Image Retrieval Re-ranking using Graph Neural Networks

Gustav Hanning

Master's thesis
2024:E18

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Computer Vision and Machine Learning

Centrum Scientiarum Mathematicarum

Abstract

Image retrieval is the task of finding images in a database similar to a given query image. The retrieved images, typically a small subset of the entire database, are initially ordered based on their similarity with the query. They can subsequently be re-ranked to improve the retrieval accuracy. Database images that are relevant to the query should increase in rank and vice versa.

In this thesis the re-ranking process is modeled as a graph neural network. The nodes of the graph are the query and retrieved database images. For each node an affinity vector is computed which encodes the visual similarity between the image and a set of anchor images. The vectors are refined by message passing between nodes, using self-attention. Database images are re-ranked according to the similarity between their refined affinity vector and that of the query.

The network is trained on a large-scale dataset and evaluated against three other re-ranking algorithms. Results show that the method proposed in the thesis achieves significantly higher precision.

Acknowledgements

I would like to thank my supervisors Viktor Larsson and Gabrielle Flood for their support and guidance throughout the work on the thesis.

The computations were enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre.

Figures 1.1, 3.1 and 4.1 contain images from the "Mapillary Street-level Sequences Dataset" by Mapillary, licensed under CC BY-NC-SA 4.0.

Contents

1 Introduction **7**
- 1.1 Background . . . 7
- 1.2 Problem Formulation . . . 8
- 1.3 Purpose and Goals . . . 8
- 1.4 Limitations . . . 9
- 1.5 Related Work . . . 9
- 1.6 Report Outline . . . 9

2 Theory **11**
- 2.1 Retrieval using Image Descriptors . . . 11
- 2.2 Image Retrieval Re-ranking . . . 12
  - 2.2.1 Query Expansion . . . 12
  - 2.2.2 SuperGlobal . . . 12
  - 2.2.3 Geometric Verification . . . 13
  - 2.2.4 Neighborhood Similarity . . . 13
- 2.3 Affinity Features . . . 13
  - 2.3.1 Positional Affinity . . . 14
- 2.4 Graph Neural Networks . . . 14
  - 2.4.1 Attention and Message Passing . . . 14
- 2.5 Evaluation Metrics . . . 15
- 2.6 AP Loss . . . 16
- 2.7 Contrastive and MSE losses . . . 18

3 Method **19**
- 3.1 Dataset . . . 19
- 3.2 Network Architecture . . . 20
- 3.3 Training Details . . . 22

4 Evaluation **23**

4.1 Experimental Setup 23
4.2 Results 24
4.3 Ablation Study 25
4.4 Discussion 26
4.5 Future Work 27

References **31**

Chapter 1

Introduction

This chapter gives the background of the thesis and introduces the problem of image retrieval re-ranking. The purpose, goals and limitations of the thesis are stated followed by a description of related work in the area.

1.1 Background

Content-based image retrieval (CBIR) is the task of finding images in a database that are similar to a given query image. Similarity can be defined in terms of color, texture, shape or other features. Since the number of images in the database may be large it is not feasible to compare the pixel data of the query image directly with all the database images. Instead

Figure 1.1: Image retrieval re-ranking. The images most similar to the query are retrieved from the database and then re-ranked to improve accuracy. The images are from the Mapillary SLS dataset [18].

an efficient representation is computed from each image and the search is performed using these image descriptors. Popular applications of image retrieval include place recognition and visual localization, where the goal is to find images of the same place and in the case of localization estimate the camera pose of the query image.

Image retrieval re-ranking, exemplified in figure 1.1, is performed by re-ordering the top images found in the database to further improve the results. Database images that are relevant to the query should increase in rank and vice versa. Recent advancements show that the re-ranking process can successfully be modeled as a graph neural network (GNN). The query image and the retrieved database images are nodes in the network and their descriptors are refined with message passing using self-attention.

1.2 Problem Formulation

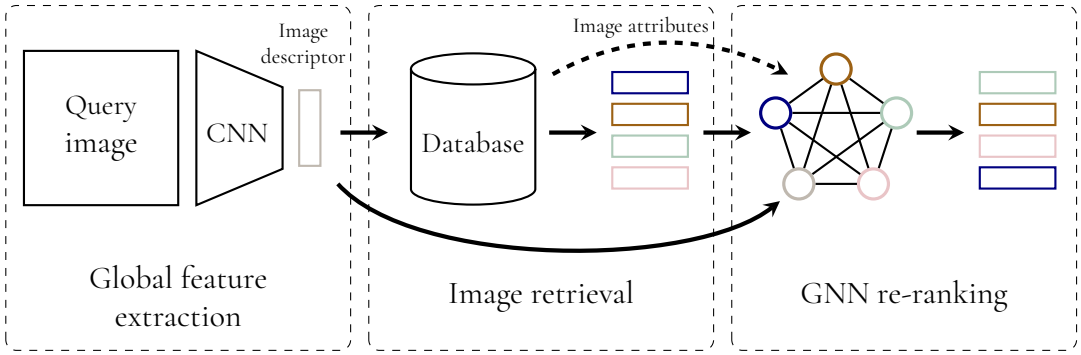

Figure 1.2: The image retrieval process, divided into three phases.

In this thesis the image retrieval process, depicted in figure 1.2, can be divided into three separate phases:

1. Extract a global image descriptor from the query image.
2. Retrieve the most similar descriptors from the database.
3. Re-rank the retrieved database images using a GNN.

In the first phase a convolutional neural network (CNN) computes a global descriptor for the query, describing the contents of the whole image. The same CNN has also been used to generate descriptors for all database images and in the second phase the descriptors best matching that of the query are retrieved. Finally in the third phase a GNN re-ranks the retrieved database images. If the database contains additional attributes like image positions they can be added to the nodes to increase the re-ranking accuracy.

1.3 Purpose and Goals

The purpose of the thesis is to explore the use of graph neural networks in the area of content-based image retrieval re-ranking. A GNN architecture suitable for re-ranking should be proposed and details given on how the network can be trained on one or more datasets.

The primary goal is to implement and train the proposed GNN and to evaluate it against other methods in terms of accuracy. A secondary goal is to include database image attributes to help guide the re-ranking process, as this can help disambiguate challenging cases.

1.4 Limitations

The focus of this thesis is the re-ranking of database images, i.e. the last phase in figure 1.2. A pre-trained CNN is utilized for creating the global descriptors and it is not fine-tuned to the specific dataset used in the thesis or trained together with the GNN.

1.5 Related Work

The problem of image retrieval dates back to the 1990s. In a survey Zheng et al. [21] show how models based on hand-crafted local descriptors such as SIFT have been replaced by CNN based models. One such CNN architecture is NetVLAD [1], which is considered to be among the current state-of-the-art methods for computing compact image representations.

Chum et al. [2] present a method for re-ranking using so-called query expansion. A new query descriptor is constructed by taking the average of the original descriptor and the descriptors retrieved from the database. Gordo, Radenovic and Berg [4] propose a GNN model (LAttQE) that learns the weights of the query expansion, consisting of a stack of self-attention encoders.

As a part of their SuperGlobal CBIR system Shao et al. [16] design a re-ranking procedure where database and query descriptors are refined in two different ways. Similarity scores are computed between the refined database descriptors and both the original and refined query descriptor. Database images are then re-ordered according to the average of the two scores.

Most similar to this thesis is the work of Ouyang et al. [11], in which image retrieval re-ranking is performed using a graph neural network with self-attention. This thesis simplifies the architecture of their network and utilizes a more efficient loss function for training. Additionally their concept of affinity features is extended to include positional data.

1.6 Report Outline

Chapter 2 presents the theory behind GNN based image retrieval re-ranking.

Chapter 3 describes the dataset used to train the GNN, the network architecture and the training process.

Chapter 4 contains the evaluation of the proposed GNN re-ranking method, along with results from several ablation experiments. The chapter ends with a summary of the findings of the thesis and ideas for future work in the area.

Chapter 2

Theory

This chapter contains the theoretical background of the proposed GNN re-ranking method. First, the basic image retrieval setup is presented. Re-ranking is introduced as a way to improve the retrieval results and four types of re-ranking algorithms are discussed. The concept of affinity features is explained and it is shown how the features can be extended with positional information.

Next, GNNs and message passing using attention to refine the features are described. Relevant metrics to evaluate an image retrieval system are stated and a loss function to maximize one of these metrics is derived. Lastly, the losses used to train the model [11] are given.

2.1 Retrieval using Image Descriptors

A global image descriptor is a feature vector $\boldsymbol{d}$ of dimension d that encodes the contents of an image. It is extracted from the image $\boldsymbol{I}$ by a function f such that $\boldsymbol{d} = f(\boldsymbol{I}) \in \mathbb{R}^d$. The function f can be either hand-crafted or learned from data.

The cosine similarity

$$s(\boldsymbol{d}_i, \boldsymbol{d}_j) = \frac{\boldsymbol{d}_i^T \boldsymbol{d}_j}{||\boldsymbol{d}_i|| ||\boldsymbol{d}_j||} \tag{2.1}$$

is used as a measure of the similarity between two images with descriptors $\boldsymbol{d}_i$ and $\boldsymbol{d}_j$. If the feature vectors are normalized to unit length, which is often the case, the cosine similarity can efficiently be computed as

$$s(\boldsymbol{d}_i, \boldsymbol{d}_j) = \boldsymbol{d}_i^T \boldsymbol{d}_j. \tag{2.2}$$

Now let $\{\boldsymbol{I}_i\}_{i=1}^N$ be a database of images with corresponding descriptors $\{\boldsymbol{d}_i\}_{i=1}^N$, and let $\boldsymbol{d}_0$ be the descriptor of a query image $\boldsymbol{I}_0$. Image retrieval is performed by computing the cosine similarity between $\boldsymbol{d}_0$ and all database image descriptors, followed by sorting of the database images in descending order of similarity. The top $\boldsymbol{K}$ database images make up the retrieval results and the indices of the retrieved database images are denoted $r_1, ..., r_K$. The best match for the query image is thus $\boldsymbol{I}_{r_1}$, the second best $\boldsymbol{I}_{r_2}$ and so on. For ease of notation we set $r_0 = 0$.

2.2 Image Retrieval Re-ranking

Retrieval of the top $\boldsymbol{K}$ images from the database may be followed by a re-ranking step where retrieved images are re-ordered for higher accuracy. Since $\boldsymbol{K}$ is typically much smaller than $\boldsymbol{N}$ a more thorough analysis of the images is possible.

2.2.1 Query Expansion

Query expansion (QE) is a class of re-ranking methods that computes a new query descriptor $\hat{\boldsymbol{d}}_0$ as a weighted average of the original query and the retrieved database descriptors. A second retrieval is then performed using the new descriptor. The simplest form of QE is average query expansion (AQE) [2] which uses the mean of the descriptors:

$$\hat{\boldsymbol{d}}_0 = \frac{1}{K+1} \sum_{i=0}^{K} \boldsymbol{d}_{r_i}. \tag{2.3}$$

In α-weighted query expansion (αQE) [13] the database descriptors are weighted by their cosine similarity with the query descriptor raised to the power of a parameter α:

$$\hat{\boldsymbol{d}}_0 = \frac{1}{\sum_{i=0}^{K} (s(\boldsymbol{d}_0, \boldsymbol{d}_{r_i}))^\alpha} \sum_{i=0}^{K} (s(\boldsymbol{d}_0, \boldsymbol{d}_{r_i}))^\alpha \boldsymbol{d}_{r_i}. \tag{2.4}$$

The expansion weights can also be learned from data like in the LAttQE model [4].

2.2.2 SuperGlobal

In the re-ranking stage of SuperGlobal [16] the descriptors of the top $\boldsymbol{K}$ database images are refined by taking a weighted average over their respective $\boldsymbol{P}$ nearest neighbors:

$$\tilde{\boldsymbol{d}}_{r_i} = \frac{\boldsymbol{d}_{r_i} + \sum_{j=1}^{P} s(\boldsymbol{d}_{r_i}, \boldsymbol{d}_{r_j^i}) \beta \boldsymbol{d}_{r_j^i}}{1 + \sum_{j=1}^{P} s(\boldsymbol{d}_{r_i}, \boldsymbol{d}_{r_j^i}) \beta}, \quad 1 \leq i \leq K. \tag{2.5}$$

Here $\tilde{\boldsymbol{d}}_{r_i}$ is the refined descriptor, $r_1^i, ..., r_P^i$ the indices of the nearest neighbors of image $\boldsymbol{I}_{r_i}$ and β a scale factor for the similarity values. Element-wise max pooling of the top $\boldsymbol{P}$ database images is used to refine the query image descriptor. With $d_{i,j}$ denoting the j:th element of the vector $\boldsymbol{d}_i$ it can be written as

$$\tilde{\boldsymbol{d}}_0 = \begin{bmatrix} \max_{1 \leq i \leq P} \tilde{d}_{r_i,1} & \max_{1 \leq i \leq P} \tilde{d}_{r_i,2} & \cdots & \max_{1 \leq i \leq P} \tilde{d}_{r_i,d} \end{bmatrix}^T. \tag{2.6}$$

The database images are re-ranked according to the averaged similarity score

$$\bar{s}_i = \frac{s(\boldsymbol{d}_0, \tilde{\boldsymbol{d}}_{r_i}) + s(\tilde{\boldsymbol{d}}_0, \tilde{\boldsymbol{d}}_{r_i})}{2}, \quad 1 \leq i \leq K. \tag{2.7}$$

2.2.3 Geometric Verification

Another way to re-rank the retrieved database images is by geometric verification [12]. Local point features are extracted from the query and database images and feature matching with RANSAC [3] is used to estimate a geometric transform between each query-database image pair. Re-ranking is done based on the number of inlier features. Geometric verification is considered to be a relatively expensive method as it requires both local feature extraction and pairwise matching.

2.2.4 Neighborhood Similarity

Other methods [20] [11] consider the neighborhood around the given descriptors in feature space and re-ranks based on neighborhood similarity. For instance if a database image descriptor shares many neighbors with the query descriptor it is typically more likely to be relevant for the query. The methods generate a new set of descriptors $\{\tilde{\boldsymbol{d}}_{r_i}\}_{i=0}^K$ and the database images are re-ordered according their similarity with the query in this embedding space.

2.3 Affinity Features

The affinity feature [11] of an image is a vector with the cosine similarities between its descriptor and those of a set of anchor images. Now, assume that we want to compute affinity features for a query image and its top $\boldsymbol{K}$ matches in the database. As in [11] the top $\boldsymbol{L}$ matches of the query in the database, and the query image itself, are used as anchors. The affinity feature can be expressed as

$$\boldsymbol{a}_i = \begin{bmatrix} s(\boldsymbol{d}_{r_i}, \boldsymbol{d}_{r_0}) & s(\boldsymbol{d}_{r_i}, \boldsymbol{d}_{r_1}) & \cdots & s(\boldsymbol{d}_{r_i}, \boldsymbol{d}_{r_L}) \end{bmatrix}^T, \quad 0 \leq i \leq K. \tag{2.8}$$

The vector $\boldsymbol{a}_i$ is hence of dimension $\boldsymbol{L+1}$ and represents the visual similarity with images in a neighborhood around the query image $\boldsymbol{I}_0$. This information is very informative for the task of re-ranking.

2.3.1 Positional Affinity

If the positions of the database images are known the affinity concept can naturally be extended with positional affinity, which captures the geometrical properties instead of the visual. With $p(I_i, I_j)$ being a measure of the positional affinity of two database images the positional affinity vector is

$$\boldsymbol{p}_i = \begin{bmatrix} p(I_{r_i}, I_{r_1}) & p(I_{r_i}, I_{r_2}) & \cdots & p(I_{r_i}, I_{r_L}) \end{bmatrix}^T, \quad 1 \leq i \leq K. \tag{2.9}$$

For the query image we simply set $\boldsymbol{p}_0 = \mathbf{0}$. To create the full affinity vector $\boldsymbol{p}_i$ is appended onto $\boldsymbol{a}_i$. See section 3.1 for an example of how $p(I_i, I_j)$ can be computed for a dataset with positional information.

2.4 Graph Neural Networks

A GNN is a special type of neural network where each node has a feature that is updated via message passing from its neighbors [19]. In this work the nodes of the graph are the query and top K database images and the graph is fully connected, meaning that all nodes are connected to each other (see figure 2.1). The node feature $\boldsymbol{x}_i$ is initially set to the affinity vector $\boldsymbol{a}_i$ of the image (potentially with included positional affinity), and the feature is gradually refined by exchanging information with the other nodes.

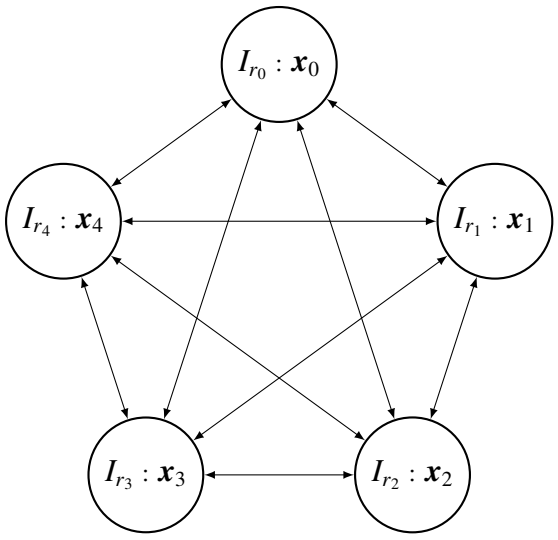

Figure 2.1: A graph neural network for re-ranking. The nodes represent the query and top $K = 4$ database images. Each node has a feature $\boldsymbol{x}_i$, initially set to the affinity vector $\boldsymbol{a}_i$, that is refined by message passing, symbolized by the arrows.

2.4.1 Attention and Message Passing

A fast and powerful technique for message passing is scaled dot-product attention [17]. A number of query, key and value vectors are packed into matrices $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$, respectively,

and the attention is computed as

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{2.10}$$

where d_k is the dimension of the keys. The softmax function, defined by

$$\text{softmax}(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{m} e^{z_j}}, \quad 1 \leq i \leq m \tag{2.11}$$

for a vector $\boldsymbol{z} = \begin{bmatrix} z_1 & z_2 & \cdots & z_m \end{bmatrix}^T$, is applied row-wise. The attention can be split into h heads where the queries, keys and values are projected to a lower dimension via learned projection matrices $\boldsymbol{W}_i^Q$, $\boldsymbol{W}_i^K$ and $\boldsymbol{W}_i^V$. The projections are passed through the attention function (2.10) followed by concatenation of the outputs of all heads and multiplication by another learned projection matrix $\boldsymbol{W}^O$:

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Concat}(\text{head}_1, ..., \text{head}_h)\boldsymbol{W}^O, \tag{2.12}$$

where

$$\text{head}_i = \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V), \quad 1 \leq i \leq h. \tag{2.13}$$

In the case of our GNN the queries, keys and values are all equal to the node feature vectors $\boldsymbol{x}_i$ (so-called self-attention):

$$\boldsymbol{Q} = \boldsymbol{K} = \boldsymbol{V} = \begin{bmatrix} \boldsymbol{x}_0 & \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_K \end{bmatrix}^T. \tag{2.14}$$

The message $\boldsymbol{m}_i$ is given by the i:th row of the multi-head attention (2.12). A node feature is updated by adding the message:

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \boldsymbol{m}_i. \tag{2.15}$$

The GNN has one or more layers of self-attention, each with its own set of learned projection matrices.

2.5 Evaluation Metrics

There are several relevant metrics for evaluating the performance of an image retrieval system. Let $\boldsymbol{y}$ be a vector of length N with elements $y_i \in \{0, 1\}$ indicating whether image $\boldsymbol{I}_i$ in the database is a relevant match for a query image and let $r_1, ..., r_K$ denote the indices of the top K images retrieved by the system. The number of relevant database images is

$$n = \sum_{i=1}^{N} y_i, \tag{2.16}$$

and the recall of the first k ($1 \leq k \leq K$) images is calculated as

$$\text{Recall@}k = \frac{1}{n} \sum_{i=1}^{k} y_{r_i}. \tag{2.17}$$

It measures the fraction of relevant images that have been retrieved. Computing the proportion of relevant matches among the first k images gives the precision:

$$\text{Precision@}k = \frac{1}{k} \sum_{i=1}^{k} y_{r_i}. \tag{2.18}$$

The average precision (AP) is the mean of the precision, but only terms corresponding to a positive match in the database are included:

$$\text{AP@}k = \frac{1}{n} \sum_{i=1}^{k} y_{r_i} \text{Precision@}i. \tag{2.19}$$

It is common to use a slight modification of equation (2.19) to better handle the case where $k < n$:

$$\text{AP@}k = \frac{1}{\min(k, n)} \sum_{i=1}^{k} y_{r_i} \text{Precision@}i. \tag{2.20}$$

Lastly the mean average precision (mAP) is the AP averaged over a set of M query images:

$$\text{mAP@}k = \frac{1}{M} \sum_{i=1}^{M} \text{AP}_i\text{@}k. \tag{2.21}$$

Here $\text{AP}_i\text{@}k$ is the average precision at k for query image i.

2.6 AP Loss

The average precision as defined by equation (2.19) is not differentiable and hence cannot be used directly to train a neural network. It is however possible to approximate it with a smooth, quantized version [14].

For brevity let R_k and P_k denote the Recall@k and Precision@k respectively and let the change in recall from $k-1$ to k be denoted

$$\Delta R_k = R_k - R_{k-1} = \frac{y_{r_k}}{n}. \tag{2.22}$$

The average precision at N can now be written as

$$\text{AP@}N = \sum_{i=1}^{N} P_i \Delta R_i. \tag{2.23}$$

Next, we note that the element y_{r_i} indicating whether the retrieved image at the i:th position is relevant can be computed as

$$y_{r_i} = \sum_{j=1}^{N} y_j \mathbb{1}[r_i = j], \tag{2.24}$$

where $\mathbb{1}$ is the indicator function and it follows that

$$P_k = \frac{1}{k} \sum_{i=1}^{k} \sum_{j=1}^{N} y_j \mathbb{1}[r_i = j], \tag{2.25}$$

$$\Delta R_k = \frac{1}{n} \sum_{j=1}^{N} y_j \mathbb{1}[r_k = j]. \tag{2.26}$$

The quantization is performed by splitting the interval $[-1, 1]$, i.e. the range of the cosine similarity, into B partially overlapping bins and computing the precision and incremental recall at each bin. A bin i, $1 \leq i \leq B$, has its center at $b_i = 1 - (i-1)\Delta^{bin}$ and covers the interval

$$\bar{b_i} = [b_i - \Delta^{bin}, b_i + \Delta^{bin}], \tag{2.27}$$

where $\Delta^{bin} = 2/(B-1)$ is half the bin width. With $s_i = s(\boldsymbol{d}_0, \boldsymbol{d}_i)$ being the similarity between the query and database image i the binned precision measure is given by

$$P_k^{bin} = \frac{\sum_{i=1}^{k} \sum_{j=1}^{N} y_j \mathbb{1}[s_j \in \bar{b_i}]}{\sum_{i=1}^{k} \sum_{j=1}^{N} \mathbb{1}[s_j \in \bar{b_i}]}. \tag{2.28}$$

The definition is analogous to that of equation (2.25) but instead of counting the number of relevant database images out of the top k ranked it computes the precision in the top k bins, which are given in descending order of similarity. The corresponding binned incremental recall function is

$$\Delta R_k^{bin} = \frac{1}{n} \sum_{j=1}^{N} y_j \mathbb{1}[s_j \in \bar{b_k}]. \tag{2.29}$$

By replacing the indicator function with the triangle function

$$\Lambda_i(x) = \max\left(1 - \frac{|x - b_i|}{\Delta^{bin}}, 0\right) \tag{2.30}$$

$\boldsymbol{P}_k^{bin}$ and $\Delta\boldsymbol{R}_k^{bin}$ can be made differentiable. If we by $\boldsymbol{\Lambda}_i(\boldsymbol{s})$ denote the vector

$$\boldsymbol{\Lambda}_i(\boldsymbol{s}) = \begin{bmatrix}\Lambda_i(s_1) & \Lambda_i(s_2) & \cdots & \Lambda_i(s_N)\end{bmatrix}^T \tag{2.31}$$

representing the soft assignment of the cosine similarities to bin $\boldsymbol{i}$ the quantized average precision can now be calculated as

$$\mathrm{AP}_Q@N = \sum_{i=1}^{B} \hat{P}_k^{bin} \Delta\hat{R}_k^{bin}, \tag{2.32}$$

where

$$\hat{P}_k^{bin} = \frac{\sum_{i=1}^{k} \boldsymbol{\Lambda}_i(\boldsymbol{s})^T \boldsymbol{y}}{\sum_{i=1}^{k} \boldsymbol{\Lambda}_i(\boldsymbol{s})^T \boldsymbol{1}}, \tag{2.33}$$

$$\Delta\hat{R}_k^{bin} = \frac{\boldsymbol{\Lambda}_k(\boldsymbol{s})^T \boldsymbol{y}}{n}. \tag{2.34}$$

In practice the neural network is trained to maximize $\mathrm{AP}_Q@K$ rather than $\mathrm{AP}_Q@N$ by selecting the appropriate subsets of $\boldsymbol{\Lambda}(\boldsymbol{s})$ and $\boldsymbol{y}$.

2.7 Contrastive and MSE losses

The re-ranking model [11] is trained with the contrastive loss

$$L_C = -\ln\left(\frac{\sum_{i=1}^{K} y_{r_i} \exp(s(\boldsymbol{d}_0, \boldsymbol{d}_{r_i})/\tau)}{\sum_{i=1}^{K} \exp(s(\boldsymbol{d}_0, \boldsymbol{d}_{r_i})/\tau)}\right), \tag{2.35}$$

where τ is a temperature parameter. It is combined with the mean squared error (MSE) loss

$$L_M = \sum_{i=0}^{K} \|\boldsymbol{a}_i - \mathrm{MLP}(\tilde{\boldsymbol{d}}_{r_i})\|^2 \tag{2.36}$$

to form the full loss function

$$L = L_C + \lambda L_M. \tag{2.37}$$

MLP denotes a multilayer perceptron with two layers and a Gaussian error linear unit (GELU) [5] non-linearity and λ is a scale factor for the MSE loss.

Chapter 3

Method

In this chapter the implementation details of the GNN are given. First the dataset used to train and evaluate the network is described. Then follows a recount of the network architecture. The chapter ends with an overview of the training process.

3.1 Dataset

Mapillary Street-Level Sequences (SLS) [18] is a large dataset consisting of 1.53 million street-level images, 640 pixels in width. The images have been collected in 30 different cities which are split into training (22 cities), validation (2 cities) and testing (6 cities) categories. For each city the images are divided into queries and a database, see table 3.1. The training set contains 93.7% of the total number of images compared to 2.0% and 4.3% for the validation and test set, respectively. Figure 3.1 displays two example images from the dataset, taken in Amsterdam.

The images of the training and validation cities are geo-located with GPS position (x, y) and heading angle α but for the test cities this data is not available. Instead an online evaluation service [10] is used for the test set. The criteria stated in [18] for determining if a database image is a positive match for a given query image given is that the distance between them should be less than 25 m and the difference in heading angle smaller than 40º.

The 2D field-of-view (FoV) overlap between images [7] is chosen as the positional affinity measure $p(I_i, I_j)$ for the dataset. Following [7] we use FoV radius $r = 50$ m and angle $\theta = 90$º. Figure 3.2 shows the parametrization of the FoV and the overlap between two nearby images. The positional affinity is computed by dividing the area of the overlap with the FoV area for a single image, resulting in a value in the $[0, 1]$ range.

Category	City	Database images	Query images
Training	Trondheim	5015	4136
	London	3291	2692
	Boston	14 024	6724
	Melbourne	101 827	88 118
	Amsterdam	11 539	7893
	Helsinki	33 248	15 228
	Tokyo	34 823	26 310
	Toronto	12 789	7352
	São Paulo	35 096	18 989
	Moscow	171 878	77 496
	Zürich	2991	2193
	Paris	9503	8480
	Bangkok	74 620	40 125
	Budapest	153 321	45 800
	Austin	28 462	14 222
	Berlin	42 965	28 197
	Ottawa	69 756	53 517
	Phoenix	106 221	50 243
	Goa	5722	5362
	Amman	953	835
	Nairobi	437	427
	Manila	6064	5378
Val.	Copenhagen	12 601	6595
	San Francisco	6315	4525
Testing	Miami	5900	5673
	Athens	3355	2466
	Buenos Aires	3710	3238
	Stockholm	12 007	6819
	Bengaluru	11 731	7026
	Kampala	2067	1870
	Total	982 231	547 929

Table 3.1: Number of images per city in Mapillary SLS.

3.2 Network Architecture

The network design largely follows that of [11]. As input the network takes the set of image descriptors $\{\boldsymbol{d}_i\}_{i=0}^{\max(K,L)}$ and optionally the positional affinity vectors $\{\boldsymbol{p}_i\}_{i=1}^{K}$. The output is a set of refined descriptors $\{\tilde{\boldsymbol{d}}_i\}_{i=0}^{K}$.

First the descriptors are passed through a fully connected layer followed by L_2 normalization before computing the affinity features $\{\boldsymbol{a}_i\}_{i=0}^{K}$ as given by equation (2.8). This fully connected layer is not included in the model [11]. If positional affinity is used then each $\boldsymbol{p}_i$ is concatenated onto the corresponding $\boldsymbol{a}_i$:

$$\boldsymbol{a}_i \leftarrow [\boldsymbol{a}_i \,\|\, \boldsymbol{p}_i]\,. \tag{3.1}$$


Figure 3.1: Two example images from the Mapillary SLS dataset [18].

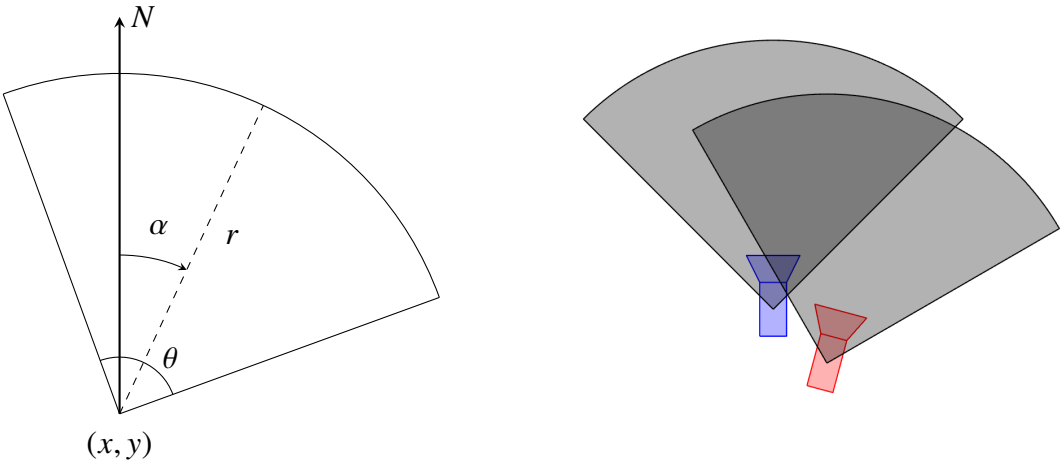

Figure 3.2: Left: The 2D field-of-view is parametrized by the position (x, y), heading angle α (clock-wise from north), FoV radius r and FoV angle θ. Right: A FoV overlap of 49.2% between two images.

Next follows another fully connected layer after which features are refined by message passing in a graph with $K + 1$ nodes, where the node features are initialized with the affinity vector:

$$\boldsymbol{x}_i^0 \leftarrow \boldsymbol{a}_i. \tag{3.2}$$

Now let $\boldsymbol{X}^l$ be the node features packed into a matrix:

$$\boldsymbol{X}^l = \begin{bmatrix} \boldsymbol{x}_0^l & \boldsymbol{x}_1^l & \cdots & \boldsymbol{x}_K^l \end{bmatrix}^T. \tag{3.3}$$

In each layer of the GNN the messages are computed using multi-head self-attention:

$$\boldsymbol{M}^l = \text{MultiHead}(\boldsymbol{X}^l, \boldsymbol{X}^l, \boldsymbol{X}^l). \tag{3.4}$$

With $\boldsymbol{m}_i^l$ being the i:th row of $\boldsymbol{M}^l$ the node feature is updated as

$$\boldsymbol{x}_i^{l+1} \leftarrow \boldsymbol{x}_i^l + \mathrm{MLP}(\boldsymbol{x}_i^l + \boldsymbol{m}_i^l) \tag{3.5}$$

where MLP is a multilayer perceptron with two layers and the GELU activation function. The output of the final GNN layer is L_2 normalized to create the output of the whole network $\{\tilde{\boldsymbol{d}}_{r_i}\}_{i=0}^K$.

Re-ranking is performed by re-ordering the database images according to the cosine similarity between the refined query descriptor $\tilde{\boldsymbol{d}}_0$ and the refined database image descriptors $\{\tilde{\boldsymbol{d}}_{r_i}\}_{i=1}^K$.

The network is trained with the quantized AP loss function (section 2.6) instead of the combination of contrastive and mean squared error (MSE) losses used in [11] (section 2.7). As a result the multilayer perceptron applied to the refined descriptors when computing the MSE loss can be removed, reducing the number of parameters.

3.3 Training Details

The GNN is trained using the Adam [6] optimizer and the quantized AP loss function. Binary labels for the loss function are created by thresholding the positional affinity $p(I_0, I_i)$ between the query and the database images. If the affinity is greater than $1/3$ then the database image is considered relevant for the query.

NetVLAD [1] is chosen as the global descriptor and is of dimension $d = 4096$. The descriptors are extracted beforehand using the hloc toolbox [15] and are kept in memory during training. The output dimensions of the two fully connected layers are set to 512 and 768 respectively, leading to refined descriptors $\tilde{\boldsymbol{d}}_i$ of dimension 768.

The number of database images to re-rank is $K = 319$ and affinity vectors are computed from the top $L = 127$ images (see section 4.3). The GNN has one layer with 12 heads in the multi-head attention unit. Dropout is applied with probability 0.2 to the input of the network and also to the attention weights. The learning rate is initialized to 10^{-4} and is multiplied by 0.9 after every epoch and a batch size of 32 is used.

The network is trained in two steps. First the initial fully connected layer is trained separately, with $K = 255$, and the loss is applied directly to its output (after L_2 normalization). Next the weights of the layer are fixed and the rest of network is trained. The fully connected layer is trained for 10 epochs and the GNN for 5 epochs. All other settings are identical in the two training steps.

In total the neural network has 9.4 million trainable parameters. Pre-training the fully connected layer, with 2.1 million parameters, takes around 3 hours on a NVIDIA TITAN V GPU. Without positional affinity the second step is finished in approximately 4 hours. Computing pair-wise positional affinity between database images is CPU intensive and adds significantly to the training time: with positional affinity the second phase takes roughly 16 hours.

Chapter 4

Evaluation

This chapter holds the evaluation of the proposed GNN re-ranking method. First the experimental setup is described. Next the results are presented and the GNN is compared against the baseline of image retrieval without re-ranking and three competing re-ranking methods. The design choices of the GNN are validated in a number of ablation experiments, followed by a concluding discussion and suggestions for future work in the domain.

4.1 Experimental Setup

The graph neural network is trained on the Mapillary SLS training set and evaluated on the validation and test sets. As outlined in section 3.3 the network is trained for 10 or 5 epochs however the weights are saved after each epoch and we choose the ones maximizing mAP@10 on the validation set.

Panorama images are excluded from both training and evaluation, as is done in Mapillary's own code [8]. The number of panorama images in the training and validation sets combined is just 16 437 or 1.1%.

A database image is considered relevant to the query if it is within 25 m. The angle criterion mentioned in section 3.1 is not used. This is again in line with Mapillary's code [8]. It is not known if the online evaluation website [10] considers the heading angle or not.

GPS positions and heading angles for the test set database images are downloaded with an API [9] provided by Mapillary. These attributes are not given as a part of Mapillary SLS but are needed to compute the positional affinity.

The main evaluation metric is mAP@k but we also report Recall@k, at $k = 1, 5, 10, 20$. The average precision is computed using equation (2.20). For consistency the definition of recall

from [8] is adopted:

$$\text{Recall@}k = \min\left(\sum_{i=1}^{k} y_{r_i}, 1\right). \tag{4.1}$$

Note how this is different from equation (2.17), being equal to one if there is any relevant database image among the top k retrieved and zero otherwise. With this definition Recall@1 is equal to mAP@1 and only the latter is reported.

Queries with no relevant database images ($n = 0$) are excluded when computing the metrics. In contrast to [18] we use all other query images and not just the center frame in each sequence. This only applies to the validation set, for the test set the online evaluation service [10] is employed.

Image retrieval without re-ranking (i.e. simply taking the top K database images based on cosine similarity) is used as the baseline. Additionally we compare against the query expansion methods AQE and αQE and the re-ranking procedure of SuperGlobal.

A grid search is performed to find the best values of QE parameters K and α. The ranges used in the search are $1 \leq K \leq 5$ and $1 \leq \alpha \leq 3$ (for αQE, in steps of 0.5) and the parameters maximizing mAP@10 for the validation set are selected. For AQE the result is $K = 1$ and for αQE $K = 1, \alpha = 1$.

For SuperGlobal we try all combinations of $K \in \{200, 400, 800\}$, $P \in \{1, 2, 3, 4\}$ and $\beta \in \{0.15, 0.3, 0.6\}$ and again choose the settings that maximize mAP@10 on the validation set. The optimal parameters in this grid search are $K = 800$, $P = 1$ and $\beta = 0.6$.

4.2 Results

Table 4.1 contains the results for the Mapillary SLS validation set. By using query expansion techniques the mAP can be improved significantly compared to the baseline of no re-ranking. However the methods suffer from low recall. SuperGlobal achieves similar precision and recall to these methods.

Training only the initial fully connected (FC) layer as described in section 3.3 gives a large improvement over the baseline, raising the mAP and recall by more than 10 percentage points. It also performs better than the QE methods.

The graph neural network is trained both without (GNN) and with (GNN+PA) positional affinity. Both versions get notably higher mAP compared to the fully connected layer, and we can see that using positional affinity is beneficial.

The results for the test set (table 4.2) paint a similar picture. Both QE methods and SuperGlobal improve the precision but at the cost of lower recall. The simple model with only a fully connected layer achieves considerably higher mAP than AQE, αQE and SuperGlobal but also has greater recall than the baseline. A GNN without positional affinity beats the fully connected layer on all metrics but by a smaller margin than on the validation set. Again

the GNN with positional affinity attains the highest precision, with mAP@10 almost twice that of the baseline.

Figure 4.1 shows two examples of GNN re-ranking. In the first example there is only one relevant image among the top five retrieved from the database. After re-ranking the top database images with a GNN the first five are all relevant. The second example also illustrates a case with a single relevant database images in the top five, but after re-ranking it has been moved down and no relevant images remain. A GNN with positional affinity was used to generate both examples.

	mAP				Recall		
Method	@1	@5	@10	@20	@5	@10	@20
No re-ranking	55.9%	36.0%	30.8%	29.5%	68.1%	72.6%	77.0%
AQE [2]	55.9%	43.0%	38.0%	36.6%	60.3%	62.5%	65.1%
αQE [13]	55.9%	41.0%	35.7%	34.3%	62.7%	67.1%	72.0%
SuperGlobal [16]	54.6%	41.5%	37.3%	36.1%	63.1%	65.1%	67.0%
FC	67.4%	47.3%	41.9%	40.4%	80.1%	**83.6%**	**86.4%**
GNN	68.5%	52.3%	47.5%	46.1%	**80.3%**	**83.6%**	86.3%
GNN+PA	**69.3%**	**60.4%**	**55.4%**	**52.9%**	78.3%	81.8%	85.0%

Table 4.1: Results on the Mapillary SLS validation set.

	mAP				Recall		
Method	@1	@5	@10	@20	@5	@10	@20
No re-ranking	34.5%	22.4%	19.7%	18.6%	45.4%	50.7%	55.8%
AQE [2]	34.5%	27.5%	24.4%	23.1%	39.0%	41.4%	44.5%
αQE [13]	34.5%	25.8%	22.7%	21.5%	42.2%	46.1%	51.1%
SuperGlobal [16]	33.5%	26.0%	23.8%	22.6%	40.7%	42.3%	45.4%
FC	46.7%	32.2%	28.8%	27.4%	60.9%	**67.3%**	72.1%
GNN	48.1%	35.6%	32.7%	31.6%	**61.8%**	**67.3%**	72.2%
GNN+PA	**48.4%**	**40.9%**	**38.2%**	**36.8%**	59.7%	65.3%	**72.3%**

Table 4.2: Results on the Mapillary SLS test set.

4.3 Ablation Study

Figure 4.2 (left) shows how the mAP varies when the value of K is changed. Increasing K, i.e. letting the network re-rank a larger number of database images, generally improves the precision up to $K = 319$. Using a too small K leads to significantly lower mAP (except for mAP@1).

The network performance is less sensitive to the choice of L, as seen in figure 4.2 (right). For example the mAP@10 is within 1% of the maximum for all values of L tested.

Adding more layers to the GNN does not improve precision (figure 4.3). The number of parameters also grows quickly with the number of layers: the networks with one, two and

three layers have 9.4, 16.5 and 23.6 million parameters, respectively. We therefore choose the variant with a single layer.

Table 4.3 displays the effect of training the network with different loss functions. The quantized AP loss improves both precision and recall greatly compared to the combination of contrastive and mean squared error losses utilized in [11]. We use the same parameter values as in that work, $\tau = 2$ and $\lambda = 0.2$.

The importance of the initial fully connected layer and the two-step training process is validated in table 4.4. Here "FC" indicates whether the model includes the layer and "Locked weights" specifies whether the weights are locked in the second training step. The precision and recall both decrease by around two percentage points if the weights of the fully connected layer are not locked while training the GNN attention unit. Excluding the layer results in an even larger drop in the evaluation metrics.

GNNs with positional affinity were used for all ablation experiments. The results are reported on the Mapillary SLS validation set.

	mAP				Recall		
Loss function	@1	@5	@10	@20	@5	@10	@20
AP	**69.3%**	**60.4%**	**55.4%**	**52.9%**	**78.3%**	**81.8%**	**85.0%**
Contrastive+MSE	48.9%	44.0%	42.3%	41.5%	68.9%	73.7%	77.5%

Table 4.3: The effect of training with different loss functions.

		mAP				Recall		
FC	**Locked weights**	@1	@5	@10	@20	@5	@10	@20
Yes	Yes	**69.3%**	**60.4%**	**55.4%**	**52.9%**	**78.3%**	**81.8%**	**85.0%**
Yes	No	67.2%	58.7%	54.0%	51.7%	76.1%	80.0%	83.6%
No	–	60.9%	53.9%	49.7%	47.7%	69.1%	73.5%	77.3%

Table 4.4: Ablation experiments for the initial fully connected layer.

4.4 Discussion

In this thesis we have studied the problem of image retrieval re-ranking using a graph neural network. A GNN design based on existing work [11] was simplified by using a more effective loss function and the concept of affinity features extended to incorporate positional affinity. The GNN was trained and evaluated on the large-scale Mapillary SLS dataset.

The goal of the thesis, "to implement and train the proposed GNN and to evaluate it against other methods", can thus be be considered fulfilled. The evaluation shows that GNN re-ranking improves the accuracy significantly both in terms of precision and recall, compared to the baseline of not re-ranking the database images and also compared to standard query expansion methods and the re-ranking procedure of SuperGlobal.

A somewhat suprising result is that the fully connected layer performs so well. With relatively few parameters and short training time it can be considered an light-weight alternative to the full GNN model, although with lower accuracy.

Adding positional affinity to the affinity vector lets the network utilize the spatial relationship between database images. Evaluation on the Mapillary SLS validation and test sets shows that this further boosts the re-ranking precision.

The advantage of directly optimizing the average precision, using the quantized AP loss, can be seen in the experimental results of table 4.3. By having a single loss function instead of combining the contrastive and MSE losses we also avoid the problem of weighting the two losses.

4.5 Future Work

There are many aspects of the re-ranking problem that would be interesting to study in more detail. One example is the generalization performance of the GNN. Can a network trained on Mapillary SLS be used to re-rank images from another dataset? Perhaps re-training only the initial fully connected layer is sufficient to make it work on the new data.

Another direction could be to explore additional types of positional affinity measures. In [7] a 3D field-of-view overlap metric based on the number of shared points in a sparse 3D reconstruction is considered for an indoor dataset. It should be straight-forward to use as the positional affinity $p(I_i, I_j)$ for the GNN.

The GNN model presented in the thesis has potential to exploit other types of sensor data. For example if the heading angle is known for the query image, which is a reasonable assumption in some scenarios, one can compute a type of "heading affinity" between the query and database images. The heading information is then appended onto the affinity vector the same way as the positional affinity.

Future work might also include a more comprehensive evaluation of the proposed GNN re-ranking method. In this work it is only compared to the relatively simple query expansion techniques AQE and αQE and the re-ranking stage of SuperGlobal. It would be interesting to see how well it performs relative to other learned methods like LAttQE [4], or comparing it to a NetVLAD [1] model fine-tuned to Mapillary SLS.

Figure 4.1: Two examples of GNN re-ranking from the Mapillary SLS dataset [18]. Database images relevant to the query are marked with a green border.

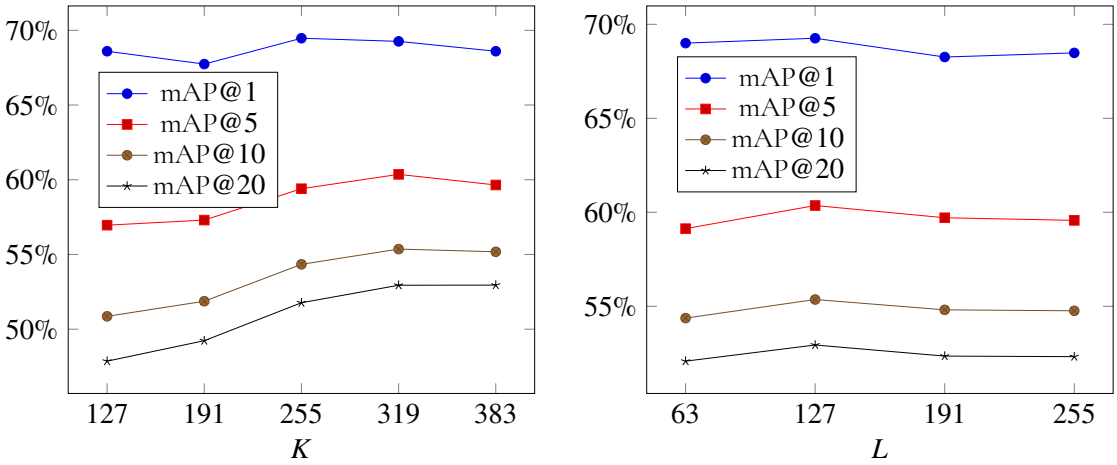

Figure 4.2: Ablation results for K and L. Left: Varying K while keeping $L = 127$ fixed. Right: Varying L while keeping $K = 319$ fixed.

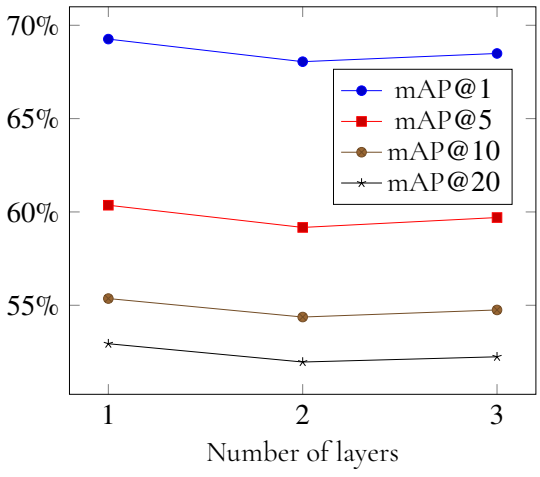

Figure 4.3: Ablation results for the number of layers in the GNN. Models are trained with $K = 319$ and $L = 127$.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[2] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[3] Martin A Fischler and Robert C Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[4] Albert Gordo, Filip Radenovic, and Tamara Berg. Attention-Based Query Expansion Learning. In *European Conference on Computer Vision*, pages 172–188. Springer, 2020.

[5] Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[6] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] María Leyva-Vallina, Nicola Strisciuglio, and Nicolai Petkov. Generalized Contrastive Optimization of Siamese Networks for Place Recognition. *arXiv preprint arXiv:2103.06638*, 2021.

[8] Mapillary. mapillary/mapillary_sls GitHub repository. `https://github.com/mapillary/mapillary_sls`, 2020.

[9] Mapillary. Mapillary API. `https://www.mapillary.com/developer/api-documentation`, 2024.

[10] Mapillary and CodaLab. MSLS Place recognition challenge. `https://codalab.lisn.upsaclay.fr/competitions/865`, 2021.

[11] Jianbo Ouyang, Hui Wu, Min Wang, Wengang Zhou, and Houqiang Li. Contextual Similarity Aggregation with Self-attention for Visual Re-ranking. *Advances in Neural Information Processing Systems*, 34:3135–3148, 2021.

[12] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[13] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning CNN Image Retrieval with No Human Annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018.

[14] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. Learning with Average Precision: Training Image Retrieval with a Listwise Loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5107–5116, 2019.

[15] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From Coarse to Fine: Robust Hierarchical Localization at Large Scale. In *CVPR*, 2019.

[16] Shihao Shao, Kaifeng Chen, Arjun Karpur, Qinghua Cui, André Araujo, and Bingyi Cao. Global Features are All You Need for Image Retrieval and Reranking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11036–11046, 2023.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in neural information processing systems*, 30, 2017.

[18] Frederik Warburg, Soren Hauberg, Manuel Lopez-Antequera, Pau Gargallo, Yubin Kuang, and Javier Civera. Mapillary Street-Level Sequences: A Dataset for Lifelong Place Recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2626–2635, 2020.

[19] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A Comprehensive Survey on Graph Neural Networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[20] Xuanmeng Zhang, Minyue Jiang, Zhedong Zheng, Xiao Tan, Errui Ding, and Yi Yang. Understanding Image Retrieval Re-Ranking: A Graph Neural Network Perspective. *arXiv preprint arXiv:2012.07620*, 2020.

[21] Liang Zheng, Yi Yang, and Qi Tian. SIFT Meets CNN: A Decade Survey of Instance Retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1224–1244, 2017.

Master's Theses in Mathematical Sciences 2024:E18
ISSN 1404-6342

LUTFMA-3531-2024

Computer Vision and Machine Learning
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/