This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Long-term Visual Map Sparsification with Heterogeneous GNN

Ming-Fang Chang[1], Yipu Zhao[2], Rajvi Shah[2], Jakob J. Engel[2], Michael Kaess[1], and Simon Lucey[3]

[1]Carnegie Mellon University [2]Meta Reality Labs Research [3]The University of Adelaide

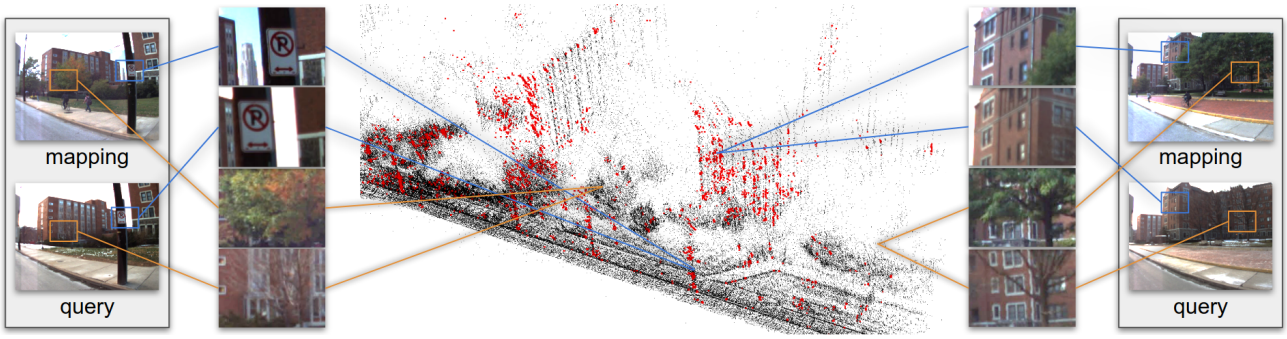


Figure 1. Given a map built from SfM, our proposed approach leverages GNNs and is able to identifies map points on stable structures (red points and blue squares), while discarding points that are prone to seasonal change, such as tree foliage (black points and orange squares).

Abstract

We address the problem of map sparsification for long-term visual localization. For map sparsification, a commonly employed assumption is that the pre-build map and the later captured localization query are consistent. However, this assumption can be easily violated in the dynamic world. Additionally, the map size grows as new data accumulate through time, causing large data overhead in the long term. In this paper, we aim to overcome the environmental changes and reduce the map size at the same time by selecting points that are valuable to future localization. Inspired by the recent progress in Graph Neural Network (GNN), we propose the first work that models SfM maps as heterogeneous graphs and predicts 3D point importance scores with a GNN, which enables us to directly exploit the rich information in the SfM map graph. Two novel supervisions are proposed: 1) a data-fitting term for selecting valuable points to future localization based on training queries; 2) a K-Cover term for selecting sparse points with full-map coverage. The experiments show that our method selected map points on stable and widely visible structures and outperformed baselines in localization performance.

1. Introduction

In long-term visual localization, a common strategy is to build and accumulate maps from the captured image streams, and then localize new incoming queries by matching against the accumulated map. In the presence of environmental changes, the accumulated map contains an increasing number of points and many of which are outdated. This will affect both the computational cost and the performance of localization in the long run. Therefore, the ability to identify and remove these invalid points is important for many applications that target dynamic environments, such as autonomous driving, field robotics, and Augmented Reality. Additionally, for devices with limited on-board memory, it enables keeping a compact map that only contains the most valuable information for future localization queries.

Existing works on map sparsification mostly fall into the category of subset selection, i.e., treating the 3D map as an over-sampled representation of a static world and aiming to select the most valuable point subset from them. The selection of point subset is typically formulated as a K-Cover problem. Assuming the map keyframes cover all the possible camera positions, the K-Cover algorithm encourages each keyframe in the map to observe K points under a total point number constraint [7, 14, 15, 17]. These methods are purely based on the historical data stored in the map, therefore lacking the ability to identify points invalidated due to environmental changes. When the environment changes, the map can only be updated by collecting new query data over the whole mapped area and solve the K-Cover problem again with the new query data, which is inefficient and expensive. Apart from sparsifying a 3D map, there are some works on selecting 2D key points, e.g., by predicting the

persistency [8] or the repeatability [6] of visual features. However, the predictors proposed only take instantaneous measurements (such as local image patches) and not exploit the full context stored in the accumulated map.

Recently, Graph Neural Networks (GNN) have shown promising results with data with different structures, such as citation graphs [29], local feature matching [20] and visibility graphs [23]. In this work, we exploit this flexibility of GNNs to formulate map sparsification as a learning problem and overcome the limitations of previous methods. First, by modeling the SfM map as a graph, we can directly employ the context-rich SfM map as the GNN input instead of instantaneous measurements. Second, in contrast to the K-Cover based methods that requires full-extent new queries to update the map, we are able to train a GNN with only partial queries and use it to sparsify the whole map. A main improvement from previous methods is the ability to incorporate the partial new data and select important points from the whole map according to the partial new data, as there is no trivial way for the baseline methods to do this without collecting new data that covers the whole mapped area.

To this end, we propose the first work that extracts features from SfM maps with a heterogeneous GNN. We first represent the SfM map with a heterogeneous graph, where 3D points, 2D key points and images are modeled as graph nodes, and the context such as the visibility between 2D and 3D points are modeled as graph edges. Afterwards, we use a heterogeneous GNN to predict map point importance scores based on the local appearance and the spatial context in the map graph. In addition, we propose two novel losses to guide the training: 1) a data-fitting term that selects points based on the appearance and the spatial distribution of the training query data, and 2) a K-Cover loss term that drives to sparse point selection with full-map coverage. When evaluated on an outdoor long-term dataset with significant environmental changes (Extended CMU Seasons [22]), our approach can select map points on stable and widely-visible structures (e.g., buildings/utility poles), while discarding points on changing object (e.g., foliage) or with highly repetitive texture (e.g., pavement). Compared with the K-Cover baseline [14], our approach outperforms in visual localization performance with the same map size.

2. Related Works

In this section, we first briefly describe the literature of robust feature learning, then review the existing map sparsification works, and finally cover relevant studies on GNNs that inspired our work.

2.1. Robust Feature Learning

Many previous works have attempted to solve the long-term visual localization problem by finding robust feature descriptors against environmental changes [25] (such as day-night, lighting conditions, and seasonal changes). Concrete examples include R2D2 [19], SOSNet [24], PixLoc [21] and [1]. Some methods look into the dynamics of visual features (and the corresponding physical environment) such as persistency [8] and repeatability [6]. Besides learning robust features, some works also attempt to overcome the environmental challenges by finding common information in 2D and 3D, such as semantic information [26, 27] and predicting depth from query images [18]. In this work, instead of finding robust features, we focus on sparsifying the SfM map globally by taking the whole map graph structure into consideration. We use Kapture [11], a modern mapping and localization library using R2D2, to generate data and evaluate the proposed method.

2.2. Map Sparsification

For a map that contains redundant information of a world, the goal of map sparsification is to select the most valuable subset. In previous works, it is common to assume that the map contains all the possible camera positions, and formulate the map compression as a K-Cover problem, which encourages each possible camera position (the key frame location in the map) to observe enough 3D points for performing robust PnP during localization under a total point number budget. The K-Cover problem is then solved using various techniques: a probabilistic approach [5], Integer Linear Programming (ILP) [7, 14] and Integer Quadratic Programming (IQP) [7, 15, 17]. A hybrid map and handcrafted heuristics were also used to determine the importance of map points [4, 13, 16]. These methods work well in a static world but suffer from performance degradation in vastly dynamic environments where many of the visibility edges in the map are outdated and invalidated.

2.3. Graph Neural Networks

Graph Neural Networks (GNNs) [10] have been applied to a variety of learning tasks with irregular data structures, such as citation graphs [29] and image visibility graphs [23]. An important advantage of Graph Neural Networks is the ability to handle heterogeneous data [31]. In this work, we represent the various information in SfM maps with heterogeneous graphs and extract features with a GNN. Recently, attention-based networks have shown strong performance in feature extraction from not only sequential data [28] but also graph structures such as 2D-3D matching [20]. Inspired by these works, we investigate the combination of heterogeneous GNN and attention, and demonstrated better final performance than the baselines.

3. Approach

Given an SfM map and a set of localization queries recorded at different times in a large-scale dynamic environment, our goal is to select a subset of 3D map points that

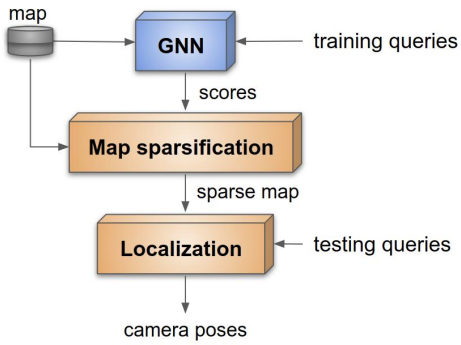


Figure 2. Overall framework. The proposed GNN learns to predict a score for each 3D point in the map. The predicted scores are used to sparsify the map. We report the performance of localizing a set of testing queries to the sparsified map.

are most informative, i.e. result in high localization performance. To achieve this, we first turn the input SfM map into a heterogeneous graph (Sec. 3.1) and train an attention-based GNN (Sec. 3.2, 3.3) to predict the importance scores for 3D map points, which are then used to sparsify the map. Finally, we localize the testing query set against the sparsified map, and report the localization performance (Sec. 4). An illustration of our overall system flow is shown in Fig. 2.

3.1. SfM Map as Heterogeneous Graph

A heterogeneous graph by definition is a graph structure that contains different types of nodes or edges. To represent an SfM map, three types of nodes are defined: 3D point nodes $\mathcal{V}_p$, 2D key point nodes $\mathcal{V}_k$, and image nodes $\mathcal{V}_m$. We also define three types of edges: visibility edges $\mathcal{E}_v$ connecting corresponding $\mathcal{V}_p$ and $\mathcal{V}_k$, kNN edges $\mathcal{E}_n$ connecting each $\mathcal{V}_p$ and its k nearest neighboring $\mathcal{V}_p$, and containing edges $\mathcal{E}_c$ connecting each $\mathcal{V}_k$ to the corresponding image $\mathcal{V}_m$. Each $\mathcal{V}_p$ might be connected to multiple $\mathcal{E}_v$s and $\mathcal{V}_k$s because it is observed by multiple map images. The SfM map is then represented with a heterogeneous graph $\mathcal{G} = \{\mathcal{V}_p, \mathcal{V}_k, \mathcal{V}_m, \mathcal{E}_v, \mathcal{E}_n, \mathcal{E}_c\}$. An illustration of our map graph is shown in Fig. 3(a)(b).

The per-point importance score is predicted based on local appearance and spatial context. We design our map graph to provide the information: first, the local appearance data are stored in $\mathcal{V}_k$ by embedding the key point descriptors extracted at the map building stage. Second, the spatial context is captured in kNN edges $\mathcal{E}_n$, which are derived from the 3D point positions stored in $\mathcal{V}_p$. The image nodes $\mathcal{V}_m$ do not carry features, but are used to trace connected $\mathcal{V}_k$ and $\mathcal{V}_p$ for ensuring the GNN selects enough number of $\mathcal{V}_p$ in the field-of-view of each $\mathcal{V}_m$, as shown in Fig. 3(c).

In practice, we store two sets of $\mathcal{V}_k$, $\mathcal{V}_m$, $\mathcal{E}_v$, and $\mathcal{E}_c$ in the map graph: one set is from the map and the other set is from localizing the query set on the map before sparsification. The first set is fed to the proposed GNN to provide information for score prediction. The second set is only available in the training area, and was only used to generate the point selection labels L_{gt} stored in $\mathcal{V}_p$ (Sec. 3.4).

Note that all the graph edges described above are directional. To be specific, $\mathcal{E}_n^{ji}$ represents a kNN edge from a neighbor $\mathcal{V}_p^j$ to the $\mathcal{V}_p^i$, and $\mathcal{E}_v^{wi}$ shows a visibility edge from key point $\mathcal{V}_k^w$ to map point $\mathcal{V}_p^i$, where i, j, w are node indices. The directionality of edges is useful in retrieving local subgraphs during network training (Sec. 3.3).

3.2. Graph Attention Network

To extract the spatial context from the map, we propose to aggregate the features from locally connected 3D point nodes with a Graph Attention Network (GATConv) [29,30]. For a 3D point node $\mathcal{V}_p^i$, a GATConv layer is applied to fuse the input node features and predict an output node feature. Formally, the GATConv operation is:

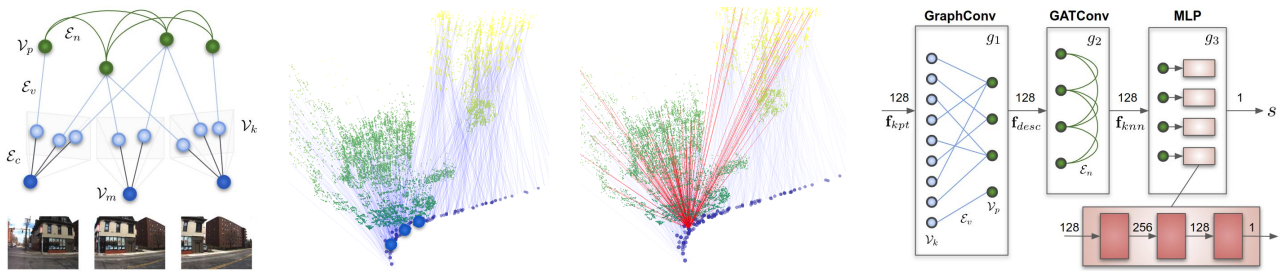
$$\begin{aligned} \alpha_{ij}^h &= softmax_j(a(\mathbf{W}^h\mathbf{h}_i, \mathbf{W}^h\mathbf{h}_j)) \\ \mathbf{h}_i^+ &= \sum_{h=1}^{H} \sum_{j\in\{1,\ldots,k+1\}} \alpha_{ij}^h \mathbf{W}^h\mathbf{h}_j, \end{aligned} \tag{1}$$

where $\mathbf{h}_j \in \mathbb{R}^F$ is an input feature from $\mathcal{V}_p^j$ to node $\mathcal{V}_p^i$ with feature dimension F. The input features are from the $\mathcal{V}_p^i$ itself and the kNN nodes, where $j \in \{1, 2, \ldots, k, i\}$ and k is the number of kNN nodes. The $\mathbf{W}^h \in \mathbb{R}^{F^+\times F}$ is a shared weight matrix, α_{ij}^h is the normalized attention coefficient, H is the number of attention heads, $a(.) : \mathbb{R}^{F^+} \times \mathbb{R}^{F^+} \rightarrow \mathbb{R}$ computes the attention coefficients. We aggregate the multi-head GATConv outputs by simple summation. The output $\mathbf{h}_i^+ \in \mathbb{R}^{F^+}$ is the output feature with dimension F^+ stored on $\mathcal{V}_p^i$. Empirically, we found this GATConv outperformed GraphConv [12] and SAGEConv [10] for our application.

3.3. Heterogeneous Graph Neural Network

We design a heterogeneous GNN to extract features and perform score prediction from the aforementioned map graph. The motivation is that the key point descriptors, although not raw pixel values, still contain valuable appearance information, enabling us to infer the 3D point scores from the connected 2D key point descriptors. The heterogeneity here enables us to define different operations according to the node and edge types.

Our GNN comprises three stages: 1) a descriptor gathering layer g_1, 2) a local feature extraction layer g_2, and 3) a final Multilayer Perceptron (MLP) layer g_3. In g_1, we trace the connected $\mathcal{E}_v$ for each $\mathcal{V}_p$ to collect the connected key point descriptors stored in $\mathcal{V}_k$. The collected descriptors are sent to a Graph Convolutional layer (GraphConv) [12] with LeakyReLU activation and summation aggregation functions. The output of g_1 is an aggregated point feature $\mathbf{f}_{desc}$ carrying the local appearance information. In g_2, we use



(a) (b) (c) (d)

Figure 3. An SfM map as a heterogeneous graph and the network structure. a) A simplified graph: dark blue circles are image nodes $\mathcal{V}_m$, light blue circles are key point nodes $\mathcal{V}_k$, and green circles are 3D point nodes $\mathcal{V}_p$. The edges $\mathcal{E}_c$, $\mathcal{E}_v$, and $\mathcal{E}_n$ are containing edges, visibility edges, and kNN edges, represented by black, light blue, and green colors. (b) A real snapshot of the Extended CMU Seasons dataset. Image nodes $\mathcal{V}_m$ and visibility edges $\mathcal{E}_v$ are as blue dots and lines. The key point nodes $\mathcal{V}_k$ are not shown. The color on the 3D points $\mathcal{V}_p$ encodes the distance to the current query image with green being low values and yellow high values. Three image node positions corresponding to the images in (a) are labeled with with dark blue circles. (c) In each training iteration, we sample an image node and trace the corresponding edges to extract a subgraph to run our GNN. The $\mathcal{E}_v$ used to extract this subgraph are shown as red lines. (d) Our network takes the key point descriptors $\mathbf{f}_{kpt}$ and predicts a score s for each map point. We define three network layers: g_1 that aggregates descriptors to 3D points, g_2 that collects 3D local information, and g_3 as the final per-point MLP (pink blocks). A dark pink block is an MLP layer, which contains a linear layer and a LeakyReLU activation. The numbers above the arrows are feature dimensions.

the GATConv layer (Sec. 3.2) to gather the nearby point features from the kNN $\mathcal{V}_p$, generating a local feature $\mathbf{f}_{knn}$ that captures spatial context. Finally, a 3-layer MLP g_3 is used to convert the point feature dimension to 1 and a sigmoid layer is used to constrain the predicted score value s to $[0, 1]$. The network structure is shown in Fig. 3(d).

Let $i, j \in \{1, 2, \ldots, N_p\}$ denote the map point indices and $w \in \{1, 2, \ldots, N_k\}$ be a key point index, where N_p and N_k are the total number of map points and key points. Let $\mathcal{G}$ denote the map graph, the score prediction steps are:

$$\begin{aligned} \mathbf{f}^i_{desc} &= g_1(\{\mathcal{V}^w_k | \mathcal{E}^{wi}_v \in \mathcal{G}\}) \\ \mathbf{f}^i_{knn} &= g_2(\{\mathbf{f}^j_{desc} | \mathcal{E}^{ji}_n \in \mathcal{G}\}) \\ s^i &= Sigmoid(g_3(\mathbf{f}^i_{knn})), \end{aligned} \tag{2}$$

where $\mathbf{h}_i = \mathbf{f}^i_{desc}$ and $\mathbf{h}^+_i = \mathbf{f}^i_{knn}$ in Eq. 1.

To facilitate GNN training on large-scale graphs, we sample a $\mathcal{V}_m$ to extract a local subgraph for each training batch and only run our GNN on the local subgraph. Given a $\mathcal{V}_m$, we first extract the connected $\mathcal{V}_k$ by tracing $\mathcal{E}_c$. Afterwards we trace $\mathcal{E}_v$ and $\mathcal{E}_n$ to extract the corresponding $\mathcal{V}^i_p$ and its neighbors. Finally, we trace the $\mathcal{E}_v$ connecting to the neighboring $\mathcal{V}^j_p$ for computing the neighboring $\mathbf{f}^j_{desc}$.

3.4. Training Losses

Our losses promote high scores on points with two properties: first, the descriptor distribution of the selected points should *align with the descriptors that are useful for training query localization*. Second, the selected points should *cover all the possible viewing poses*, so that all the queries would observe a sufficient amount of points within the field-of-view. We propose a training loss with two terms:

Data Fitting Term. Since the ILP baseline performs well in a static environment [14], we use it as an oracle to generate point selection labels. We first localize the training queries on the map, collecting the 2D-3D matches between the training queries and the map, and run the ILP baseline [14] to obtain the point selection results, which is a binary vector L_{gt}. The ILP baseline in this setting, denoted as ILP (query), factors out the environmental changes and performs well (Fig. 6(a)), but cannot be achieved in the real world unless the training queries cover the whole mapped area. The data fitting term is then computed by comparing the predicted scores $\mathbf{S}$ and L_{gt} with a Binary Cross Entropy (BCE) loss $\mathcal{L}_{BCE}$:

$$\mathcal{L}_{BCE} = BCE(L_{gt}, \mathbf{S}). \tag{3}$$

For the maps we evaluated with, we found the computation of ILP formulation is tractable to process the whole map. It is also possible to use IQP [17] for label generation, but in practice IQP is computationally intractable to run on large-scale maps without additional graph partition steps. The potential effect of graph partition on localization performance is beyond the focus of this paper.

K-Cover Term. Training the network with $\mathcal{L}_{BCE}$ alone would only encourage point selection that aligns with L_{gt} in the training set, but it does not guarantee map point coverage across the whole map. To compensate this, we leverage transductive learning and additionally encourage the sum of the scores of all the $\mathcal{V}_p$ connected to each $\mathcal{V}_m$ to be close to a predefined positive integer K, which indicates the number of 3D points each image should observe to support robust localization. Empirically, we observed that this setup converges faster during training than the case not penalizing the

samples larger than K. Upon satisfying the K-Cover constraint, we also encourage the score sparsity to select fewer points with an L_1 norm loss. Letting l be the index of image node $\mathcal{V}_m$, we define ϕ_l as the set of map point indices that selects the set of $\mathcal{V}_p$ whose connected $\mathcal{V}_k$ is within $\mathcal{V}_m^l$ (as the red edges in Fig. 3(c)). The score prediction of $\mathcal{V}_p^i$ is denoted as s_i. The final K-Cover loss is:

$$\phi_l = \{i | \mathcal{E}_c^{lw} \in \mathcal{G} \cap \mathcal{E}_v^{wi} \in \mathcal{G}\},$$
$$\mathcal{L}_{KC} = \sum_l |K - \sum_{i \in \phi_l} s_i| + \lambda ||\mathbf{S}||_1. \tag{4}$$

By adding both terms, we propose the final loss as:

$$\mathcal{L} = \mathcal{L}_{BCE} + \mathcal{L}_{KC}. \tag{5}$$

The data split and usage is summarized in Tab. 1. Note that the training and testing queries are spatially non-overlapping and the pre-built map covers both the train and test areas. The role of the training queries is to provide up-to-date appearance information that cannot be obtained from the outdated map data, as we focus on the temporal appearance difference. In this case, the training and testing data should not overlap spatially but can overlap temporally.

4. Evaluation

In this section, we describe the data preparation process, implementation details and experimental results.

Data Preparation We evaluated our approach on Extended CMU Seasons dataset [2, 22], which consists of 12 sessions recorded by two cameras across months in multiple locations. To simulate the natural accumulation of map data, we used sessions 0-5 to build a multi-session map, and used sessions 6-11 as the query set to localize. The mapping and query sets have significantly different appearance. The map was built with Kapture [11]. The localization performance was measured by registering the query sets on the multi-session maps built from session 0-5 . We used 13 slices (scenes) for evaluation, including the Urban and Suburban slices (3-4, 6-16), and discarded the Park slices and slice 2, 5 due to poor localization performance on the raw multi-session map before sparsification. The 13 slices for evaluation contained various objects such as vegetation, buildings, and moving objects, and multiple weathers like sunny, cloudy, and snowy. An example of seasonal appearance changes is shown in Fig. 4(a)(b).

We further split the query set by the two cameras (camera 0, camera 1), and used camera 0 of all the 13 slices for training, the camera 1 of slice 3 for validation, and the camera 1 of the other 12 slices for testing. The number of mapping/query images in each data set split are 17837/16077 for training, 1333/1428 for validation, and 16498/15627 for testing. Note that the camera 0 and camera 1 point towards two sides of the road and have no overlap as Fig. 4(c)(d).

Table 1. The data splits by type and usage. There are two cameras in the Extended CMU Seasons dataset, noted by c0 and c1. We separated the 12 sessions temporally and used the old sessions (0-5) for mapping, the new sessions as queries (6-11).

Data Type	Spatial		Temporal		Used for
	train	test	old	new	
map ($\mathcal{G}$)	✓(c0)	✓(c1)	✓		$\mathcal{L}_{KC}, \mathcal{L}_{BCE}$
query (train)	✓(c0)			✓	$\mathcal{L}_{BCE}$
query (test)		✓(c1)		✓	not used

Implementation Details The proposed GNN is implemented with PyTorch and Deep Graph Library (DGL) [30]. During the training process, we loop through the map image nodes $\mathcal{V}_m$ in the training set to extract subgraphs to run GNN on. A four-layer DGL node sampler ($\mathcal{V}_m \leftarrow \mathcal{V}_k \leftarrow \mathcal{V}_p \leftarrow knn\ \mathcal{V}_p \leftarrow \mathcal{V}_k\ of\ knn\ \mathcal{V}_p$) was used to extract the subgraph in each training iteration to provide necessary information. It took about 3.97s to process a map graph (with average 4.12×10^5 map points) on an Nvidia Quadro RTX 3000 GPU and an i7-10850H CPU @ 2.70GHz. More graph statistics are in supplementary material.

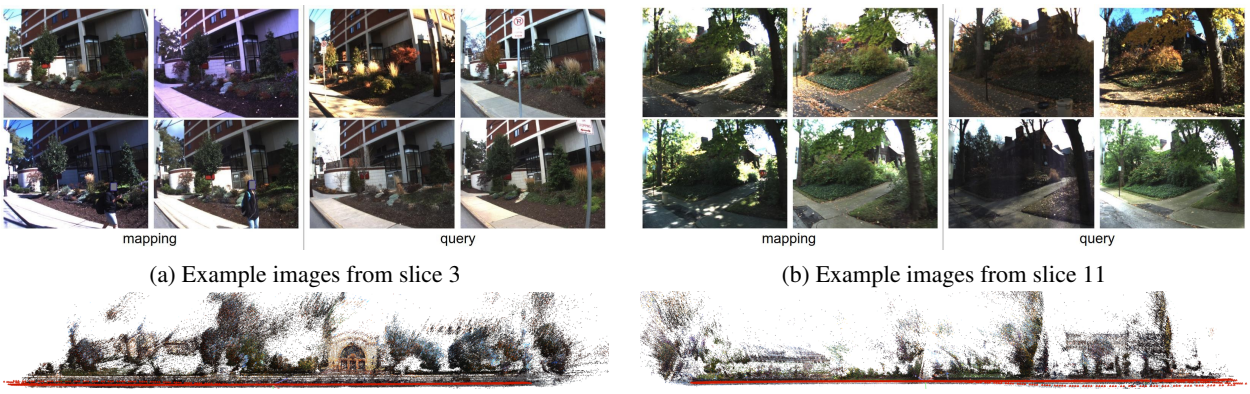
As for parameters, we used $k = 9$ to build kNN edges among 3D points, $K = 30$ and $\lambda = 0.01$ in the K-Cover loss. The ILPs [14] were implemented using Gurobi [9], and is configured with $b = 30$. We used $n_{desired} = 500$ [14] to generate L_{gt}. The network was trained with an AdamW optimizer with learning rate 0.001 and βs $(0.9, 0.999)$ for 20 epochs. For each case, we selected the epoch with the best validation performance for testing.

Final evaluation is conducted with the Kapture localization pipeline [11]. Given a query image, it first retrieves the map images with similar global features, and then performs 2D-2D key point descriptor matching between the query image and the retrieved map images. The 3D points corresponding to the matched map key points are used to perform PnP with the matched query key points. The Kapture default R2D2 [19] descriptor is used in map building, localization, and as our network input $\mathbf{f}_{kpt}$.

4.1. Localization Performance on Sparsified Maps

For each map sparsification method, we first obtained its point selection result, and reconstructed the multi-session map in Kapture format with only the the key points and descriptors that correspond to the selected points. We used the number of point descriptors remaining in the map (#kpts) as map size proxy, since these high-dimension descriptors (e.g., 128 for R2D2) occupied most of the map storage space. Three baselines were compared:

- **Random** : randomly select a subset of map points up to the allowed budget.
- **ILP (map)** : the conventional ILP [14], which assem-



(a) Example images from slice 3

(b) Example images from slice 11

(c) Training set map example (camera 0 of slice 4)

(d) Test set map example (camera 1 of slice 4)

Figure 4. Example images from Extended CMU Seasons dataset. We observed large seasonal changes across the whole dataset. In (a)(b), on the left are the map image examples and on the right are query image examples recorded at similar locations. In addition, the Extended CMU Seasons dataset was recorded by two cameras. We used camera 0 (c) for training and camera 1 (d) for validation/testing. The training and test sets capture two sides of the road with no spatial overlap. The red dots at the bottom are the map image locations.

bles the K-Cover problem with 1) the visibility edges stored in the map, and 2) the per-point weight based on number of observations in the map.

- **ILP (query)** : the ideal ILP [14] that has access to test queries. The K-Cover problem is constructed using visibility edges from localizing the test queries on the map before sparsification, and points are weighted according to the number of observations during the test query localization. This approach indicates the ideal performance of ILP approach without environmental changes and cannot be achieved in the real world.

We obtained data points by sweeping the desired total point number $n_{desired}$ [14]. For our method, we randomly selected points with predicted scores larger than 0.1. If there were not enough points with scores larger than 0.1 to satisfy $n_{desired}$, we randomly selected from the rest of the points. We observed that predicted score distribution is close to binary (due to the L_1 norm sparsity loss) and the point selection result is not sensitive to the score threshold.

Overall, our proposed approach outperformed the ILP (map) baseline in all the testing slices by achieving higher localization recall (success rate) under the same map sizes, as shown in Tab. 2 and Fig. 6. Qualitatively, we observed that compared with the ILP (map) baseline, the proposed method selects map points on static structures that are more useful for query set localization, as in Fig. 7 and Fig. 8.

Network structures. We also compared the following configurations for the g_2 GNN layer: GraphConv [12], SAGEConv (with mean aggregation function) [10], and GATConv (with $H = 4$) [29]. The compared networks had the same feature dimensions and the LeakyReLU ($slope = 0.1$) activations. Our results showed GATConv

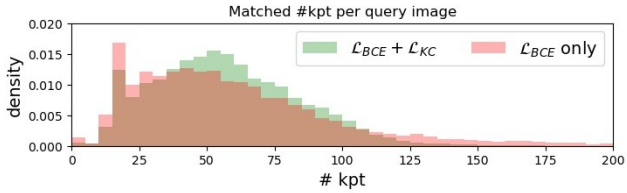


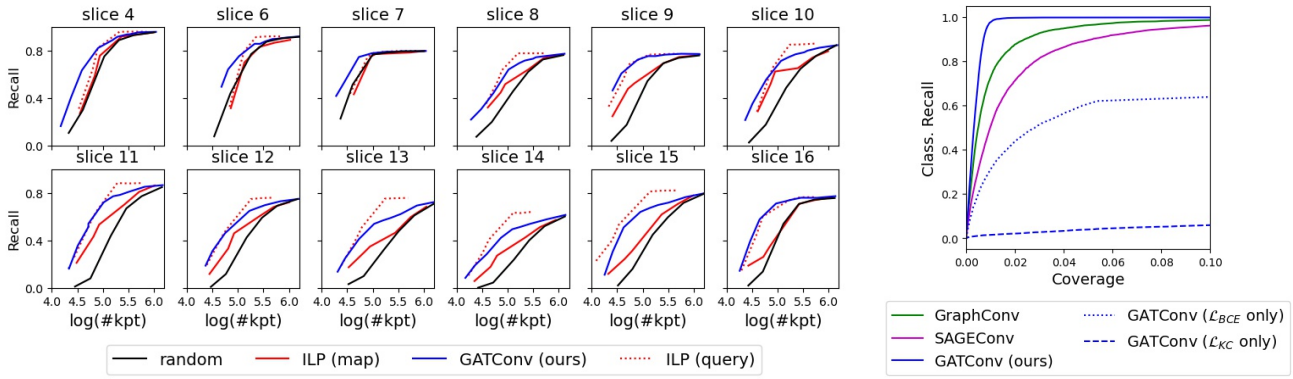
Figure 5. The density histogram of 2D-3D matching number for each testing query image during localization. After applying $\mathcal{L}_{KC}$ we observed less images with extreme number of matches, which is preferred for consistent localization performance under a map size budget. Both histograms are generated under the same map size budget (total #kpt is $\sim 6.3 \times 10^5$).

outperformed GraphConv and SAGEConv significantly in terms of not only localization recall (Tab. 2) under the same map sizes, but also classification performance with respect to ILP (query) as shown in Fig. 6(b).

Training losses. Finally, the network trained without either $\mathcal{L}_{BCE}$ or $\mathcal{L}_{KC}$ performed worse than the one with combined loss, as shown in Tab. 2 and Fig. 6(b). The $\mathcal{L}_{BCE}$ was only trained in the training area, since no labels are available in the testing area. The $\mathcal{L}_{KC}$ were trained with the whole input map graph (which covers both the training and testing areas). Interestingly, although the $\mathcal{L}_{BCE}$-only configuration got the lowest training $\mathcal{L}_{BCE}$, adding $\mathcal{L}_{KC}$ improved the classification performance in the test set. We further observed that when localizing testing queries, the map sparsified with $\mathcal{L}_{KC}$ obtained less extreme numbers of matched key points (Fig. 5). This is favorable because each query obtained enough matches, but not too many that caused a waste in map storage.

Table 2. Average recall under different map sizes. For each slice (a sequence in The Extended CMU Seasons dataset), we linearly interpolated the recall curves to obtain the recall numbers under the same number of key point descriptors, and computed the average recalls with respect to the number of images. Three recall thresholds were compared. The recall number represents the ratio of image samples with localization pose errors less than the corresponding recall threshold. As a reference, the average key point number before sparsification is $\sim 2.8 \times 10^6$. The detailed map graph statistics and the full recall curves are in the supplementary material.

Recall threshold	**0.25m, 2.0°**				**0.5m, 5.0°**				**5.0m, 10.0°**			
Avg. map size (10^4 #kpts)	3	5	10	20	3	5	10	20	3	5	10	20
Random	0.07	0.18	0.41	0.59	0.07	0.20	0.44	0.63	0.09	0.23	0.49	0.70
ILP (map)	0.15	0.31	0.53	0.64	0.19	0.36	0.59	0.69	0.25	0.43	0.66	0.76
GraphConv	0.31	0.48	0.64	**0.73**	0.34	0.52	0.69	0.77	0.39	0.58	0.76	0.85
SAGEConv	0.27	0.42	0.58	0.68	0.30	0.46	0.62	0.72	0.34	0.51	0.68	0.79
GATConv (ours)	**0.35**	**0.52**	**0.67**	**0.73**	**0.40**	**0.57**	**0.72**	**0.78**	**0.46**	**0.64**	**0.80**	**0.86**
GATConv ($\mathcal{L}_{BCE}$ only)	0.25	0.38	0.53	0.65	0.28	0.42	0.57	0.70	0.32	0.47	0.64	0.77
GATConv ($\mathcal{L}_{KC}$ only)	0.09	0.23	0.42	0.60	0.10	0.25	0.45	0.64	0.12	0.29	0.52	0.71
ILP (query)	0.24	0.46	0.69	0.80	0.30	0.53	0.75	0.85	0.38	0.60	0.83	0.92



(a) The recall vs. map size curves for each slice in the test set

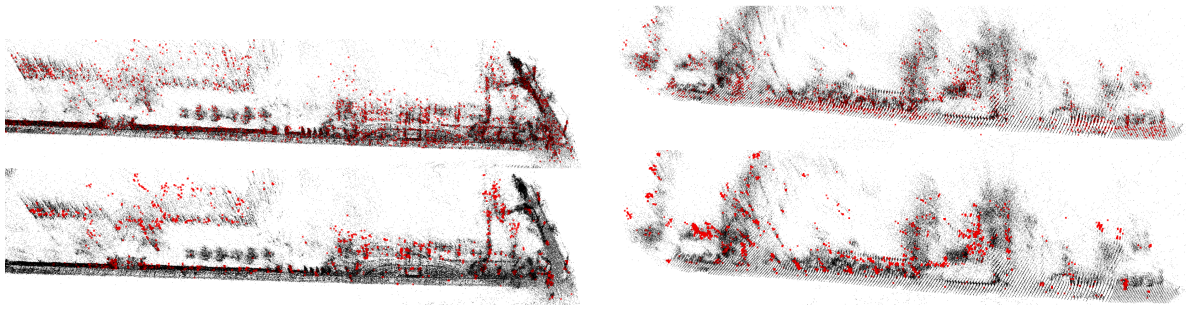
(b) Classification performance

Figure 6. Localization and classification recall comparisons. (a) Our approach outperformed the ILP (map) and the random baselines in all test slices, achieving higher recalls (success rate) under the same map size budgets. On the other hand, the ILP (query) also significantly outperformed ILP (map), showing the impact of environmental changes on baselines. The recall error thresholds here are 0.25m and 2.0°. (b) Compared with ILP (query), the GATConv trained with the full proposed loss achieved the highest classification recall (ratio of selected positive labels) under the same coverage (ratio of the number of selected points against total number of points).

5. Discussion and Limitations

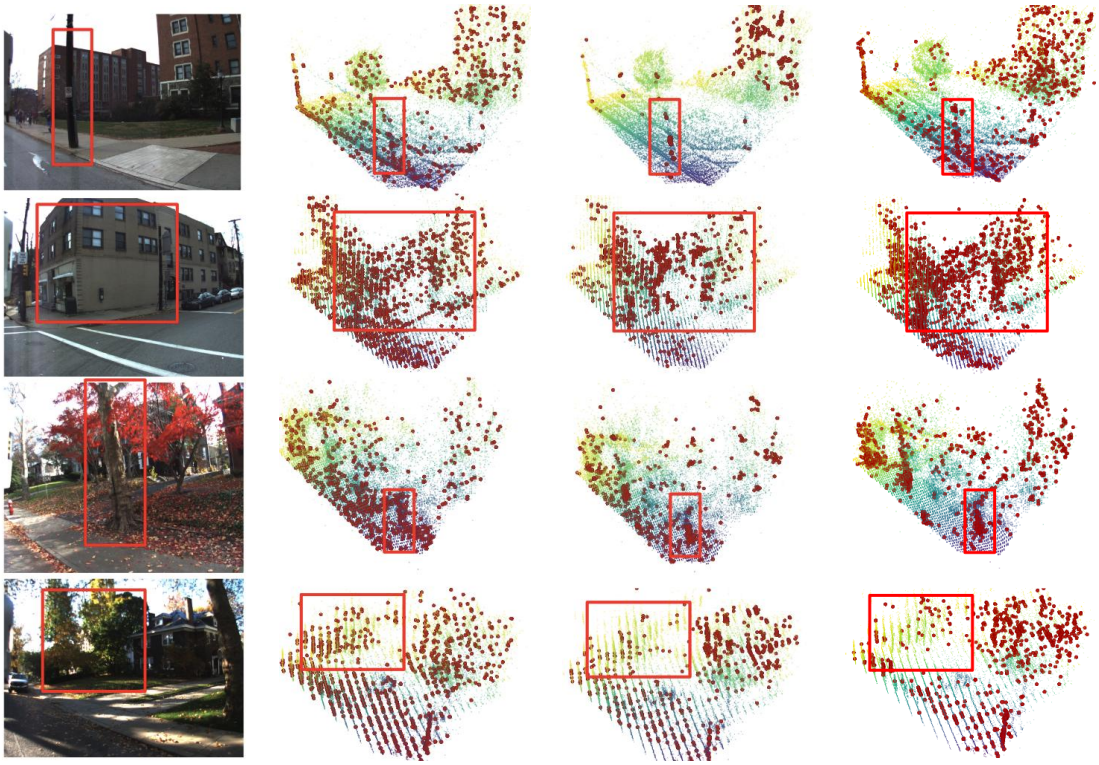
The heterogeneous graph used in this work is so flexible that it is easy to include more information as additional node or edge features. This implies a great potential for future works. Choices of additional information include timestamps (for capturing periodic environmental change) or the data from other sensors. It is also easy to apply other training losses to sparsify the map for different tasks other than conventional localization. Furthermore, we observed that certain objects, such as buildings and utility poles. are more likely to get higher scores. This implies the possibility of using semantic labels to assist point score prediction. It is also worth mentioning that the heterogeneous GNN framework can potentially be applied to other practical graphs, such as the factor graph for in SLAM. Comparing the GNN-based method with the existing factor graph sparsification works [3] is another interesting future direction. On the other hand, one important factor affecting the result is the point sampling strategy. Given the same set of predicted scores, different point selection strategies would lead to different performance. In our system, we used simple random down-sampling and a score threshold that achieved outstanding performance, but exploring different point sampling strategies can be an interesting future work.

As for limitations, typically the key in map sparsification is to compress a map of a given scene, thus the generalization to an unseen scene has not been our focus. For the K-Cover setup to work, the camera trajectories at query time should be a subset of the camera trajectories in the map. This applies to ours and the related works. Besides, we only focused on removing points from an existing map, so the result is limited by the localization performance on



(a) slice 3 (b) slice 11

Figure 7. Large-scale point selection results. The upper row is the results from ILP (map) and the lower row is ours with a 0.1 score threshold. The black points are the map 3D points before sparsification and the red points are the selected points. Our method selects points on static structures, such as building walls, utility poles, and tree stems and avoids foliage that changes across seasons.



(a) images (b) ILP (map) (c) GATConv (ours) (d) ILP (query)

Figure 8. Qualitative visualizations. The camera positions are at the bottoms of the point cloud visualizations (b)(c)(d). The corresponding parts in each row are labels by red boxes. Overall, we observe that the point selection of ILP (map) is less discriminative in selecting static points than ILP (query) and ours. We compared the cases with similar numbers of key points so the total 3D point number varies.

the raw map. How to add/merge new information to the map is also worth exploring in the future. Finally, naive data splits (by camera and by slice) is used in our experiments, but in practice it is better to minimize the training set size to reduce the map update workload.

6. Conclusion

In conclusion, we proposed a heterogeneous GNN for visual map sparsification and proved its effectiveness in real-world environment. This work opens a new avenue for applying the abundant GNN related techniques to SfM applications. Our future work would be map sparsification for multi-sensor maps and more map graph representations.

Acknowledgement

Ming-Fang Chang is supported by the CMU Argo AI Center for Autonomous Vehicle Research. We also thank Tianwei Shen and dear labmates in CMU and Meta for providing valuable discussion and suggestions.

References

[1] Long-term visual localization challenges in iccv 2021. https://sites.google.com/view/ltvl2021/challenges. Accessed: 2022-03-09. 2

[2] Hernan Badino, Daniel Huber, and Takeo Kanade. The CMU Visual Localization Data Set. http://3dvis.ri.cmu.edu/data-sets/localization, 2011. 5

[3] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reidand, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. 2016. 7

[4] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid scene compression for visual localization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 2

[5] Song Cao and Noah Snavely. Minimal Scene Descriptions from Structure from Motion Models. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014. 2

[6] Anh-Dzung Doan, Daniyar Turmukhambetov, Yasir Latif, Tat-Jun Chin, and Soohyun Bae. Learning to Predict Repeatability of Interest Points. In *IEEE Int. Conf. Robotics and Automation*, 2021. 2

[7] Marcin Dymczyk, Simon Lynen, Michael Bosse, and Roland Siegwart. Keep It Brief: Scalable Creation of Compressed Localization Maps. In *IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2015. 1, 2

[8] Marcin Dymczyk, Elena Stumm, Juan Nieto, Roland Siegwart, and Igor Gilitschenski. Will it last? Learning stable features for long-term visual localization. In *IEEE Int. Conf. on 3D Vis.*, 2016. 2

[9] Gurobi Optimization, LLC. Gurobi - The Fastest Solver. https://www.gurobi.com/, 2021. 5

[10] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Conf. Neural Inform. Process. Syst.*, 2017. 2, 3, 6

[11] Martin Humenberger, Yohann Cabon, Nicolas Guerin, Julien Morat, Jérôme Revaud, Philippe Rerole, Noé Pion, Cesar de Souza, Vincent Leroy, and Gabriela Csurka. Robust Image Retrieval-based Visual Localization using Kapture. *arXiv:2007.13867 [cs]*, 2020. 2, 5

[12] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Int. Conf. Learn. Represent.*, 2017. 3, 6

[13] Stefan Luthardt, Volker Willert, and Jürgen Adamy. LLama-SLAM: Learning high-quality visual landmarks for long-term mapping and localization. In *IEEE Int. Conf. Intell. Transportation Syst.*, 2018. 2

[14] Simon Lynen, Bernhard Zeisl, Dror Aiger, Michael Bosse, Joel Hesch, Marc Pollefeys, Roland Siegwart, and Torsten Sattler. Large-scale, real-time visual-inertial localization revisited. *Int. J. of Robotics Res.*, 2020. 1, 2, 4, 5, 6

[15] Marcela Mera-Trujillo, Benjamin Smith, and Victor Fragoso. Efficient Scene Compression for Visual-based Localization. In *IEEE Int. Conf. on 3D Vis.*, 2020. 1, 2

[16] Peter Mühlfellner, Mathias Bürki, M. Bosse, W. Derendarz, Roland Philippsen, and P. Furgale. Summary maps for lifelong visual localization. In *J. of Field Robotics*, 2016. 2

[17] Hyun Soo Park, Yu Wang, Eriko Nurvitadhi, James C. Hoe, Yaser Sheikh, and Mei Chen. 3D Point Cloud Reduction Using Mixed-Integer Quadratic Programming. In *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2013. 1, 2, 4

[18] Nathan Piasco, Désiré Sidibé, Valérie Gouet-Brunet, and Cedric Demonceaux. Learning Scene Geometry for Visual Localization in Challenging Conditions. In *IEEE Int. Conf. Robotics and Automation*, 2019. 2

[19] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2D2: Repeatable and Reliable Detector and Descriptor. In *Conf. Neural Inform. Process. Syst.*, 2019. 2, 5

[20] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning Feature Matching With Graph Neural Networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[21] Paul-Edouard Sarlin, Ajaykumar Unagar, Mans Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, et al. Back to the Feature: Learning Robust Camera Localization from Pixels to Pose. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 2

[22] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomas Pajdla. Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 2, 5

[23] Yan Shen, Zhang Maojun, Lai, Shiming, Liu Yu, and Peng Yang. Image Retrieval for Structure-from-Motion via Graph Convolutional Network. *Inform. Sci.*, 2021. 2

[24] Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen, and Vassileios Balntas. SoSnet: Second Order Similarity Regularization for Local Descriptor Learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 2

[25] Carl Toft, Will Maddern, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, et al. Long-Term Visual Localization Revisited. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020. 2

[26] Carl Toft, Carl Olsson, and Fredrik Kahl. Long-term 3D Localization and Pose from Semantic Labellings. In *Int. Conf. Comput. Vis. Worksh.*, 2017. 2

[27] Carl Toft, Erik Stenborg, Lars Hammarstrand, Lucas Brynte, Marc Pollefeys, Torsten Sattler, and Fredrik Kahl. Semantic Match Consistency for Long-Term Visual Localization. In *Eur. Conf. Comput. Vis.*, 2018.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Conf. Neural Inform. Process. Syst.*, 2017. 2

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph At-

tention Networks. In *Int. Conf. Learn. Represent.*, 2018. 2, 3, 6

[30] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. *arXiv:1909.01315 [cs, stat]*, 2020. 3, 5

[31] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Peng Cui, P. Yu, and Yanfang Ye. Heterogeneous Graph Attention Network. In *The World Wide Web Conf.*, 2021. 2