and to (5.2) at time $t = T_1$ yields the solution

$$V_1(T_1) = T_1\varphi\big(T_1 A_1(\Omega)\big)\Big(B_1 + T_0\varphi\big(T_0 A_0(\Omega_0)\big)B_0 L\Big), \tag{5.3}$$

which is a *composition* of linearized assignment flows and hence linear too, due to the *sequential* coupling of (5.1) and (5.2). *Parallel* coupling of the dynamical systems is feasible as well and leads to larger matrix φ that is *structured* and linearly depends on the components $A_0(\Omega_0), A_1(\Omega_1), L$. Designing larger networks of this sort by repeating this steps is straightforward.

In either case, the overall basic structure of (1.1), (1.3) is preserved. This enables us to broaden the scope of assignment flows for applications and to study, in a controlled manner, various mathematical aspects of deep networks in terms of sequences of generalized linearized assignment flow, analogous to (1.6).

Acknowledgement. This work is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2181/1 - 390900948 (the Heidelberg STRUCTURES Excellence Cluster), and within the DFG priority programme 2298 on the "Theoretical Foundations of Deep Learning", grant SCHN 457/17-1.

REFERENCES

[AAB+16] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, OSDI, 2016.

[AMH11] A. H. Al-Mohy and N. J. Higham, *Computing the Action of the Matrix Exponential, with an Application to Exponential Integrators*, SIAM Journal on Scientific Computing **33** (2011), no. 2, 488–511.

[ÅPSS17] F. Åström, S. Petra, B. Schmitzer, and C. Schnörr, *Image Labeling by Assignment*, Journal of Mathematical Imaging and Vision **58** (2017), no. 2, 211–238.

[BBC19] P. Bader, S. Blanes, and F. Casas, *Computing the Matrix Exponential with an Optimized Taylor Polynomial Approximation*, Mathematics **7** (2019), no. 12, 1174.

[BPRS18] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, and J.M. Siskind, *Automatic Differentiation in Machine Learning: a Survey*, J. Machine Learning Research **18** (2018), 1–43.

[BS17] M. Benzi and V. Simoncini, *Approximation of Functions of Large Matrices with Kronecker Structure*, Numerische Mathematik **135** (2017), no. 1, 1–26.

[Hig08] N. J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, 2008.

[HL97] M. Hochbruck and C. Lubich, *On Krylov Subspace Approximations to the Matrix Exponential Operator*, SIAM J. Numer. Anal. **34** (1997), no. 5, 1911–1925.

[HNW08] E. Hairer, S.P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I*, 3rd ed., Springer, 2008.

[HO10] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numerica **19** (2010), 209–286.

[HOS09] M. Hochbruck, A. Ostermann, and J. Schweitzer, *Exponential Rosenbrock-Type Methods*, SIAM J. Numer. Anal. **47** (2009), no. 1, 786–803.

[IMKNZ00] A. Iserles, H. Z. Munthe-Kaas, S. P. Nørsett, and A. Zanna, *Lie-Group Methods*, Acta Numerica **09** (2000), 215–365.

[KKRS21] P. Kandolf, A. Koskela, S. D. Relton, and M. Schweitzer, *Computing Low-rank Approximations of the Fréchet Derivative of a Matrix Function Using Krylov Subspace Methods*, Numerical Linear Algebra with Applications (2021).

[MVL03] C. Moler and C. Van Loan, *Nineteen Dubious Ways to Compute the Exponential of a Matrix, Twenty-Five Years Later*, SIAM Review **45** (2003), no. 1, 3–49.

[NW12] J. Niesen and W.M. Wright, *Algorithm 919: A Krylov Subspace Algorithm for Evaluating the φ-Functions Appearing in Exponential Integrators*, ACM Trans. Math. Software **38** (2012), no. 3, Article 22.

[PGM+19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, NIPS, vol. 32, Curran Associates, Inc., 2019.

[Saa92] Y. Saad, *Analysis of Some Krylov Subspace Approximations to the Matrix Exponential Operator*, SIAM Journal on Numerical Analysis **29** (1992), no. 1, 209–228.

[Saa03] ———, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.

[Saa11] ———, *Numerical Methods for Large Eigenvalue Problems*, Classics in Applied Mathematics, Society for Industrial and Applied Mathematics, 2011.

[Sch20] C. Schnörr, *Assignment Flows*, Variational Methods for Nonlinear Geometric Data and Applications (P. Grohs, M. Holler, and A. Weinmann, eds.), Springer, 2020, pp. 235–260.

[Tes12] G. Teschl, *Ordinary Differential Equations and Dynamical Systems*, Grad. Studies Math., vol. 140, Amer. Math. Soc., 2012.

[VL00] C. F. Van Loan, *The Ubiquitous Kronecker Product*, J. Comput. Appl. Math. **123** (2000), 85–100.

[ZPS21] A. Zeilmann, S. Petra, and C. Schnörr, *Learning Linear Assignment Flows for Image Labeling via Exponential Integration*, Scale Space and Variational Methods in Computer Vision (SSVM) (A. Elmoataz, J. Fadili, Y. Quéau, J. Rabin, and L. Simon, eds.), LNCS, vol. 12679, 2021, pp. 385–397.

[ZSPS20] A. Zeilmann, F. Savarino, S. Petra, and C. Schnörr, *Geometric Numerical Integration of the Assignment Flow*, Inverse Problems **36** (2020), no. 3, 034004 (33pp).

[ZZS20] A. Zern, A. Zeilmann, and C. Schnörr, *Assignment Flows for Data Labeling on Graphs: Convergence and Stability*, arXiv:2002.11571 (2020).

APPENDIX A. PROOFS

A.1. **Proofs of Section 3.2.2.**

Proof of Lemma 3.2. Regarding the differential of the mapping (2.12e) with respect to its second argument, we have $d\exp_p(u)v = R_{\exp_p(u)}v$ by [ZSPS20, Lemma 4.5], with R given by (2.12b). Applying this relation to (3.17) where $\exp_{\mathbb{1}_\mathcal{W}}$ acts row-wise analogous to the mapping R_W as explained by (2.13) and (2.15), yields

$$df_1(\Omega)Y = R_{\exp_{\mathbb{1}_\mathcal{W}}(-\frac{1}{\rho}\Omega D)}\Big(-\frac{1}{\rho}YD\Big) = R_{f_1(\Omega)}\Big(-\frac{1}{\rho}YD\Big), \qquad \forall Y \in \mathbb{R}^{|I|\times|I|}, \tag{A.1}$$

which is (3.18a). As for the transpose, we vectorize both sides using again (2.15),

$$\operatorname{vec}_r\big(df_1(\Omega)Y\big) = \operatorname{Diag}(R_{f_1(\Omega)})\operatorname{vec}_r\Big(-\frac{1}{\rho}YD\Big) = -\frac{1}{\rho}\operatorname{Diag}(R_{f_1(\Omega)})(I_{|I|}\otimes D^\top)\operatorname{vec}_r(Y). \tag{A.2}$$

Applying the transposed matrix to any vector $\operatorname{vec}_r(Z)$ with $Z \in \mathbb{R}^{|I|\times|J|}$ and taking into account the symmetry of the matrix $\operatorname{Diag}(R_{f_1(\Omega)})$, yields

$$\begin{aligned} df_1(\Omega)^\top Z &= -\frac{1}{\rho}\operatorname{vec}_r^{-1}\big((I_{|I|}\otimes D)\operatorname{Diag}(R_{f_1(\Omega)})\operatorname{vec}_r(Z)\big) && \text{(A.3a)}\\ &\overset{(2.15)}{=} -\frac{1}{\rho}\operatorname{vec}_r^{-1}\big((I_{|I|}\otimes D)\operatorname{vec}_r(R_{f_1(\Omega)}Z)\big) = -\frac{1}{\rho}R_{f_1(\Omega)}(Z)D^\top. && \square \end{aligned}$$

Proof of Lemma 3.3. Since R_{W_0} does not depend on Ω and vec_r is linear, we directly obtain (3.20a). Regarding the transpose map, we expand the right-hand side of (3.20a),

$$df_2(\Omega)Y \overset{(2.15)}{=} \operatorname{Diag}(R_{W_0})\operatorname{vec}_r(df_1(\Omega)Y) \overset{(A.2)}{=} -\frac{1}{\rho}\operatorname{Diag}(R_{W_0})\operatorname{Diag}(R_{f_1(\Omega)})(I_{|I|}\otimes D^\top)\operatorname{vec}_r(Y). \tag{A.4}$$

Applying the transposed matrix to any vector $\operatorname{vec}_r(Z) \in \mathbb{R}^{|I|^2}$ yields (recall that the matrices $\operatorname{Diag}(R_{W_0})$, $\operatorname{Diag}(R_{f_1(\Omega)})$ are symmetric)

$$\begin{aligned} df_2(\Omega)^\top Z &= -\frac{1}{\rho}\operatorname{vec}_r^{-1}\big((I_{|I|}\otimes D)\operatorname{Diag}(R_{f_1(\Omega)})\operatorname{Diag}(R_{W_0})\operatorname{vec}_r(Z)\big) && \text{(A.5a)}\\ &\overset{(2.15)}{=} -\frac{1}{\rho}\operatorname{vec}_r^{-1}\big((I_{|I|}\otimes D)\operatorname{Diag}(R_{f_1(\Omega)})\operatorname{vec}_r(R_{W_0}Z)\big) && \text{(A.5b)}\\ &\overset{(2.15)}{=} -\frac{1}{\rho}\operatorname{vec}_r^{-1}\Big((I_{|I|}\otimes D)\operatorname{vec}_r\big(R_{f_1(\Omega)}(R_{W_0}Z)\big)\Big) = -\frac{1}{\rho}R_{f_1(\Omega)}(R_{W_0}Z)D^\top && \text{(A.5c)}\\ &\overset{(3.18b)}{=} df_1(\Omega)^\top(R_{W_0}Z). && \square \end{aligned}$$

Proof of Lemma 3.4. We have

$$df_3(\Omega)Y = \big(d\operatorname{Diag}(R_{f_1(\Omega)})Y\big)(\Omega\otimes I_{|J|}) + \operatorname{Diag}(R_{f_1(\Omega)})(Y\otimes I_{|J|}), \quad \forall Y \in \mathbb{R}^{|I|\times|I|} \tag{A.6}$$

and have to the differential in the first summand on the right-hand side. By (2.15),

$$\operatorname{Diag}(R_{f_1(\Omega)})\operatorname{vec}_r(S) = \operatorname{vec}_r(R_{f_1(\Omega)}S), \quad \forall S \in \mathbb{R}^{|I|\times|J|} \tag{A.7}$$

and hence $d\operatorname{Diag}(R_{f_1(\Omega)})$ is given by

$$\big(d\operatorname{Diag}(R_{f_1(\Omega)})Y\big)\operatorname{vec}_r(S) = \operatorname{vec}_r\big((dR_{f_1(\Omega)}Y)S\big), \qquad \forall Y \in \mathbb{R}^{|I|\times|I|}, \quad \forall S \in \mathbb{R}^{|I|\times|J|}. \tag{A.8}$$

It remains to compute $dR_{f_1(\Omega)}$ and to evaluate the defining right-hand side, to obtain the left-hand side in explicit form. Focusing on a single component $R_{f_{1i}}(\Omega)$ of the mapping $R_{f_1(\Omega)}$, we have by (2.12b)

$$R_{f_{1i}}(\Omega) = \operatorname{Diag}\big(f_{1i}(\Omega)\big) - f_{1i}(\Omega) f_{1i}(\Omega)^\top \tag{A.9a}$$

$$dR_{f_{1i}(\Omega)}Y = \operatorname{Diag}\big(df_{1i}(\Omega)Y\big) - \big(df_{1i}(\Omega)Y\big) f_{1i}(\Omega)^\top - f_{1i}(\Omega)\big(df_{1i}(\Omega)Y\big)^\top \tag{A.9b}$$

and hence for any $S_i \in \mathbb{R}^{|J|}$ and $S = (\dots, S_i, \dots)^\top \in \mathbb{R}^{|I|\times|J|}$

$$(dR_{f_{1i}(\Omega)}Y)S_i = \big((dR_{f_1(\Omega)}Y)S\big)_i, \quad i \in I. \tag{A.9c}$$

Thus, analogous to (2.15), we obtain

$$(dR_{f_1(\Omega)}Y)S = \big(\dots, (dR_{f_{1i}(\Omega)}Y)S_i, \dots\big)^\top = \operatorname{vec}_r^{-1}\Big(\big(\operatorname{Diag}(dR_{f_1(\Omega)})Y\big)\operatorname{vec}_r(S)\Big). \tag{A.9d}$$

Applying vec_r to both sides and comparing with (A.8), we conclude

$$d\operatorname{Diag}(R_{f_1(\Omega)})Y = \operatorname{Diag}(dR_{f_1(\Omega)}Y) \tag{A.9e}$$

which proves (3.22). □

Proof of Lemma 3.7. The mapping $\exp_p$ specified by (2.12e) satisfies $\exp_p = \exp_p \circ \Pi_0$ and a short computation [ÅPSS17, Appendix]) shows that the restriction $\exp_p|_{T_0}$, again denoted by $\exp_p$, has the inverse

$$\exp_p^{-1}\colon \mathcal{S} \to T_0, \qquad q \mapsto \Pi_0(\log q - \log p) \tag{A.10}$$

and consequently the differential

$$d\exp_p^{-1}(q)u = \Pi_0\Big(\frac{u}{q}\Big), \quad u \in T_0. \tag{A.11}$$

For $W, \widetilde{W} \in \mathcal{W}$ and $V \in \mathcal{T}_0$, this differential applies componentwise, i.e.

$$\big(d\exp_W^{-1}(\widetilde{W})V\big)_i = \Pi_0\Big(\frac{V_i}{\widetilde{W}_i}\Big), \quad i \in I. \tag{A.12}$$

Application to (3.6) yields for any $Y \in \mathcal{Y}_\Omega$ equation (3.32). □

(A. Zeilmann, C. Schnörr) IMAGE AND PATTERN ANALYSIS GROUP, HEIDELBERG UNIVERSITY, GERMANY
URL: https://ipa.math.uni-heidelberg.de

(S. Petra) MATHEMATICAL IMAGING GROUP, HEIDELBERG UNIVERSITY, GERMANY
URL: https://www.stpetra.com