Scalable Optimal Transport in High Dimensions for Graph Distances, Embedding Alignment, and More

Johannes Gasteiger [1] **Marten Lienen** [1] **Stephan Günnemann** [1]

Abstract

The current best practice for computing optimal transport (OT) is via entropy regularization and Sinkhorn iterations. This algorithm runs in quadratic time as it requires the full pairwise cost matrix, which is prohibitively expensive for large sets of objects. In this work we propose two effective log-linear time approximations of the cost matrix: First, a sparse approximation based on locality-sensitive hashing (LSH) and, second, a Nyström approximation with LSH-based sparse corrections, which we call locally corrected Nyström (LCN). These approximations enable general log-linear time algorithms for entropy-regularized OT that perform well even for the complex, high-dimensional spaces common in deep learning. We analyse these approximations theoretically and evaluate them experimentally both directly and end-to-end as a component for real-world applications. Using our approximations for unsupervised word embedding alignment enables us to speed up a state-of-the-art method by a factor of 3 while also improving the accuracy by 3.1 percentage points without any additional model changes. For graph distance regression we propose the graph transport network (GTN), which combines graph neural networks (GNNs) with enhanced Sinkhorn. GTN outcompetes previous models by 48 % and still scales log-linearly in the number of nodes.

1. Introduction

Measuring the distance between two distributions or sets of objects is a central problem in machine learning. One common method of solving this is optimal transport (OT). OT is concerned with the problem of finding the transport plan for moving a source distribution (e.g. a pile of earth) to a sink distribution (e.g. a construction pit) with the cheapest cost w.r.t. some pointwise cost function (e.g. the Euclidean distance). The advantages of this method have been shown numerous times, e.g. in generative modelling (Arjovsky et al., 2017; Bousquet et al., 2017; Genevay et al., 2018), loss functions (Frogner et al., 2015), set matching (Wang et al., 2019), or domain adaptation (Courty et al., 2017). Motivated by this, many different methods for accelerating OT have been proposed in recent years (Indyk & Thaper, 2003; Papadakis et al., 2014; Backurs et al., 2020). However, most of these approaches are specialized methods that do not generalize to modern deep learning models, which rely on dynamically changing high-dimensional embeddings.

In this work we make OT computation for high-dimensional point sets more scalable by introducing two fast and accurate approximations of entropy-regularized optimal transport: Sparse Sinkhorn and LCN-Sinkhorn, the latter relying on our novel locally corrected Nyström (LCN) method. Sparse Sinkhorn uses a sparse cost matrix to leverage the fact that in entropy-regularized OT (also known as the Sinkhorn distance) (Cuturi, 2013) often only each point's nearest neighbors influence the result. LCN-Sinkhorn extends this approach by leveraging LCN, a general similarity matrix approximation that fuses local (sparse) and global (low-rank) approximations, allowing us to simultaneously capture interactions between both close and far points. LCN-Sinkhorn thus fuses sparse Sinkhorn and Nyström-Sinkhorn (Altschuler et al., 2019). Both sparse Sinkhorn and LCN-Sinkhorn run in log-linear time.

We theoretically analyze these approximations and show that sparse corrections can lead to significant improvements over the Nyström approximation. We furthermore validate these approximations by showing that they are able to reproduce both the Sinkhorn distance and transport plan significantly better than previous methods across a wide range of regularization parameters and computational budgets (as e.g. demonstrated in Fig. 1). We then show the impact of these improvements by employing Sinkhorn approximations end-to-end in two high-impact machine learning tasks. First, we incorporate them into Wasserstein Procrustes for word embedding alignment (Grave et al., 2019). Without any further model changes LCN-Sinkhorn improves upon the

[1]Technical University of Munich, Germany. Correspondence to: Johannes Gasteiger <j.gasteiger@in.tum.de>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

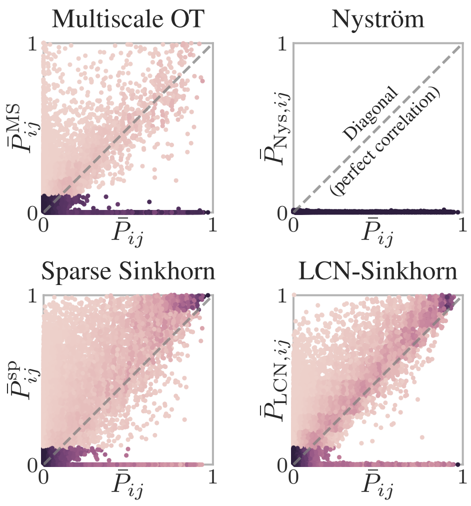

Figure 1: The proposed methods (sparse and LCN-Sinkhorn) show a clear correlation with the full Sinkhorn transport plan, as opposed to previous methods. Entries of approximations (y-axis) and full Sinkhorn (x-axis) for pre-aligned word embeddings (EN-DE). Color denotes density.

original method's accuracy by 3.1 percentage points using a third of the training time. Second, we develop the graph transport network (GTN), which combines graph neural networks (GNNs) with optimal transport for graph distance regression, and further improve it via learnable unbalanced OT and multi-head OT. GTN with LCN-Sinkhorn is the first model that both overcomes the bottleneck of using a single embedding per graph and scales log-linearly in the number of nodes. Our implementation is available online.[1] In summary, our paper's main contributions are:

- **Locally Corrected Nyström (LCN)**, a flexible log-linear time approximation for similarity matrices, merging local (sparse) and global (low-rank) approximations.
- Entropy-regularized optimal transport (a.k.a. Sinkhorn distance) with log-linear runtime via **sparse Sinkhorn** and **LCN-Sinkhorn**. These are the first log-linear approximations that are stable enough to substitute full entropy-regularized OT in models using high-dimensional spaces.
- The **graph transport network (GTN)**, a siamese GNN using multi-head unbalanced LCN-Sinkhorn. GTN both sets the state of the art on graph distance regression and still scales log-linearly in the number of nodes.

2. Entropy-regularized optimal transport

This work focuses on optimal transport between two discrete sets of points. We use entropy regularization, which enables fast computation and often performs better than regular OT

[1] `https://www.daml.in.tum.de/lcn`

(Cuturi, 2013). Formally, given two categorical distributions modelled via the vectors $\boldsymbol{p} \in \mathbb{R}^n$ and $\boldsymbol{q} \in \mathbb{R}^m$ supported on two sets of points $X_\text{p} = \{\boldsymbol{x}_{\text{p}1}, \dots, \boldsymbol{x}_{\text{p}n}\}$ and $X_\text{q} = \{\boldsymbol{x}_{\text{q}1}, \dots, \boldsymbol{x}_{\text{q}m}\}$ in $\mathbb{R}^d$ and the cost function $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ (e.g. the L_2 distance) giving rise to the cost matrix $\boldsymbol{C}_{ij} = c(\boldsymbol{x}_{\text{p}i}, \boldsymbol{x}_{\text{q}i})$ we aim to find the Sinkhorn distance d_c^λ and the associated optimal transport plan $\bar{\boldsymbol{P}}$ (Cuturi, 2013)

$$\begin{aligned} \bar{\boldsymbol{P}} &= \underset{\boldsymbol{P}}{\arg\min} \langle \boldsymbol{P}, \boldsymbol{C} \rangle_\text{F} - \lambda H(\boldsymbol{P}), \\ d_c^\lambda &= \langle \boldsymbol{P}, \boldsymbol{C} \rangle_\text{F} - \lambda H(\boldsymbol{P}), \\ \text{s.t.} \quad & \boldsymbol{P}\boldsymbol{1}_m = \boldsymbol{p}, \ \boldsymbol{P}^T\boldsymbol{1}_n = \boldsymbol{q}, \end{aligned} \tag{1}$$

with the Frobenius inner product $\langle ., . \rangle_\text{F}$ and the entropy $H(\boldsymbol{P}) = -\sum_{i=1}^n \sum_{j=1}^m \boldsymbol{P}_{ij} \log \boldsymbol{P}_{ij}$. Note that d_c^λ includes the entropy and can thus be negative, while Cuturi (2013) originally used $d^{1/\lambda}_{\text{Cuturi},c} = \langle \bar{\boldsymbol{P}}, \boldsymbol{C} \rangle_\text{F}$. This optimization problem can be solved by finding the vectors $\bar{\boldsymbol{s}}$ and $\bar{\boldsymbol{t}}$ that normalize the columns and rows of the matrix $\bar{\boldsymbol{P}} = \operatorname{diag}(\bar{\boldsymbol{s}}) \boldsymbol{K} \operatorname{diag}(\bar{\boldsymbol{t}})$ with the similarity matrix $\boldsymbol{K}_{ij} = e^{-\frac{\boldsymbol{C}_{ij}}{\lambda}}$, so that $\bar{\boldsymbol{P}}\boldsymbol{1}_m = \boldsymbol{p}$ and $\bar{\boldsymbol{P}}^T\boldsymbol{1}_n = \boldsymbol{q}$. We can achieve this via the Sinkhorn algorithm, which initializes the normalization vectors as $\boldsymbol{s}^{(1)} = \boldsymbol{1}_n$ and $\boldsymbol{t}^{(1)} = \boldsymbol{1}_m$ and updates them alternatingly via (Sinkhorn & Knopp, 1967)

$$\boldsymbol{s}^{(i)} = \boldsymbol{p} \oslash (\boldsymbol{K}\boldsymbol{t}^{(i-1)}), \quad \boldsymbol{t}^{(i)} = \boldsymbol{q} \oslash (\boldsymbol{K}^T\boldsymbol{s}^{(i)}) \tag{2}$$

until convergence, where $\oslash$ denotes elementwise division.

3. Sparse Sinkhorn

The Sinkhorn algorithm is faster than non-regularized EMD algorithms, which run in $\mathcal{O}(n^2 m \log n \log(n \max(\boldsymbol{C})))$ (Tarjan, 1997). However, its computational cost is still quadratic in time $\mathcal{O}(nm)$, which is prohibitively expensive for large n and m. We overcome this by observing that the matrix $\boldsymbol{K}$, and hence also $\bar{\boldsymbol{P}}$, is negligibly small everywhere except at each point's closest neighbors because of the exponential used in $\boldsymbol{K}$'s computation. We propose to leverage this by approximating $\boldsymbol{C}$ via the sparse matrix $\boldsymbol{C}^\text{sp}$, where

$$\boldsymbol{C}^\text{sp}_{ij} = \begin{cases} \boldsymbol{C}_{ij} & \text{if } \boldsymbol{x}_{\text{p}i} \text{ and } \boldsymbol{x}_{\text{q}j} \text{ are “near”}, \\ \infty & \text{otherwise}. \end{cases} \tag{3}$$

$\boldsymbol{K}^\text{sp}$ and $\bar{\boldsymbol{P}}^\text{sp}$ follow from the definitions of $\boldsymbol{K}$ and $\bar{\boldsymbol{P}}$. Finding “near” neighbors can be approximately solved via locality-sensitive hashing (LSH) on $X_\text{p} \cup X_\text{q}$.

Locality-sensitive hashing. LSH tries to filter “near” from “far” data points by putting them into different hash buckets. Points closer than a certain distance r_1 are put into the same bucket with probability at least p_1, while those beyond some distance $r_2 = c \cdot r_1$ with $c > 1$ are put into the same bucket with probability at most $p_2 \ll p_1$. There is a plethora of LSH methods for different metric spaces and their associated cost (similarity/distance) functions (Wang et al., 2014;

Shrivastava & Li, 2014), and we can use any of them. In this work we focus on cross-polytope LSH (Andoni et al., 2015) and k-means LSH (Paulevé et al., 2010) (see App. H). We can control the (average) number of neighbors via the number of hash buckets. This allows sparse Sinkhorn with LSH to scale log-linearly with the number of points, i.e. $\mathcal{O}(n \log n)$ for $n \approx m$ (see App. A and App. K). Unfortunately, Sinkhorn with LSH can fail when e.g. the cost is evenly distributed or the matrix $\boldsymbol{K}^{\text{sp}}$ does not have support (see App. B). However, we can alleviate these limitations by fusing $\boldsymbol{K}^{\text{sp}}$ with the Nyström method.

4. Locally corrected Nyström and LCN-Sinkhorn

Nyström method. The Nyström method is a popular way of approximating similarity matrices that provides performance guarantees for many important tasks (Williams & Seeger, 2001; Musco & Musco, 2017). It approximates a positive semi-definite (PSD) similarity matrix $\boldsymbol{K}$ via its low-rank decomposition $\boldsymbol{K}_{\text{Nys}} = \boldsymbol{U}\boldsymbol{A}^{-1}\boldsymbol{V}$. Since the optimal decomposition via SVD is too expensive to compute, Nyström instead chooses a set of l landmarks $L = \{\boldsymbol{x}_{\text{l}1}, \dots, \boldsymbol{x}_{\text{l}l}\}$ and obtains the matrices via $\boldsymbol{U}_{ij} = k(\boldsymbol{x}_{\text{p}i}, \boldsymbol{x}_{\text{l}j})$, $\boldsymbol{A}_{ij} = k(\boldsymbol{x}_{\text{l}i}, \boldsymbol{x}_{\text{l}j})$, and $\boldsymbol{V}_{ij} = k(\boldsymbol{x}_{\text{l}i}, \boldsymbol{x}_{\text{q}j})$, where $k(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is an arbitrary PSD kernel, e.g. $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\frac{c(\boldsymbol{x}_1, \boldsymbol{x}_2)}{\lambda}}$ for Sinkhorn. Common methods of choosing landmarks from $X_{\text{p}} \cup X_{\text{q}}$ are uniform and ridge leverage score (RLS) sampling. We instead focus on k-means Nyström and sampling via k-means++, which we found to be significantly faster than recursive RLS sampling (Zhang et al., 2008) and perform better than both uniform and RLS sampling (see App. H).

Sparse vs. Nyström. Exponential kernels like the one used for $\boldsymbol{K}$ (e.g. the Gaussian kernel) typically correspond to a reproducing kernel Hilbert space that is infinitely dimensional. The resulting Gram matrix $\boldsymbol{K}$ thus usually has full rank. A low-rank approximation like the Nyström method can therefore only account for its global structure and not the local structure around each point $\boldsymbol{x}$. As such, it is ill-suited for any moderately low entropy regularization parameter, where the transport matrix $\bar{\boldsymbol{P}}$ resembles a permutation matrix. Sparse Sinkhorn, on the other hand, cannot account for global structure and instead approximates all non-selected distances as infinity. It will hence fail if more than a handful of neighbors are required per point. These approximations are thus opposites of each other, and as such not competing but rather *complementary* approaches.

Locally corrected Nyström. Since the entries in our sparse approximation are exact, we can directly fuse it with the Nyström approximation. For the indices of all non-zero values in the sparse approximation $\boldsymbol{K}^{\text{sp}}$ we calculate the corresponding entries in the Nyström approximation, obtaining the sparse matrix $\boldsymbol{K}^{\text{sp}}_{\text{Nys}}$. To obtain the locally corrected Nyström (LCN) approximation[2] we subtract these entries from $\boldsymbol{K}_{\text{Nys}}$ and replace them with their exact values, i.e.

$$\boldsymbol{K}_{\text{LCN}} = \boldsymbol{K}_{\text{Nys}} - \boldsymbol{K}^{\text{sp}}_{\text{Nys}} + \boldsymbol{K}^{\text{sp}} = \boldsymbol{K}_{\text{Nys}} + \boldsymbol{K}^{\text{sp}}_{\Delta}. \tag{4}$$

LCN-Sinkhorn. To obtain the approximate transport plan $\bar{\boldsymbol{P}}_{\text{LCN}}$ we run the Sinkhorn algorithm with $\boldsymbol{K}_{\text{LCN}}$ instead of $\boldsymbol{K}$. However, we never fully instantiate $\boldsymbol{K}_{\text{LCN}}$. Instead, we directly use the decomposition and calculate the matrix-vector product in Eq. (2) as $\boldsymbol{K}_{\text{LCN}}\boldsymbol{t} = \boldsymbol{U}(\boldsymbol{A}^{-1}\boldsymbol{V}\boldsymbol{t}) + \boldsymbol{K}^{\text{sp}}_{\Delta}\boldsymbol{t}$, similarly to Altschuler et al. (2019). As a result we obtain the decomposed approximate OT plan $\bar{\boldsymbol{P}}_{\text{LCN}} = \bar{\boldsymbol{P}}_{\text{Nys}} + \bar{\boldsymbol{P}}^{\text{sp}}_{\Delta} = \bar{\boldsymbol{P}}_U \bar{\boldsymbol{P}}_W + \bar{\boldsymbol{P}}^{\text{sp}} - \bar{\boldsymbol{P}}^{\text{sp}}_{\text{Nys}}$ and the approximate distance (using Lemma A from Altschuler et al. (2019))

$$\begin{aligned} d^{\lambda}_{\text{LCN},c} = \lambda(&\boldsymbol{s}^T \bar{\boldsymbol{P}}_U \bar{\boldsymbol{P}}_W \boldsymbol{1}_m + \boldsymbol{1}_n^T \bar{\boldsymbol{P}}_U \bar{\boldsymbol{P}}_W \boldsymbol{t} \\ &+ \boldsymbol{s}^T \bar{\boldsymbol{P}}^{\text{sp}}_{\Delta} \boldsymbol{1}_m + \boldsymbol{1}_n^T \bar{\boldsymbol{P}}^{\text{sp}}_{\Delta} \boldsymbol{t}). \end{aligned} \tag{5}$$

This approximation scales log-linearly with dataset size (see App. A and App. K for details). It allows us to smoothly move from Nyström-Sinkhorn to sparse Sinkhorn by varying the number of landmarks and neighbors. We can thus freely choose the optimal "operating point" based on the underlying problem and regularization parameter. We discuss the limitations of LCN-Sinkhorn in App. B.

5. Theoretical analysis

Approximation error. The main question we aim to answer in our theoretical analysis is what improvements to expect from adding sparse corrections to Nyström Sinkhorn. To do so, we first analyse approximations of $\boldsymbol{K}$ in a uniform and a clustered data model. In these we use Nyström and LSH schemes that largely resemble k-means, as used in most of our experiments. Relevant proofs and notes for this section can be found in App. C to G.

Theorem 1. *Let X_{p} and X_{q} have n samples that are uniformly distributed on a d-dimensional closed, locally Euclidean manifold. Let $\boldsymbol{C}_{ij} = \|\boldsymbol{x}_{\text{p}i} - \boldsymbol{x}_{\text{q}j}\|_2$ and $\boldsymbol{K}_{ij} = e^{-\boldsymbol{C}_{ij}/\lambda}$. Let the landmarks be arranged a priori, with a minimum distance $2R$ between each other. Then the expected error of the Nyström approximation $\boldsymbol{K}_{\text{Nys}}$ between a point $\boldsymbol{x}_{\text{p}i}$ and its kth-nearest neighbor $\boldsymbol{x}_{\text{q}i_k}$ is*

$$\mathbb{E}[\boldsymbol{K}_{i,i_k} - \boldsymbol{K}_{\text{Nys},i,i_k}] = \mathbb{E}[e^{-\delta_k/\lambda}] - \mathbb{E}[\boldsymbol{K}_{\text{Nys},i,i_k}], \tag{6}$$

with δ_k denoting the kth-nearest neighbor distance. With $\Gamma(.,.)$ denoting the upper incomplete Gamma function the

[2]LCN has an unrelated namesake on integrals, which uses high-order term to correct quadrature methods around singularities (Canino et al., 1998).

second term is bounded by

$$\mathbb{E}[\boldsymbol{K}_{\text{Nys},i,i_k}] \leq \frac{d(\Gamma(d) - \Gamma(d, 2R/\lambda))}{(2R/\lambda)^d} + \mathcal{O}(e^{-2R/\lambda}). \tag{7}$$

Eq. (6) is therefore dominated by $\mathbb{E}[e^{-\delta_k/\lambda}]$ if $\delta_k \ll R$, which is a reasonable assumption given that R only decreases slowly with the number of landmarks l since $R \geq (\frac{(d/2)!}{l})^{1/d} \frac{1}{2\sqrt{\pi}}$ (Cohn, 2017). In this case the approximation's largest error is the one associated with the point's nearest neighbor. LCN uses the exact result for these nearest neighbors and therefore removes the largest errors, providing significant benefits even for uniform data. For example, just removing the first neighbor's error we obtain a 68 % decrease in the dominant first term ($d = 32$, $\lambda = 0.05$, $n = 1000$). This is even more pronounced in clustered data.

Theorem 2. *Let $X_\text{p}, X_\text{q} \subseteq \mathbb{R}^d$ be inside c (shared) clusters. Let r be the maximum L_2 distance of a point to its cluster center and D the minimum distance between different cluster centers, with $r \ll D$. Let $\boldsymbol{C}_{ij} = \|\boldsymbol{x}_{\text{p}i} - \boldsymbol{x}_{\text{q}j}\|_2$ and $\boldsymbol{K}_{ij} = e^{-\boldsymbol{C}_{ij}/\lambda}$. Let each LSH bucket used for the sparse approximation $\boldsymbol{K}^\text{sp}$ cover at least one cluster. Let $\boldsymbol{K}_\text{Nys}$ and $\boldsymbol{K}_\text{LCN}$ both use one landmark at each cluster center. Then the maximum possible error is*

$$\max_{\boldsymbol{x}_{\text{p}i}, \boldsymbol{x}_{\text{q}j}} \boldsymbol{K}_{ij} - \boldsymbol{K}_{\text{Nys},i,j} = 1 - e^{-2r/\lambda} - \mathcal{O}(e^{-2(D-r)/\lambda}), \tag{8}$$

$$\max_{\boldsymbol{x}_{\text{p}i}, \boldsymbol{x}_{\text{q}j}} \boldsymbol{K}_{ij} - \boldsymbol{K}^\text{sp}_{ij} = e^{-(D-2r)/\lambda}, \tag{9}$$

$$\max_{\boldsymbol{x}_{\text{p}i}, \boldsymbol{x}_{\text{q}j}} \boldsymbol{K}_{ij} - \boldsymbol{K}_{\text{LCN},i,j} = e^{-(D-2r)/\lambda}(1 - e^{-2r/\lambda}(2 - e^{-2r/\lambda}) + \mathcal{O}(e^{-2D/\lambda})). \tag{10}$$

This shows that the error in $\boldsymbol{K}_\text{Nys}$ is close to 1 for any reasonably large $\frac{r}{\lambda}$ (which is the maximum error possible). The errors in $\boldsymbol{K}^\text{sp}$ and $\boldsymbol{K}_\text{LCN}$ on the other hand are vanishingly small in this case, since $r \ll D$.

The reduced maximum error directly translates to an improved Sinkhorn approximation. We can show this by adapting the Sinkhorn approximation error bounds due to Altschuler et al. (2019).

Definition 1. *A generalized diagonal is the set of elements $\boldsymbol{M}_{i\sigma(i)}\ \forall i \in \{1, \dots, n\}$ with matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ and permutation σ. A non-negative matrix has support if it has a strictly positive generalized diagonal. It has total support if $\boldsymbol{M} \neq 0$ and all non-zero elements lie on a strictly positive generalized diagonal.*

Theorem 3. *Let $X_\text{p}, X_\text{q} \subseteq \mathbb{R}^d$ have n samples. Denote ρ as the maximum distance between two samples. Let $\tilde{\boldsymbol{K}}$ be a non-negative matrix with support, which approximates the similarity matrix $\boldsymbol{K}$ with $\boldsymbol{K}_{ij} = e^{-\|x_{\text{p}i} - x_{\text{q}j}\|_2/\lambda}$ and $\max_{i,j} |\tilde{\boldsymbol{K}}_{ij} - \boldsymbol{K}_{ij}| \leq \frac{\varepsilon'}{2} e^{-\rho/\lambda}$, where $\varepsilon' = \min(1, \frac{\varepsilon}{50(\rho + \lambda \log \frac{\lambda n}{\varepsilon})})$. When performing the Sinkhorn algorithm until $\|\tilde{\boldsymbol{P}}\boldsymbol{1}_N - \boldsymbol{p}\|_1 + \|\tilde{\boldsymbol{P}}^T\boldsymbol{1}_N - \boldsymbol{q}\|_1 \leq \varepsilon'/2$, the resulting approximate transport plan $\tilde{\boldsymbol{P}}$ and distance $\tilde{d}^\lambda_c$ are bounded by*

$$|d^\lambda_c - \tilde{d}^\lambda_{\tilde{c}}| \leq \varepsilon, \qquad D_\text{KL}(\bar{\boldsymbol{P}} \| \tilde{\boldsymbol{P}}) \leq \varepsilon/\lambda. \tag{11}$$

Convergence rate. We next show that sparse and LCN-Sinkhorn converge as fast as regular Sinkhorn by adapting the convergence bound by Dvurechensky et al. (2018) to account for sparsity.

Theorem 4. *Given a non-negative matrix $\tilde{\boldsymbol{K}} \in \mathbb{R}^{n \times n}$ with support and $\boldsymbol{p} \in \mathbb{R}^n$, $\boldsymbol{q} \in \mathbb{R}^n$. The Sinkhorn algorithm gives a transport plan satisfying $\|\tilde{\boldsymbol{P}}\boldsymbol{1}_N - \boldsymbol{p}\|_1 + \|\tilde{\boldsymbol{P}}^T\boldsymbol{1}_N - \boldsymbol{q}\|_1 \leq \varepsilon$ in iterations*

$$k \leq 2 + \frac{-4 \ln(\min_{i,j}\{\tilde{\boldsymbol{K}}_{ij} | \tilde{\boldsymbol{K}}_{ij} > 0\} \min_{i,j}\{\boldsymbol{p}_i, \boldsymbol{q}_j\})}{\varepsilon}. \tag{12}$$

Backpropagation. Efficient gradient computation is almost as important for modern deep learning models as the algorithm itself. These models usually aim at learning the embeddings in X_p and X_q and therefore need gradients w.r.t. the cost matrix $\boldsymbol{C}$. We can estimate these either via automatic differentiation of the unrolled Sinkhorn iterations or via the analytic solution that assumes exact convergence. Depending on the problem at hand, either the automatic or the analytic estimator will lead to faster overall convergence (Ablin et al., 2020). LCN-Sinkhorn works flawlessly with automatic backpropagation since it only relies on basic linear algebra (except for choosing Nyström landmarks and LSH neighbors, for which we use a simple straight-through estimator (Bengio et al., 2013)). To enable fast analytic backpropagation we provide analytic gradients in Proposition 1. Note that both backpropagation methods have runtime linear in the number of points n and m.

Proposition 1. *In entropy-regularized OT and LCN-Sinkhorn the derivatives of the distances d^λ_c and $d^\lambda_{\text{LCN},c}$ (Eqs.* (1) *and* (5)*) and the optimal transport plan $\bar{\boldsymbol{P}} \in \mathbb{R}^{n \times m}$ w.r.t. the (decomposed) cost matrix $\boldsymbol{C} \in \mathbb{R}^{n \times m}$ with total support are*

$$\frac{\partial d^\lambda_c}{\partial \boldsymbol{C}} = \bar{\boldsymbol{P}}, \tag{13}$$

$$\frac{\partial d^\lambda_{\text{LCN},c}}{\partial \boldsymbol{U}} = -\lambda \bar{\boldsymbol{s}}(\boldsymbol{W}\bar{\boldsymbol{t}})^T, \quad \frac{\partial d^\lambda_{\text{LCN},c}}{\partial \boldsymbol{W}} = -\lambda(\bar{\boldsymbol{s}}^T\boldsymbol{U})^T\bar{\boldsymbol{t}}^T,$$
$$\frac{\partial d^\lambda_{\text{LCN},c}}{\partial \log \boldsymbol{K}^\text{sp}} = -\lambda \bar{\boldsymbol{P}}^\text{sp}, \quad \frac{\partial d^\lambda_{\text{LCN},c}}{\partial \log \boldsymbol{K}^\text{sp}_\text{Nys}} = -\lambda \bar{\boldsymbol{P}}^\text{sp}_\text{Nys}. \tag{14}$$

This allows backpropagation in time $\mathcal{O}((n+m)l^2)$.

6. Graph transport network

Graph distance learning. Predicting similarities or distances between graph-structured objects is useful across a wide range of applications. It can be used to predict the reaction rate between molecules (Houston et al., 2019), or search for similar images (Johnson et al., 2015), similar molecules for drug discovery (Birchall et al., 2006), or similar code for vulnerability detection (Li et al., 2019). We propose the graph transport network (GTN) to evaluate approximate Sinkhorn on a full deep learning model and advance the state of the art on this task.

Graph transport network. GTN uses a Siamese graph neural network (GNN) to embed two graphs independently as *sets* of node embeddings. These sets are then matched using multi-head unbalanced OT. Node embeddings represent the nodes' local environments, so similar neighborhoods will be close in embedding space and matched accordingly. Since Sinkhorn is symmetric and permutation invariant, any identical pair of graphs will thus by construction have a predicted distance of 0 (ignoring the entropy offset).
More precisely, given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with node set $\mathcal{V}$ and edge set $\mathcal{E}$, node attributes $\boldsymbol{x}_i \in \mathbb{R}^{H_\text{x}}$ and (optional) edge attributes $\boldsymbol{e}_{i,j} \in \mathbb{R}^{H_\text{e}}$, with $i, j \in \mathcal{V}$, we update the node embeddings in each GNN layer via

$$\boldsymbol{h}^{(l)}_{\text{self},i} = \sigma(\boldsymbol{W}^{(l)}_\text{node} \boldsymbol{h}^{(l-1)}_i + \boldsymbol{b}^{(l)}), \tag{15}$$

$$\boldsymbol{h}^{(l)}_i = \boldsymbol{h}^{(l)}_{\text{self},i} + \sum_{j \in \mathcal{N}_i} \eta^{(l)}_{i,j} \boldsymbol{h}^{(l)}_{\text{self},j} \mathbf{W}_\text{edge} \boldsymbol{e}_{i,j}, \tag{16}$$

with $\mathcal{N}_i$ denoting the neighborhood of node i, $\boldsymbol{h}^{(0)}_i = \boldsymbol{x}_i$, $\boldsymbol{h}^{(l)}_i \in \mathbb{R}^{H_\text{N}}$ for $l \geq 1$, the bilinear layer $\mathbf{W}_\text{edge} \in \mathbb{R}^{H_\text{N} \times H_\text{N} \times H_\text{e}}$, and the degree normalization $\eta^{(1)}_{i,j} = 1$ and $\eta^{(l)}_{i,j} = 1/\sqrt{\deg_i \deg_j}$ for $l > 1$. This choice of $\eta_{i,j}$ allows our model to handle highly skewed degree distributions while still being able to represent node degrees. We found the choice of non-linearity σ not to be critical and chose a LeakyReLU. We do not use the bilinear layer $\mathbf{W}_\text{edge} \boldsymbol{e}_{i,j}$ if there are no edge attributes. We aggregate each layer's node embeddings to obtain the overall embedding of node i

$$\boldsymbol{h}^\text{GNN}_i = [\boldsymbol{h}^{(1)}_{\text{self},i} \,\|\, \boldsymbol{h}^{(1)}_i \,\|\, \boldsymbol{h}^{(2)}_i \,\|\, \dots \,\|\, \boldsymbol{h}^{(L)}_i]. \tag{17}$$

We then compute the embeddings for matching via $\boldsymbol{h}^\text{final}_i = \text{MLP}(\boldsymbol{h}^\text{GNN}_i)$. Having obtained the embedding sets H^final_1 and H^final_2 of both graphs we use the L_2 distance as a cost function for the Sinkhorn distance. Finally, we calculate the prediction from the Sinkhorn distance via $d = d^\lambda_c w_\text{out} + b_\text{out}$, with learnable w_out and b_out. GTN is trained end-to-end via backpropagation. For small graphs we use the full Sinkhorn distance and scale to large graphs with LCN-Sinkhorn. GTN is more expressive than models that aggregate node embeddings to a single fixed-size embedding but still scales log-linearly in the number of nodes, as opposed to previous approaches which scale quadratically.

Learnable unbalanced OT. Since GTN regularly encounters graphs with disagreeing numbers of nodes it needs to be able to handle cases where $\|\boldsymbol{p}\|_1 \neq \|\boldsymbol{q}\|_1$ or where not all nodes in one graph have a corresponding node in the other, i.e. $\boldsymbol{P}\mathbf{1}_m < \boldsymbol{p}$ or $\boldsymbol{P}^T\mathbf{1}_n < \boldsymbol{q}$. Unbalanced OT allows handling both of these cases (Peyré & Cuturi, 2019), usually by swapping the strict balancing requirements with a uniform divergence loss term on $\boldsymbol{p}$ and $\boldsymbol{q}$ (Frogner et al., 2015; Chizat et al., 2018). However, this *uniformly* penalizes deviations from balanced OT and therefore cannot adaptively ignore parts of the distribution. We propose to improve on this by swapping the cost matrix $\boldsymbol{C}$ with the bipartite matching (BP) matrix (Riesen & Bunke, 2009)

$$\boldsymbol{C}_\text{BP} = \begin{bmatrix} \boldsymbol{C} & \boldsymbol{C}^{(\text{p},\varepsilon)} \\ \boldsymbol{C}^{(\varepsilon,\text{q})} & \boldsymbol{C}^{(\varepsilon,\varepsilon)} \end{bmatrix}, \quad \boldsymbol{C}^{(\text{p},\varepsilon)}_{ij} = \begin{cases} c_{i,\varepsilon} & i = j \\ \infty & i \neq j \end{cases},$$
$$\boldsymbol{C}^{(\varepsilon,\text{q})}_{ij} = \begin{cases} c_{\varepsilon,j} & i = j \\ \infty & i \neq j \end{cases}, \quad \boldsymbol{C}^{(\varepsilon,\varepsilon)}_{ij} = 0, \tag{18}$$

and obtain the deletion cost $c_{i,\varepsilon}$ and $c_{\varepsilon,j}$ from the input sets X_p and X_q. Using the BP matrix only adds minor computational overhead since we just need to save the diagonals $\boldsymbol{c}_{\text{p},\varepsilon}$ and $\boldsymbol{c}_{\varepsilon,\text{q}}$ of $\boldsymbol{C}_{\text{p},\varepsilon}$ and $\boldsymbol{C}_{\varepsilon,\text{q}}$. We can then use $\boldsymbol{C}_\text{BP}$ in the Sinkhorn algorithm (Eq. (2)) via

$$\boldsymbol{K}_\text{BP}\boldsymbol{t} = \begin{bmatrix} \boldsymbol{K}\hat{\boldsymbol{t}} + \boldsymbol{c}_{\text{p},\varepsilon} \odot \check{\boldsymbol{t}} \\ \boldsymbol{c}_{\varepsilon,\text{q}} \odot \hat{\boldsymbol{t}} + \mathbf{1}_n^T \check{\boldsymbol{t}} \end{bmatrix}, \; \boldsymbol{K}_\text{BP}^T\boldsymbol{s} = \begin{bmatrix} \boldsymbol{K}^T\hat{\boldsymbol{s}} + \boldsymbol{c}_{\varepsilon,\text{q}} \odot \check{\boldsymbol{s}} \\ \boldsymbol{c}_{\text{p},\varepsilon} \odot \hat{\boldsymbol{s}} + \mathbf{1}_m^T \check{\boldsymbol{s}} \end{bmatrix}, \tag{19}$$

where $\hat{\boldsymbol{t}}$ denotes the upper and $\check{\boldsymbol{t}}$ the lower part of the vector $\boldsymbol{t}$. To calculate d^λ_c we can decompose the transport plan $\boldsymbol{P}_\text{BP}$ in the same way as $\boldsymbol{C}_\text{BP}$, with a single scalar for $\boldsymbol{P}_{\varepsilon,\varepsilon}$. For GTN we obtain the deletion cost via $c_{i,\varepsilon} = \|\boldsymbol{\alpha} \odot \boldsymbol{x}_{\text{p}i}\|_2$, with a learnable vector $\boldsymbol{\alpha} \in \mathbb{R}^d$.

Multi-head OT. Inspired by attention models (Vaswani et al., 2017) and multiscale kernels (Bermanis et al., 2013) we further improve GTN by using multiple OT heads. Using K heads means that we calculate K separate sets of embeddings representing the same pair of objects by using separate linear layers, i.e. $\boldsymbol{h}^\text{final}_{k,i} = \boldsymbol{W}^{(k)} \boldsymbol{h}^\text{GNN}_i$ for head k. We then calculate OT in parallel for these sets using a series of regularization parameters $\lambda_k = 2^{k-K/2}\lambda$. This yields a set of distances $\boldsymbol{d}^\lambda_c \in \mathbb{R}^K$. We obtain the final prediction via $d = \text{MLP}(\boldsymbol{d}^\lambda_c)$. Both learnable unbalanced OT and multi-head OT might be of independent interest.

7. Related work

Hierarchical kernel approximation. These methods usually hierarchically decompose the kernel matrix into separate blocks and use low-rank or core-diagonal approximations for each block (Si et al., 2017; Ding et al., 2017). This idea is similar in spirit to LCN, but LCN boils it down to its essence by using one purely global part and a fine-grained LSH method to obtain one exact and purely local part.

Log-linear optimal transport. For an overview of optimal transport and its foundations see Peyré & Cuturi (2019). On low-dimensional grids and surfaces OT can be solved using dynamical OT (Papadakis et al., 2014; Solomon et al., 2014), convolutions (Solomon et al., 2015), or embedding/hashing schemes (Indyk & Thaper, 2003; Andoni et al., 2008). In higher dimensions we can use tree-based algorithms (Backurs et al., 2020) or hashing schemes (Charikar, 2002), which are however limited to a previously fixed set of points X_p, X_q, on which only the distributions $\boldsymbol{p}$ and $\boldsymbol{q}$ change. Another approach are sliced Wasserstein distances (Rabin et al., 2011). However, they do not provide a transport plan, require the L_2 distance as a cost function, and are either unstable in convergence or prohibitively expensive for high dimensions ($\mathcal{O}(nd^3)$) (Meng et al., 2019). For high-dimensional sets that change dynamically (e.g. during training) one method of achieving log-linear runtime is a multiscale approximation of entropy-regularized OT (Schmitzer, 2019; Gerber & Maggioni, 2017). Tenetov et al. (2018) recently proposed using a low-rank approximation of the Sinkhorn similarity matrix obtained via a semidiscrete approximation of the Euclidean distance. Altschuler et al. (2019) improved upon this approach by using the Nyström method for the approximation. However, these approaches still struggle with high-dimensional real-world problems, as we will show in Sec. 8.

Accelerating Sinkhorn. Another line of work has been pursuing accelerating entropy-regularized OT without changing its computational complexity w.r.t. the number of points. Original Sinkhorn requires $\mathcal{O}(1/\varepsilon^2)$ iterations, but Dvurechensky et al. (2018) and Jambulapati et al. (2019) recently proposed algorithms that reduce the computational complexity to $\mathcal{O}(\min(n^{9/4}/\varepsilon, n^2/\varepsilon^2))$ and $\mathcal{O}(n^2/\varepsilon)$, respectively. Mensch & Peyré (2020) proposed an online Sinkhorn algorithm to significantly reduce its memory cost. Alaya et al. (2019) proposed reducing the size of the Sinkhorn problem by screening out neglectable components, which allows for approximation guarantees. Genevay et al. (2016) proposed using a stochastic optimization scheme instead of Sinkhorn iterations. Essid & Solomon (2018) and Blondel et al. (2018) proposed alternative regularizations to obtain OT problems with similar runtimes as the Sinkhorn algorithm. This work is largely orthogonal to ours.

Embedding alignment. For an overview of cross-lingual word embedding models see Ruder et al. (2019). Unsupervised word embedding alignment was proposed by Conneau et al. (2018), with subsequent advances by Alvarez-Melis & Jaakkola (2018); Grave et al. (2019); Joulin et al. (2018).

Graph matching and distance learning. Graph neural networks (GNNs) have recently been successful on a wide variety of graph-based tasks (Kipf & Welling, 2017; Gasteiger et al., 2019; 2020; Zambaldi et al., 2019). GNN-based approaches for graph matching and graph distance learning either rely on a single fixed-dimensional graph embedding (Bai et al., 2019; Li et al., 2019), or only use attention or some other strongly simplified variant of optimal transport (Bai et al., 2019; Riba et al., 2018; Li et al., 2019). Others break permutation invariance and are thus ill-suited for this task (Ktena et al., 2017; Bai et al., 2018). So far only approaches using a single graph embedding allow faster than quadratic scaling in the number of nodes. Compared to the Sinkhorn-based image model proposed by Wang et al. (2019) GTN uses no CNN or cross-graph attention, but an enhanced GNN and embedding aggregation scheme. OT has recently been proposed for graph kernels (Maretic et al., 2019; Vayer et al., 2019), which can (to some extent) be used for graph matching, but not for distance learning.

8. Experiments

Approximating Sinkhorn. We start by directly investigating different Sinkhorn approximations. To do so we compute entropy-regularized OT on (i) pairs of 10^4 word embeddings from Conneau et al. (2018), which we preprocess with Wasserstein Procrustes alignment in order to obtain both close and distant neighbors, (ii) the armadillo and dragon point clouds from the Stanford 3D Scanning Repository (Stanford, 2014) (with 10^4 randomly subsampled points), and (iii) pairs of 10^4 data points that are uniformly distributed in the d-ball ($d = 16$). We let every method use the same total number of 40 average neighbors and landmarks (LCN uses 20 each) and set $\lambda = 0.05$ (as e.g. in Grave et al. (2019)). Besides the Sinkhorn distance we measure transport plan approximation quality by (a) calculating the Pearson correlation coefficient (PCC) between all entries in the approximated plan and the true $\bar{\boldsymbol{P}}$ and (b) comparing the sets of 0.1 % largest entries in the approximated and true $\bar{\boldsymbol{P}}$ using the Jaccard similarity (intersection over union, IoU). Note that usually the OT plan is more important than the distance, since it determines the training gradient and tasks like embedding alignment are exclusively based on the OT plan. In all figures the error bars denote standard deviation across 5 runs, which is often too small to be visible.

Table 1 shows that for word embeddings both sparse Sinkhorn, LCN-Sinkhorn and factored OT (Forrow et al., 2019) obtain distances that are significantly closer to the true d_c^λ than Multiscale OT and Nyström-Sinkhorn. Furthermore, the transport plan computed by sparse Sinkhorn and LCN-Sinkhorn show both a PCC and IoU that are around twice as high as Multiscale OT, while Nyström-Sinkhorn and

Table 1: Mean and standard deviation of relative Sinkhorn distance error, IoU of top 0.1 % and correlation coefficient (PCC) of OT plan entries across 5 runs. Sparse Sinkhorn and LCN-Sinkhorn achieve the best approximation in all 3 measures.

	EN-DE			EN-ES			3D point cloud			Uniform in d-ball ($d\!=\!16$)		
	Rel. err. d_c^λ	PCC	IoU	Rel. err. d_c^λ	PCC	IoU	Rel. err. d_c^λ	PCC	IoU	Rel. err. d_c^λ	PCC	IoU
Factored OT	0.318 ±0.001	0.044 ±0.001	0.019 ±0.002	0.332 ±0.001	0.037 ±0.002	0.026 ±0.005	6.309 ±0.004	0.352 ±0.001	0.004 ±0.001	1.796 ±0.001	0.096 ±0.001	0.029 ±0.000
Multiscale OT	0.634 ±0.011	0.308 ±0.014	0.123 ±0.005	0.645 ±0.014	0.321 ±0.006	0.125 ±0.012	**0.24 ±0.07**	0.427 ±0.008	0.172 ±0.011	**0.03 ±0.02**	0.091 ±0.005	0.021 ±0.001
Nyström Skh.	1.183 ±0.005	0.077 ±0.001	0.045 ±0.005	1.175 ±0.018	0.068 ±0.001	0.048 ±0.006	1.89 ±0.07	**0.559 ±0.009**	0.126 ±0.014	1.837 ±0.006	0.073 ±0.000	0.018 ±0.000
Sparse Skh.	**0.233 ±0.002**	0.552 ±0.004	0.102 ±0.001	**0.217 ±0.001**	0.623 ±0.004	0.102 ±0.001	0.593 ±0.015	0.44 ±0.03	0.187 ±0.014	0.241 ±0.002	**0.341 ±0.004**	**0.090 ±0.001**
LCN-Sinkhorn	0.406 ±0.015	**0.673 ±0.012**	**0.197 ±0.007**	0.368 ±0.012	**0.736 ±0.003**	**0.201 ±0.003**	1.91 ±0.28	**0.564 ±0.008**	**0.195 ±0.013**	0.435 ±0.009	0.328 ±0.006	0.079 ±0.001

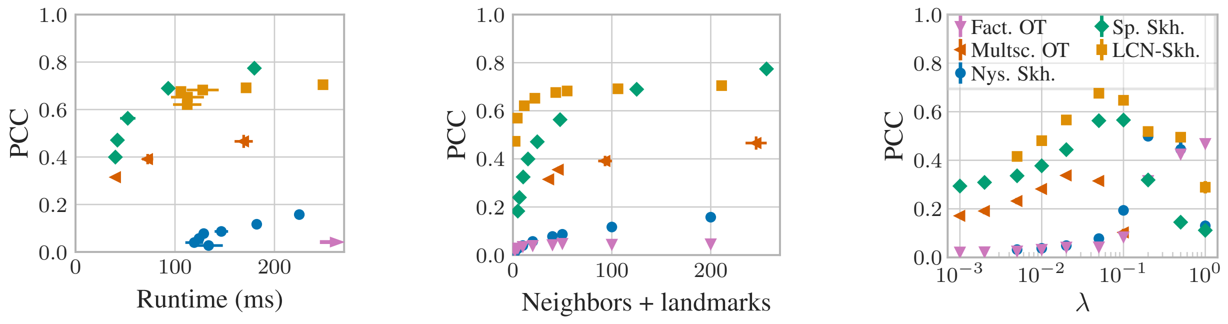

Figure 2: OT plan approximation quality for EN-DE, via PCC. **Left:** Sparse Sinkhorn offers the best tradeoff with runtime, with LCN-Sinkhorn closely behind. **Center:** LCN-Sinkhorn achieves the best approximation for low and sparse Sinkhorn for high numbers of neighbors/landmarks. **Right:** Sparse Sinkhorn performs best for low, LCN-Sinkhorn for moderate and factored OT for very high entropy regularization λ. The arrow indicates factored OT results far outside the range.

factored OT exhibit almost no correlation. LCN-Sinkhorn performs especially well in this regard. This is also evident in Fig. 1, which shows how the $10^4 \times 10^4$ approximated OT plan entries compared to the true Sinkhorn values. Multiscale OT shows the best distance approximation on 3D point clouds and random high-dimensional data. However, sparse Sinkhorn and LCN-Sinkhorn remain the best OT plan approximations, especially in high dimensions.

Fig. 2 shows that sparse Sinkhorn offers the best trade-off between runtime and OT plan quality. Factored OT exhibits a runtime 2 to 10 times longer than the competition due to its iterative refinement scheme. LCN-Sinkhorn performs best for use cases with constrained memory (few neighbors/landmarks). The number of neighbors and landmarks directly determines memory usage and is linearly proportional to the runtime (see App. K). Fig. 2 furthermore shows that sparse Sinkhorn performs best for low regularizations, where LCN-Sinkhorn fails due to the Nyström part going out of bounds. Nyström Sinkhorn performs best at high values and LCN-Sinkhorn always performs better than both (as long as it can be calculated). Interestingly, all approximations except factored OT seem to fail at high λ. We defer analogously discussing the distance approximation to App. L. All approximations scale linearly both in the number of neighbors/landmarks and dataset size, as shown in App. K. Overall, we see that sparse Sinkhorn and LCN-Sinkhorn yield significant improvements over previous approximations. However, do these improvements also translate to better performance on downstream tasks?

Embedding alignment. Embedding alignment is the task of finding the orthogonal matrix $\boldsymbol{R} \in \mathbb{R}^{d\times d}$ that best aligns the vectors from two different embedding spaces, which is e.g. useful for unsupervised word translation. We use the experimental setup established by Conneau et al. (2018) by migrating Grave et al. (2019)'s implementation to PyTorch. The only change we make is using the full set of 20 000 word embeddings and training for 300 steps, while reducing the learning rate by half every 100 steps. We do not change *any* other hyperparameters and do not use unbalanced OT. After training we match pairs via cross-domain similarity local scaling (CSLS) (Conneau et al., 2018). We use 10 Sinkhorn iterations, 40 neighbors on average for sparse Sinkhorn, and

Table 2: Accuracy and standard deviation across 5 runs for unsupervised word embedding alignment with Wasserstein Procrustes. LCN-Sinkhorn improves upon the original by 3.1 pp. before and 2.0 pp. after iterative CSLS refinement. *Migrated and re-run on GPU via PyTorch

	Time (s)	EN-ES	ES-EN	EN-FR	FR-EN	EN-DE	DE-EN	EN-RU	RU-EN	**Avg.**
Original*	268	79.2±0.2	78.8±2.8	81.0±0.3	79.4±0.9	71.7±0.2	65.7±3.4	36.3±1.1	51.1±1.1	67.9
Full Sinkhorn	402	81.1	**82.0**	81.2	81.3	**74.1**	70.7	37.3	53.5	70.1
Multiscale OT	88.2	24 ±31	74.7±3.3	27 ±32	6.3±4.4	36 ±10	47 ±21	0.0	0.2±0.1	26.8
Nyström Skh.	102	64.4±1.0	59.3±1.2	64.1±1.6	56.8±4.0	54.1±0.6	47.1±3.5	14.1±1.2	22.5±2.4	47.8
Sparse Skh.	49.2	80.2±0.2	81.7±0.4	80.9±0.3	80.1±0.2	72.1±0.6	65.1±1.7	35.5±0.6	51.5±0.4	68.4
LCN-Sinkhorn	86.8	**81.8±0.2**	**81.3±1.8**	**82.0±0.4**	**82.1±0.3**	73.6±0.2	**71.3±0.9**	**41.0±0.8**	**55.1±1.4**	**71.0**
Original* + ref.	268+81	83.0±0.3	**82.0±2.5**	**83.8±0.1**	83.0±0.4	**77.3±0.3**	69.7±4.3	46.2±1.0	54.0±1.1	72.4
LCN-Skh. + ref.	86.8+81	**83.5±0.2**	**83.1±1.3**	**83.8±0.2**	**83.6±0.1**	**77.2±0.3**	**72.8±0.7**	**51.8±2.6**	**59.2±1.9**	**74.4**

20 neighbors and landmarks for LCN-Sinkhorn (for details see App. H). We allow both multiscale OT and Nyström Sinkhorn to use as many landmarks and neighbors as can fit into GPU memory and finetune both methods.

Table 2 shows that using full Sinkhorn yields a significant improvement in accuracy on this task compared to the original approach of performing Sinkhorn on randomly sampled subsets of embeddings (Grave et al., 2019). LCN-Sinkhorn even outperforms the *full* version in most cases, which is likely due to regularization effects from the approximation. It also runs 4.6x faster than full Sinkhorn and 3.1x faster than the original scheme, while using 88 % and 44 % less memory, respectively. Sparse Sinkhorn runs 1.8x faster than LCN-Sinkhorn but cannot match its accuracy. LCN-Sinkhorn still outcompetes the original method after refining the embeddings with iterative local CSLS (Conneau et al., 2018). Both multiscale OT and Nyström Sinkhorn fail at this task, despite their larger computational budget. This shows that the improvements achieved by sparse Sinkhorn and LCN-Sinkhorn have an even larger impact in practice.

Graph distance regression. The graph edit distance (GED) is useful for various tasks such as image retrieval (Xiao et al., 2008) or fingerprint matching (Neuhaus & Bunke, 2004), but its computation is NP-complete (Bunke & Shearer, 1998). For large graphs we therefore need an effective approximation. We use the Linux dataset by Bai et al. (2019) and generate 2 new datasets by computing the exact GED using the method by Lerouge et al. (2017) on small graphs ($\leq$ 30 nodes) from the AIDS dataset (Riesen & Bunke, 2008) and a set of preferential attachment graphs. We compare GTN to 3 state-of-the-art baselines: SiameseMPNN (Riba et al., 2018), SimGNN (Bai et al., 2019), and the Graph Matching Network (GMN) (Li et al., 2019). We tune the hyperparameters of all baselines and GTN via grid search. For more details see App. H to J.

We first test GTN and the proposed OT enhancements. Table 3 shows that GTN improves upon other models by 20 % with a single head and by 48 % with 8 OT heads. Its performance breaks down with regular unbalanced (using KL-divergence loss for the marginals) and balanced OT, showing the importance of learnable unbalanced OT.

Table 3: RMSE for GED regression across 3 runs and the targets' standard deviation σ. GTN outperforms previous models by 48 %.

	Linux	AIDS30	Pref. att.
σ	0.184	16.2	48.3
SiamMPNN	0.090±0.007	13.8±0.3	12.1±0.6
SimGNN	0.039	4.5±0.3	8.3±1.4
GMN	0.015	10.3±0.6	7.8±0.3
GTN, 1 head	0.022±0.001	3.7±0.1	4.5±0.3
8 OT heads	**0.012±0.001**	**3.2±0.1**	**3.5±0.2**
Unbalanced OT	0.033±0.002	15.7±0.5	9.7±0.9
Balanced OT	0.034±0.001	15.3±0.1	27.4±0.9

Having established GTN as a state-of-the-art model we next ask whether we can sustain its performance when using approximate OT. For this we additionally generate a set of larger graphs with around 200 nodes and use the Pyramid matching (PM) kernel (Nikolentzos et al., 2017) as the prediction target, since these graphs are too large to compute the GED. See App. J for hyperparameter details. Table 4 shows that both sparse Sinkhorn and the multiscale method using 4 (expected) neighbors fail at this task, demonstrating that the low-rank approximation in LCN has a crucial stabilizing effect during training. Nyström Sinkhorn with 4 landmarks performs surprisingly well on the AIDS30 dataset, suggesting an overall low-rank structure with Nyström acting as regularization. However, it does not perform as well on the other two datasets. Using LCN-Sinkhorn with 2 neighbors and landmarks works well on all three datasets, with an RMSE increased by only 10 % compared to full GTN. App. K furthermore shows that GTN with LCN-Sinkhorn indeed scales linearly in the number of nodes across multiple

Table 4: RMSE for graph distance regression across 3 runs and the targets' standard deviation σ. Using LCN-Sinkhorn with GTN increases the error by only 10 % and allows log-linear scaling.

	GED		PM $[10^{-2}]$
	AIDS30	Pref. att.	Pref. att. 200
σ	16.2	48.3	10.2
Full Sinkhorn	3.7±0.1	4.5±0.3	1.27±0.06
Nyström Skh.	**3.6±0.3**	6.2±0.6	2.43±0.07
Multiscale OT	11.2±0.3	27.4±5.4	6.71±0.44
Sparse Skh.	44 ±30	40.7±8.1	7.57±1.09
LCN-Skh.	4.0±0.1	**5.1±0.4**	**1.41±0.15**

orders of magnitude. This model thus enables graph matching and distance learning on graphs that are considered large even for simple node-level tasks (20 000 nodes).

9. Conclusion

Locality-sensitive hashing (LSH) and the novel locally corrected Nyström (LCN) method enable fast and accurate approximations of entropy-regularized OT with log-linear runtime: Sparse Sinkhorn and LCN-Sinkhorn. The graph transport network (GTN) is one example for such a model, which can be substantially improved with learnable unbalanced OT and multi-head OT. It sets the new state of the art for graph distance learning while still scaling log-linearly with graph size. These contributions enable new applications and models that are both faster and more accurate, since they can sidestep workarounds such as pooling.

ACKNOWLEDGMENTS

We would like to thank Johannes Pitz, Oleksandr Shchur, Aleksandar Bojchevski, and Daniel Zügner for their support, suggestions, and feedback during the process of creating this paper. This research was supported by the Deutsche Forschungsgemeinschaft (DFG) through the Emmy Noether grant GU 1409/2-1 and the TUM International Graduate School of Science and Engineering (IGSSE), GSC 81.

References

Pierre Ablin, Gabriel Peyré, and Thomas Moreau. Super-efficiency of automatic differentiation for functions defined as a minimum. In *ICML*, 2020.

Mokhtar Z. Alaya, Maxime Berar, Gilles Gasso, and Alain Rakotomamonjy. Screening Sinkhorn Algorithm for Regularized Optimal Transport. In *NeurIPS*, 2019.

Jason Altschuler, Francis Bach, Alessandro Rudi, and Jonathan Niles-Weed. Massively scalable Sinkhorn distances via the Nyström method. In *NeurIPS*, 2019.

David Alvarez-Melis and Tommi S. Jaakkola. Gromov-Wasserstein Alignment of Word Embedding Spaces. In *EMNLP*, 2018.

Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth mover distance over high-dimensional spaces. In *ACM-SIAM symposium on Discrete algorithms (SODA)*, 2008.

Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya P. Razenshteyn, and Ludwig Schmidt. Practical and Optimal LSH for Angular Distance. In *NeurIPS*, 2015.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein Generative Adversarial Networks. In *ICML*, 2017.

Arturs Backurs, Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Scalable Nearest Neighbor Search for Optimal Transport. In *ICML*, 2020.

Yunsheng Bai, Hao Ding, Yizhou Sun, and Wei Wang. Convolutional Set Matching for Graph Similarity. In *Relational Representation Learning Workshop, NeurIPS*, 2018.

Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. SimGNN: A Neural Network Approach to Fast Graph Similarity Computation. In *WSDM*, 2019.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv*, 1308.3432, 2013.

Christian Berg, Jens Peter Reus Christensen, and Paul Ressel. *Harmonic Analysis on Semigroups*. Number 100 in Graduate Texts in Mathematics. 1984.

Amit Bermanis, Amir Averbuch, and Ronald R. Coifman. Multiscale data sampling and function extension. *Applied and Computational Harmonic Analysis*, 34(1):15–29, 2013.

Kristian Birchall, Valerie J. Gillet, Gavin Harper, and Stephen D. Pickett. Training Similarity Measures for Specific Activities: Application to Reduced Graphs. *Journal of Chemical Information and Modeling*, 46(2):577–586, 2006.

Mathieu Blondel, Vivien Seguy, and Antoine Rolet. Smooth and Sparse Optimal Transport. In *AISTATS*, 2018.

Olivier Bousquet, Sylvain Gelly, Ilya Tolstikhin, Carl-Johann Simon-Gabriel, and Bernhard Schoelkopf. From optimal transport to generative modeling: the VEGAN cookbook. *arXiv*, 1705.07642, 2017.

Horst Bunke and Kim Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3):255–259, 1998.

Lawrence F. Canino, John J. Ottusch, Mark A. Stalzer, John L. Visher, and Stephen M. Wandzura. Numerical Solution of the Helmholtz Equation in 2D and 3D Using a High-Order Nyström Discretization. *Journal of Computational Physics*, 146(2):627–663, 1998.

Moses Charikar. Similarity estimation techniques from rounding algorithms. In *ACM symposium on Theory of computing (STOC)*, 2002.

Lénaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced optimal transport problems. *Mathematics of Computation*, 87(314):2563–2609, 2018.

Henry Cohn. A Conceptual Breakthrough in Sphere Packing. *Notices of the American Mathematical Society*, 64(02): 102–115, 2017.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. In *ICLR*, 2018.

Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *NeurIPS*, 2017.

Marco Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *NeurIPS*, 2013.

Yi Ding, Risi Kondor, and Jonathan Eskreis-Winkler. Multiresolution Kernel Approximation for Gaussian Process Regression. In *NeurIPS*, 2017.

Pavel E. Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational Optimal Transport: Complexity by Accelerated Gradient Descent Is Better Than by Sinkhorn's Algorithm. In *ICML*, 2018.

Montacer Essid and Justin Solomon. Quadratically Regularized Optimal Transport on Graphs. *SIAM Journal on Scientific Computing*, 40(4):A1961–A1986, 2018.

Matthias Fey and Jan E. Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In *Workshop on Representation Learning on Graphs and Manifolds, ICLR*, 2019.

Aden Forrow, Jan-Christian Hütter, Mor Nitzan, Philippe Rigollet, Geoffrey Schiebinger, and Jonathan Weed. Statistical Optimal Transport via Factored Couplings. In *AISTATS*, 2019.

Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya-Polo, and Tomaso A. Poggio. Learning with a Wasserstein Loss. In *NeurIPS*, 2015.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then Propagate: Graph Neural Networks Meet Personalized PageRank. In *ICLR*, 2019.

Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional Message Passing for Molecular Graphs. In *ICLR*, 2020.

Aude Genevay, Marco Cuturi, Gabriel Peyré, and Francis R. Bach. Stochastic Optimization for Large-scale Optimal Transport. In *NeurIPS*, 2016.

Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning Generative Models with Sinkhorn Divergences. In *AISTATS*, 2018.

Samuel Gerber and Mauro Maggioni. Multiscale Strategies for Computing Optimal Transport. *J. Mach. Learn. Res.*, 18:72:1–72:32, 2017.

Edouard Grave, Armand Joulin, and Quentin Berthet. Unsupervised Alignment of Embeddings with Wasserstein Procrustes. In *AISTATS*, 2019.

Paul L. Houston, Apurba Nandi, and Joel M. Bowman. A Machine Learning Approach for Prediction of Rate Constants. *The Journal of Physical Chemistry Letters*, 10 (17):5250–5258, 2019.

Piotr Indyk and Nitin Thaper. Fast image retrieval via embeddings. In *International Workshop on Statistical and Computational Theories of Vision, ICCV*, 2003.

Arun Jambulapati, Aaron Sidford, and Kevin Tian. A Direct tilde{O}(1/epsilon) Iteration Parallel Algorithm for Optimal Transport. In *NeurIPS*, 2019.

Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. Image retrieval using scene graphs. In *CVPR*, 2015.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion. In *EMNLP*, 2018.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

Sofia Ira Ktena, Sarah Parisot, Enzo Ferrante, Martin Rajchl, Matthew Lee, Ben Glocker, and Daniel Rueckert. Distance Metric Learning Using Graph Convolutional Networks: Application to Functional Brain Networks. In *MICCAI*, 2017.

Julien Lerouge, Zeina Abu-Aisheh, Romain Raveaux, Pierre Héroux, and Sébastien Adam. New binary linear programming formulation to compute the graph edit distance. *Pattern Recognit.*, 72:254–265, 2017.

Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph Matching Networks for Learning the Similarity of Graph Structured Objects. In *ICML*, 2019.

Hermina Petric Maretic, Mireille El Gheche, Giovanni Chierchia, and Pascal Frossard. GOT: An Optimal Transport framework for Graph comparison. In *NeurIPS*, 2019.

Cheng Meng, Yuan Ke, Jingyi Zhang, Mengrui Zhang, Wenxuan Zhong, and Ping Ma. Large-scale optimal transport map estimation using projection pursuit. In *NeurIPS*, 2019.

Arthur Mensch and Gabriel Peyré. Online Sinkhorn: Optimal Transport distances from sample streams. In *NeurIPS*, 2020.

Cameron Musco and Christopher Musco. Recursive Sampling for the Nystrom Method. In *NeurIPS*, 2017.

Michel Neuhaus and Horst Bunke. An Error-Tolerant Approximate Matching Algorithm for Attributed Planar Graphs and Its Application to Fingerprint Classification. In *Structural, Syntactic, and Statistical Pattern Recognition*, 2004.

Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Matching Node Embeddings for Graph Similarity. In *AAAI*, 2017.

David Nistér and Henrik Stewénius. Scalable Recognition with a Vocabulary Tree. In *CVPR*, 2006.

Nicolas Papadakis, Gabriel Peyré, and Édouard Oudet. Optimal Transport with Proximal Splitting. *SIAM J. Imaging Sciences*, 7(1):212–238, 2014.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019.

Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognit. Lett.*, 31 (11):1348–1358, 2010.

Allon G Percus and Olivier C Martin. Scaling Universalities ofkth-Nearest Neighbor Distances on Closed Manifolds. *Advances in Applied Mathematics*, 21(3):424–436, 1998.

Gabriel Peyré and Marco Cuturi. Computational Optimal Transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein Barycenter and Its Application to Texture Mixing. In *Scale Space and Variational Methods in Computer Vision (SSVM)*, 2011.

Pau Riba, Andreas Fischer, Josep Lladós, and Alicia Fornés. Learning Graph Distances with Message Passing Neural Networks. In *ICPR*, 2018.

Kaspar Riesen and Horst Bunke. IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning. In *Structural, Syntactic, and Statistical Pattern Recognition*, 2008.

Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vis. Comput.*, 27(7):950–959, 2009.

Sebastian Ruder, Ivan Vulic, and Anders Søgaard. A Survey of Cross-lingual Word Embedding Models. *J. Artif. Intell. Res.*, 65:569–631, 2019.

Bernhard Schmitzer. Stabilized Sparse Scaling Algorithms for Entropy Regularized Transport Problems. *SIAM Journal on Scientific Computing*, 41(3):A1443–A1481, 2019.

Anshumali Shrivastava and Ping Li. Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). In *NeurIPS*, 2014.

Si Si, Cho-Jui Hsieh, and Inderjit S. Dhillon. Memory Efficient Kernel Approximation. *Journal of Machine Learning Research*, 18(20):1–32, 2017.

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

Justin Solomon, Raif M. Rustamov, Leonidas J. Guibas, and Adrian Butscher. Earth mover's distances on discrete surfaces. *ACM Trans. Graph.*, 33(4):67:1–67:12, 2014.

Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas J. Guibas. Convolutional wasserstein distances: efficient optimal transportation on geometric domains. *ACM Trans. Graph.*, 34(4):66:1–66:11, 2015.

Computer Graphics Laboratory Stanford. The Stanford 3D Scanning Repository, 2014. URL `http://graphics.stanford.edu/data/3Dscanrep/`.

Robert E. Tarjan. Dynamic trees as search trees via euler tours, applied to the network simplex algorithm. *Mathematical Programming*, 78(2):169–177, 1997.

Evgeny Tenetov, Gershon Wolansky, and Ron Kimmel. Fast Entropic Regularized Optimal Transport Using Semidiscrete Cost Approximation. *SIAM J. Sci. Comput.*, 40(5): A3400–A3422, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *NeurIPS*, 2017.

Titouan Vayer, Nicolas Courty, Romain Tavenard, Laetitia Chapel, and Rémi Flamary. Optimal Transport for structured data with application on graphs. In *ICML*, 2019.

Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for Similarity Search: A Survey. *arXiv*, 1408.2927, 2014.

Runzhong Wang, Junchi Yan, and Xiaokang Yang. Learning Combinatorial Embedding Networks for Deep Graph Matching. In *ICCV*, 2019.

Christopher K. I. Williams and Matthias Seeger. Using the Nyström Method to Speed Up Kernel Machines. In *NeurIPS*, 2001.

Bing Xiao, Xinbo Gao, Dacheng Tao, and Xuelong Li. HMM-based graph edit distance for image indexing. *Int. J. Imaging Systems and Technology*, 18(2-3):209–218, 2008.

Vinícius Flores Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David P. Reichert, Timothy P. Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter W. Battaglia. Deep reinforcement learning with relational inductive biases. In *ICLR*, 2019.

Kai Zhang, Ivor W. Tsang, and James T. Kwok. Improved Nyström low-rank approximation and error analysis. In *ICML*, 2008.