Foundation Models for Visual Place Recognition

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering
by Research

by

Avneesh Mishra
2021701032
avneesh.mishra@research.iiit.ac.in

International Institute of Information Technology
(Deemed to be University)
Hyderabad - 500 032, INDIA
March 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled **"Foundation Models for Visual Place Recognition"** by **Avneesh Mishra**, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Dr. K. Madhava Krishna

To Serendipity

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Madhava Krishna. His constant support and encouragement were invaluable at every step of this journey. His guidance on mobile robotics systems, SLAM, local feature methods, and related fields during the initial stages of the project proved particularly helpful. This work required significant resources (compute and time) and I thank him again to provide me with those.

I am equally grateful to my co-advisors, Sourav Garg and Krishna Murthy. Sourav Garg's expertise in local features and VPR systems was instrumental in shaping the technical direction of this work. Krishna Murthy's insightful brainstorming sessions and critical reviews greatly improved the quality of my research.

My sincere thanks to Nikhil Varma Keetha for leading this project. His clear guidance in assigning tasks and his willingness to expose me to the fascinating world of research helped me grow immensely as a researcher. I would also like to thank Jay Karhade for his significant contributions to the project's success, particularly his work on the website, dataloaders, and help in benchmarking. I thank their guide at CMU AirLab, Sebastian Scherer, for letting me collaborate on this amazing journey.

I would also like to thank staff at RRC and the IIIT Hyderabad ecosystem. Special thanks to Dr. Semparithi Aravindan for maintaining our HPC cluster Ada and to promptly help me with my queries. I owe a lot of my systems knowledge to him. I would also like to thank various organizations like GitHub, Microsoft, GNU Foundation, DuckDuckGo, Google, etc. without whose tools none of this work would be possible.

Finally, I would like to thank my mother, father, and sister, for supporting me throughout my entire life and particularly during my master's degree. It is because of their unconditional love and prayers that I stand where I am today. Lastly, I would thank God for the opportunities given to me.

Abstract

Imagine a robot navigating a complex and unfamiliar environment - underwater, subterranean, or even a dilapidated building. Current Visual Place Recognition (VPR) techniques often struggle with such diverse scenarios, requiring re-training for each new environment. This limitation hinders the development of truly autonomous robots. This work introduces AnyLoc, a novel VPR system taking a significant step towards universality. AnyLoc leverages feature representations learned by powerful foundation models, eliminating the need for VPR-specific training. Furthermore, by combining these features with unsupervised aggregation techniques, AnyLoc can uncover unique visual characteristics that define different environments. Our experiments demonstrate AnyLoc's potential to function across multiple environments (outdoor, indoor, aerial, underwater, subterranean, and dilapidated) without retraining, laying the groundwork for VPR solutions that could be deployed anywhere, anytime, and across anyview.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Foreword

Imagine a mobile robot navigating a complex environment for the first time. Limited training data often hinders its ability to recognize previously unseen locations. This challenge highlights the need for robust visual place recognition (VPR) techniques in mobile robotics. The fields of deep learning and mobile robotics are undergoing rapid advancements, with a growing focus on developing more generalized solutions. One promising approach involves *foundation models*, capable of learning transferable representations from massive datasets. These models are paving the way for tackling open-set problems, where robots can excel even in scenarios not explicitly encountered during training. This thesis investigates the potential of foundation models to improve the perception capabilities of mobile robots, focusing on the development of a novel VPR technique called AnyLoc.

AnyLoc is a novel image retrieval system that leverages latent features extracted by foundation models. It employs aggregation techniques to combine these features, resulting in robust place descriptors crucial for accurate visual place recognition.

1.1.1 Contribution

This thesis, along with the publication supporting it (AnyLoc [18]), has the following content and contributions

- Chapter 1: An overview of mobile robotics, Simultaneous Localization and Mapping (SLAM), and Visual Place Recognition (VPR) systems.
- Chapter 2: An overview of foundation models: ViT architecture, training pipeline for self-supervised learning, and the backbone models for AnyLoc.
- Chapter 3: Proposes an image retrieval method, AnyLoc, that is a new state-of-the-art on various testing scenarios, yielding twice the performance of supervised VPR benchmarks. Interestingly, it looses no performance even after a `95x` reduction in descriptor size, which is *unprecedented* in VPR.
- Chapter 4: Concludes this thesis with presenting solutions to current problems and some ideas for future development.

1.2 Understanding the Area of Contribution

This thesis introduces AnyLoc [18], a novel image retrieval system that leverages features extracted from foundation models and employs conventional aggregation techniques to create robust place descriptors. Image retrieval (IR) plays a crucial role in various applications, including:

- *Visual Place Recognition* (VPR) in Mobile Robotics [36]: VPR systems rely on IR to identify previously encountered locations, enabling tasks like navigation and loop closure within Simultaneous Localization and Mapping (SLAM). This is also the direction in which this thesis is oriented.
- Remote Sensing and *Geographic Information Systems* (GIS) [63]: IR techniques are used to analyze and classify satellite or aerial imagery, aiding tasks like land cover change detection or identifying specific features in large datasets.
- *Medical Image Retrieval* [124, 190]: In healthcare, IR helps medical professionals retrieve relevant medical images (e.g., X-rays, MRIs) based on specific criteria, assisting in diagnosis and treatment planning.
- *Content-based Image Search* [15]: IR powers search engines and applications that allow users to find similar images based on visual content, facilitating tasks like product identification or searching for visually similar items online and organizing a large album of unorganized photos.

The reminder of this chapter describes a mobile robotics system, parts of a SLAM system (Section 1.2.2), a VPR system in SLAM (Section 1.2.3), and associated methods and jargon. Chapter 2 introduces foundation models and related concepts needed for this thesis. Our method, AnyLoc, is presented in Chapter 3. We conclude with possible future directions in Chapter 4.

1.2.1 Mobile Robot Systems

The software of mobile robots is composed of various modules. Most systems comprise of the following parts (also highlighted in Fig. 1.1)

- A *data processing* module collects information from the environment through sensors and performs signal processing to get meaningful information.
- A *localization and mapping* module creates and maintains a map of the environment (in which the robot can navigate). It also localizes the robot in that map (usually as a temporal pose sequence).
- A *cognition* module transforms the map and robot poses into information that is needed for downstream tasks. For example: identifying obstacles and a goal for the task of navigating while avoiding obstacles.
- A *path planning* module contains motion planners to guide the robot through the environment. Generally, they are comprised of local planners for immediate vicinity and global planners for high-level goals.

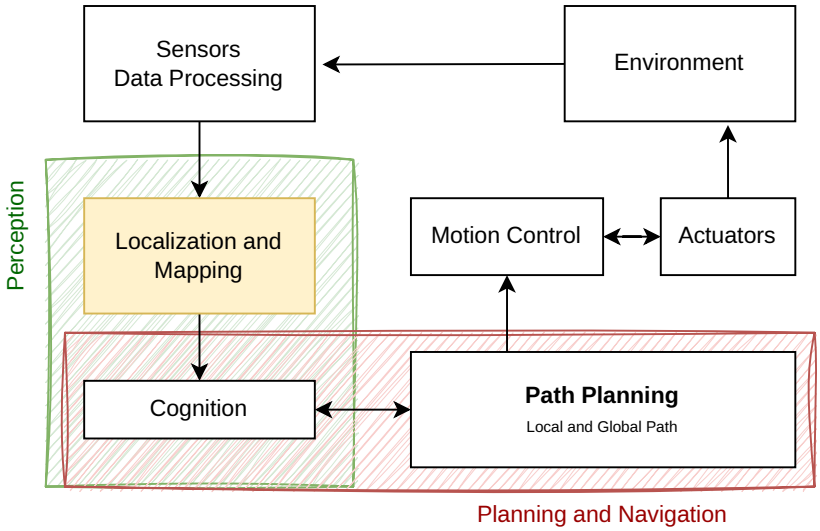


Figure 1.1 Mobile Robotics System
The *perception* stack makes sense of the environment and the *planning and navigation* stack makes the robot navigate in the environment.

- A *motion control* module is tightly coupled with *actuators* which cause motion. It usually contains a motion model to estimate the actuation values.

The *perception* stack consists of the localization and mapping and the cognition module while the *planning and navigation* stack consists of the cognition module and the path planning module. Note that the cognition module could share parts with the perception and the planning and navigation stacks.

This thesis primarily focuses on parts of the *localization and mapping* module. A larger overview of mobile robot systems can be found in [104]. A more comprehensive study of various aspects of mobile robotics can be found in [182] (for overview and background), [189] (perception), [100] (planning and navigation), [117] (navigation and obstacle avoidance), and [194] (motion control).

1.2.2 Localization and Mapping

Simultaneous Localization and Mapping (SLAM) is a type of localization and mapping system where the location of the robot (agent) and the map of the environment are jointly estimated. A review of different types of visual SLAM algorithms can be found in [32, 128, 134].

This is conventionally achieved using system models and filtering methods like Kalmann Filters (KF), Extended Kalman Filters (EKFs), particle filters, etc. Most algorithms minimize a photometric projection error to estimate the sensor's pose for a coarse localization in the map. Lately, visual SLAM algorithms employ optimization techniques on various timescales and loop closures across various distances. A system of equations, projecting objects in the map in camera frames, is solved to simultaneously obtain the camera poses and map objects (usually like sparse point cloud). This is formulated as an optimization problem on a graph of nodes [174].

A generic template for SLAM systems can be found in Fig. 1.2. Most SLAM systems can be divided into two parts

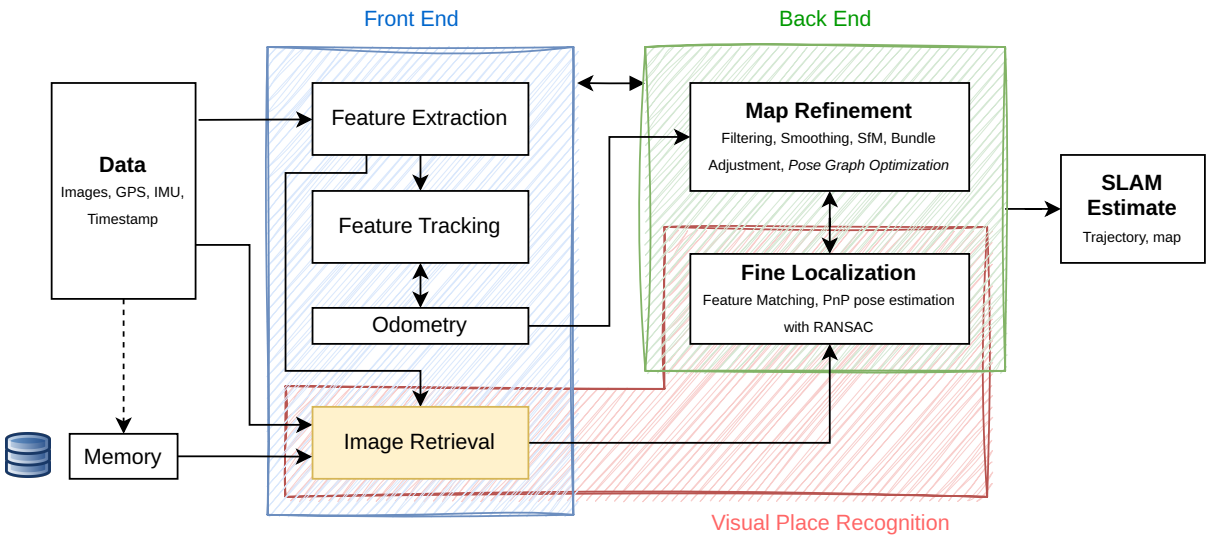


Figure 1.2 General SLAM Systems
The *front end* module performs feature extraction, feature tracking, and gets the odometry of the robot. It also does coarse visual place recognition by image retrieval. The *back end* module does fine localization (gets loop closures) and does map refinement. The map (containing the poses and the environment) is shared with both the modules.

1. A *front end* module deals with all the data that enters the system. Common parts of it are centered around
    - *Feature extraction* where keypoint features are extracted from the images
    - *Feature tracking* where keypoints are tracked between consecutive images (to get relations)
    - *Odometry* which relates two consecutive poses (locations)
    - *Image Retrieval* where an agent can recognize a previously visited part of the map (stored in its memory). This gives a coarse location of a place the agent must have revisited.
2. A *back end* module works on refining (optimizing) the map and other estimates (like trajectory). It usually consists of
    - *Fine Localization* where a more precise location of an image retrieval is estimated. This is usually done using PnP RANSAC techniques.
    - *Map Refinement* where the map estimates are fine tuned using optimizers. This is also called *map optimization*.

This thesis primarily focuses on the Visual Place Recognition (VPR) aspect of SLAM, which is further described in Section 1.2.3. The reminder of this subsection provides related details and a brief overview of SLAM systems.

1.2.2.1 Systems Similar to SLAM

Structure from Motion (SfM) aims to recover 3D geometry of a scene and the camera poses from a set of 2D images. Bundle Adjustment (BA) aims to refine the map and pose estimates through minimizing

a reprojection error. Indeed, SfM tools like COLMAP [142, 141] are used in SLAM systems as a pre-processing tool. SLAM systems usually operate online (live, in real-time). Maps used by SLAM systems are also capable of deeper environment understanding.

Visual Odometry (VO) and similar mapping techniques aim to estimate relative movement; SLAM aims to build a global and consistent map of the environment while estimating pose. VO systems are prone to drift, whereas SLAM systems correct drift through loop closures formed by the visual place recognition module.

1.2.2.2 Conventional SLAM Systems

Some conventional SLAM systems are described hereon

MonoSLAM (2007) [183] uses a known target for initialization. The system's state vector is updated by using a constant velocity model. Camera motion and environment structure (map) are estimated using an Extended Kalman Filter (EKF). The algorithm does not employ global optimization or loop closure techniques.

PTAM (2007) [185] (Parallel Tracking and Mapping) has two threads: tracking and mapping. The tracking thread updates camera poses in the frames through feature search (matching). The mapping thread performs data association, integration of new frames, local and global bundle adjustment.

FAB-MAP (2008) [180] (FAst Binary appearance-based Mapping) builds a map of visually similar locations based on the environments appearance. The visual representations for place recognition are built using a Bag-of-Words (BoW) representation over image keypoint features. It incrementally builds and updates map in a probabilistic framework.

DTAM (2011) [169] (Dense Tracking and Mapping) has two parts (like PTAM but dense). The dense tracking module estimates motion of camera by aligning a virtual camera in the recorded map. The dense mapping module estimates the (inverse) depth values of images by jointly minimizing the photometric error.

FAB-MAP 2.0 (2011) [167] builds on FAB-MAP with a sparse approximation to the model, thus allowing deployment on a much larger scale.

SVO (2014) [154] (Semi-Direct Visual Odometry) minimizes the photometric error and performs feature alignment for motion estimation. It has a probabilistic depth filter for mapping.

LSD-SLAM (2014) [153] (Large-Scale Direct monocular SLAM) minimizes the photometric error for pose estimation, it then estimates depth map (through addition in a new keyframe or by refining), finally it uses a pose-graph optimization algorithm to smooth the map. It also employs loop closures through image alignment and constraint search.

ORB-SLAM (2015) [149] is one of the first lifelong SLAM algorithms that defined the methodology of modern SLAM systems. It uses ORB features [171] for all tasks (tracking, mapping, relocalization and loop closing). It has three threads: tracking, local mapping and loop closing.

The tracking thread localizes the camera and decides when to insert a new keyframe, the local mapping thread performs local bundle adjustment (BA) and culls redundant keyframes, loop closing thread searches for loops, aligns them, and performs a pose graph optimization for global consistency.

It uses a Bag-of-Words approach (DBoW2) for place recognition and maintains a covisibility graph for the already traversed map (to facilitate faster relocalization). It is also the first SLAM algorithm to have *automatic map initialization*; achieved using parallel model computations of homography and fundamental matrix (and choosing the appropriate model by scoring).

ORB-SLAM2 (2017) [138] extends ORB-SLAM with stereo and RGB-D (depth) capabilities, giving a more accurate and dense map while maintaining the real-time characteristic of the system.

ORB-SLAM3 (2020) [76] further adds visual-inertial (IMU) and multi-map support (by maintaining an atlas for active and non-active maps). It still uses a DBoW2 based approach for place recognition, but with modifications for verification in active map and checks for gravity direction.

1.2.2.3 Deep Learning in SLAM

Review of Deep Learning in SLAM

CNN-SLAM (2017) [129] fuses depth maps predicted by a CNN with those obtained by direct monocular SLAM. It also fuses semantic segmentation maps for coherent scene reconstructions.

Neural SLAM (2017) [132] integrates deep reinforcement learning and differentiable SLAM modules for exploration tasks. It uses a neural turing machine (NTM) [155] for external storage (which stores an implicit map representation). It does not use graph optimization for the map, but instead uses weights of the NTM. It uses a search in this space (through a convolution kernel) for localization.

CNN-SVO (2018) [110] uses depth predictions as prior to reduce the uncertainty for identifying corresponding features as the camera moves.

DRIOD-SLAM (2021) [69] (Differentiable Recurrent Optimization-Inspired Design) applies iterative updates to optical flow (inspired by RAFT [90]). It also uses a differentiable Dense Bundle Adjustment (DBA) layer to update camera poses and dense per-pixel depth.

∇SLAM (2020) [98] (gradSLAM) makes dense SLAM differentiable by proposing differentiable versions of the operations. This makes SLAM end-to-end differentiable (from 3D objects in the map to the pixel observations in the sensor). Specifically, they propose differentiable LM solver, map construction, fusion, and ray backprojection. The authors demonstrate fully differentiable versions of KinectFusion [170] and PointFusion [160].

1.2.2.4 SLAM Beyond RGB Cameras

This incorporates data from other sensor modalities like depth. It also covers modern techniques like implicit representation for mapping. A review and benchmarking of these algorithms can be found in [3, 166]

PointFusion (2013) [160] uses depth maps as inputs to perform mapping. It uses pyramid-based Iterative Closest Point (ICP) alignment to estimate camera pose from points in the map and the depth image captured by the camera. Points fused in the global map go from unstable to stable as they are more frequently observed. It also identifies dynamic regions by constructing an ICP status map.

KinectFusion (2018) [170] uses raw depth from a Microsoft Kinect to map a scene using truncated signed distance function (TSDF). It uses ICP of predicted and measured surface for pose estimation.

Kimera (2021) [67] is a full-fledged SLAM system that gives high-utility perception maps. Using stereo RGB and an IMU, it builds a dynamic scene graph (DSG) that contains the environment information (map) in a multi-layered, hierarchical, and abstracted manner. Internally, it uses semantic segmentation and visual inertial odometry (VIO) methods. It comprises of VIO (for 3D pose estimation), mesher (to build local 3D meshes), semantics (for global 3D meshes), and a pose graph and mesh optimization (PGMO) library (that enforces loop closures).

Kimera-Multi (2021) [70] extends Kimera to a multi-robot setting where agents can perform inter and intra-robot loop closures and can create the map in a distributed (but networked) manner.

iMAP (2021) [68] uses an implicit neural scene representation for mapping (with data captured using a hand-held RGB-D camera). The scene reconstruction has a small memory footprint. It jointly optimizes over the camera poses for keyframes and the implicit scene network parameters. It yields more complete reconstructions as they're renderings produced by querying the network, in a process similar to Neural Radiance Fields (NeRF) [88].

NICE-SLAM (2021) [75] uses a hierarchical implicit scene representation that incorporates information on multiple levels. It applies tri-linear interpolation on a hierarchical feature grid for mapping. It uses photometric, depth, and geometric reconstruction (bundle adjustment) losses for jointly estimating the network parameters (grid and color networks) and the camera extrinsics (pose in scene).

NICER-SLAM (2023) [30] builds up on NICE-SLAM, adding losses for scene warping, optical flow, surface normals and Eikonal/SDF output (for accurate surfaces in the map). It also operates on a continuous RGB stream (not needing depth).

RO-SLAM (2023) [13] uses monocular RGB inputs to create object representations based on neural radiance fields. These are then coupled with a light-weight object SLAM (ORB-SLAM2). It trains multiple NeRF networks in parallel (for the objects in the scene).

GS-SLAM (2023) [25] uses Gaussian Splatting [20] as the scene representation. It proposes an expansion strategy to add and remove 3D gaussian representations to accommodate newly captured views. It re-renders and optimizes camera pose (for tracking) by a coarse-to-fine method. It uses bundle adjustment to jointly optimize the camera poses and the 3D Gaussian scene representation.

Gaussian-SLAM (2023) [27] performs optimization over sub-maps instead of the entire Gaussian point cloud. This gives better results for novel view synthesis, makes the computation feasible, and helps avoiding catastrophic forgetting. It uses structured similarity metric (SSIM) for color supervision (along with the usual L1 loss).

SplaTAM (2023) [19] also uses 3D Gaussian Splatting for the scene representation. It estimates the camera pose using silhouette-guided differentiable rendering. It then densifies the map by adding new Gaussians to the map. It the updates the map using RGB and depth errors.

Kimera2 (2024) [1] improves Kimera by modifying the VIO pipeline to support multiple modalities (RGB-D, stereo, monocular, and wheel odometry) and upgrades the pose graph optimization (PGO) backends with Graduated Non-Convexity (GNC) [108] for robustness against spurious loop closures.

Khronos (2024) [2] integrates a spatio-temporal approach that monitors short and long-term dynamics into SLAM pipelines, thus yielding a Spatio-temporal Metric SLAM (SMS) system. It uses semantically annotated RGBD sensor data with odometry to estimate object fragments through an active window approach. It then applies a novel factorization approach to track spatio-temporal fragments (for short-term and long-term dynamic objects in the scene).

1.2.2.5 Deployment

The nature of SLAM entails multiple processes running in parallel. Robot Operating System (ROS) [179] is a backbone that allows implementing each sub-system separately and handles communication between them. It also has useful tools for visualization and debugging. ROS 2 [42] is the latest version of ROS. Several SLAM projects and tools are collected at OpenSLAM.org, most rely on LiDAR laser scans for mapping. Some common implementations in ROS are described below

GMapping (2007) [184] (Grid Mapping) uses a particle filter for building maps using 2D lidar data. It is a widely used conventional SLAM library for small spaces. However, it fails in large spaces due to poor loop closures at scale.

KartoSLAM (2010) [176] uses sparse bundle adjustment and a pose-graph based method.

HectorSLAM (2011) [168] uses a 3D navigation filter based on EKF state estimation. It primarily uses LiDAR scan matching for generating 2D pose estimates and does not rely on odometry or loop closures, leading to high drift over time.

Cartographer (2016) [135] It combines a scan-to-submap matching method with loop closure optimization using the Ceres Solver [4]. However, it requires good odometry for good results, and the package is no longer maintained.

SLAM Toolbox (2021) [62] is the new default SLAM vendor in ROS 2 (replacing GMapping). It builds on top of Open KartoSLAM [176] with code optimizations for synchronous and asynchronous operations, multi-session mapping, better graph optimization, distributed mapping applications, etc.

1.2.3 Visual Place Recognition

In Visual Place Recognition (VPR), an agent leverages visual data to identify a previously visited location. While "place" typically denotes a broader region, it can also refer to a specific position. VPR systems typically consist of two parts (also highlighted in Fig. 1.2):

1. *Image Retrieval*: Given a database of geo-tagged images in the map, and a query image (recent observation), the agent identifies the database images that are from the same location as that of the query. This is also called *coarse localization* as it only gives the location of the retrieved database image which is an approximate location of the query in the map.

2. *Fine Localization*: Given a query and the closest relevant database images, this step aims to find the precise location of the query in the mapped scene. This is usually accomplished through finding points in map that correspond with keypoints in the query image (thereby giving a 2D-3D correspondence list). Usually, this involves geometric verification techniques like Epipolar Geometry, Homography Estimation, Perspective-n-Point solvers with RANSAC, Bundle Adjustment (BA), etc. or some combination of these [193]. Many modern implementations create differentiable formulations of the same thing and train a system to perform this stage.

As shown in Fig. 1.2, the image retrieval is associated with the front-end and the fine localizations is associated with the back end. Combining these two components provides loop closure (LC) information for pose-graph optimizers. This information is crucial for mitigating issues like drift in long-term operations covering large areas and catastrophic forgetting in maps. Most SLAM system described in Section 1.2.2 uses loop closures for improving the quality of maps. It also finds applications beyond SLAM, including Augmented Reality (AR), Virtual Reality (VR), and Mixed Reality (XR). A review of visual place recognition techniques, terminologies, and use cases can be found in [34, 64, 53, 94, 114, 137].

1.2.3.1 Image Retrieval from Image Descriptors

A typical Image Retrieval (IR) system is illustrated in Fig. 1.3. The *image processing* stage extracts a single, unique descriptor that captures the overall visual characteristics of the entire input image. From a database of representative images (and their known locations), the image processing pipeline first builds a database of global descriptors. Given N_d images of shape C, H, W each (where H, W is the size of

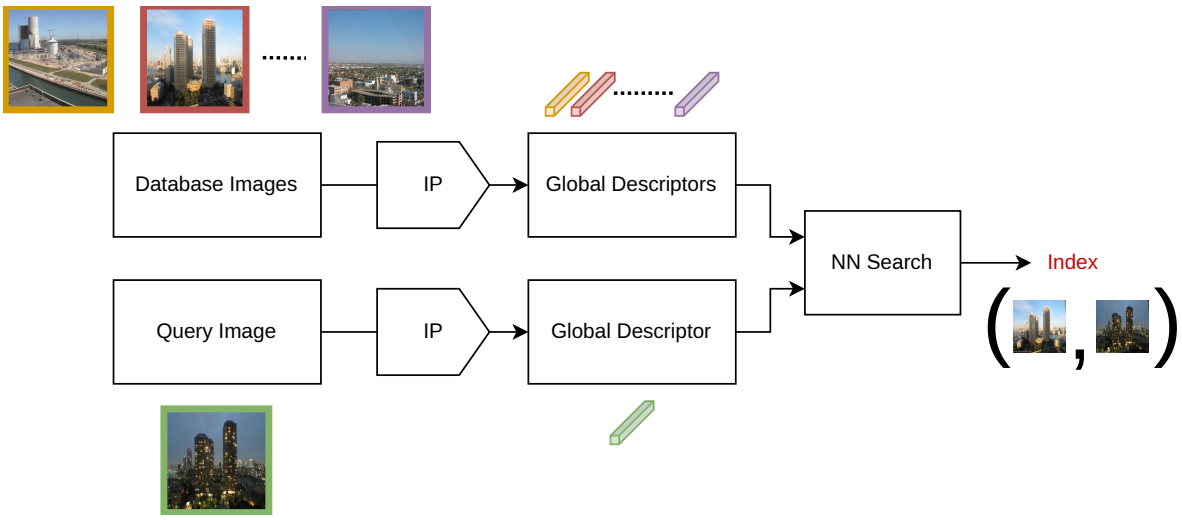


Figure 1.3 Image Retrieval Systems
An Image Processing (IP) algorithm builds *global descriptors* from input images. A nearest neighbor search takes a query descriptor and a set of global database descriptors, and it returns the database descriptor that is closest to the query descriptor (as index in set).

the image and C is the number of channels - 3 for RGB, 1 for greyscale), we apply the following steps to each image

1. *Extract* visual context: This summarizes the visual content in an image, yielding descriptors of shape N, d_p (where N is the number of descriptors and d_p is the dimensionality). There are two main techniques to creating image descriptors for extracting visual context of an image:

   (a) *Local Feature-based*: Identify and describe distinctive local features within an image, like corners, edges, or blobs. These features are known as keypoints, and each keypoint has a corresponding descriptor capturing its specific visual characteristics.

   (b) *Global Context-based*: Extract descriptors that directly represents the entire image. Unlike local feature-based methods, they do not rely on identifying and describing individual keypoints. Instead, these techniques may process the image in different ways, including dividing it into smaller overlapping or non-overlapping patches, to capture the global visual properties. This allows them to capture characteristics such as color distribution, texture patterns, and spatial relationships between image elements.

2. *Aggregate* descriptors: The previous step yields multiple descriptors (keypoints or patches), this step creates a single descriptor representing the entire image. This is usually accomplished through clustering and creating histograms. We get a d-dimensional *global* descriptor per image.

Ideally, global descriptors of similar images (taken from nearby or same places) should exhibit greater similarity compared to those from dissimilar images. These descriptors are stored in the memory of the system. When presented with a query image, the system extracts its descriptor and performs a nearest-neighbor (NN) search within the database, retrieving the most similar descriptors and their corresponding images. In some cases, the system may return multiple top-k matching image entries. The

downstream fine localization task takes the query image, database image(s) with location(s), and the registered map to further localize the query in the map.

Mathematical Formulation Let $\mathbf{I_{DB}} = \{\mathbf{I}_{DB}[1], \mathbf{I}_{DB}[2], \ldots, \mathbf{I}_{DB}[N_d]\}$ be a set of database images (where $\mathbf{I}_{DB}[i] \in \mathbb{R}^{C,H,W}$ and $\mathbf{I} \in \mathbb{R}^{N_d,C,H,W}$) and $\mathbf{D_{DB}} = \{\mathbf{d}_{DB}[1], \mathbf{d}_{DB}[2], \ldots, \mathbf{d}_{DB}[N_d]\}$ be the corresponding global descriptors obtained using $\mathbf{d}_{DB}[i] = \text{GD}(\mathbf{I}_{DB}[i])$ (where $\mathbf{d}_{DB}[i] \in \mathbb{R}^d$ and $\mathbf{D_{DB}} \in \mathbb{R}^{N_d,d}$) and where $\text{GD} : \mathbb{R}^{C,H,W} \to \mathbb{R}^d$ is comprised of the steps above (extract and aggregate).

A query image $\mathbf{I}_q \in \mathbb{R}^{C,H,W}$ is passed through the same pipeline $\text{GD}(\bullet)$ to obtain its global descriptor $\mathbf{d}_q = \text{GD}(\mathbf{I}_q)$ (where $\mathbf{d}_q \in \mathbb{R}^d$). The nearest neighbor search to retrieve the top-k most similar database descriptors is done as follows

$$\mathbf{S}_u = \left\{ \underbrace{\text{sim}\left(\mathbf{d}_{DB}[1], \mathbf{d}_q\right), \text{sim}\left(\mathbf{d}_{DB}[2], \mathbf{d}_q\right), \ldots, \text{sim}\left(\mathbf{d}_{DB}[N_d], \mathbf{d}_q\right)}_{\text{Similarity Scores}} \right\} \tag{1.1}$$

$$\mathbf{P} = \text{argsort}(\mathbf{S}_u, \text{descending}) \tag{1.2}$$

$$\mathbf{V}_{\text{top-k}} = \mathbf{P}[1:k] \tag{1.3}$$

Where $\mathbf{S}_u \in \mathbb{R}^{N_d}$ is a set of unordered distances between the query and each database descriptor calculated using a similarity function $\text{sim} : \left(\mathbb{R}^d, \mathbb{R}^d\right) \to \mathbb{R}$ which is a metric like cosine distance (0 for no similarity and 1 for high similarity). The $\text{argsort}(\bullet, \bullet)$ function takes a set and the order (`ascending` or `descending`) and returns a list of indices $\mathbf{P} \in \mathbb{N}^{N_d}$ corresponding to the order. For instance, for `descending` order, $\mathbf{S}_u[\mathbf{P}[1]]$ is the largest value in $\mathbf{S}_u$, $\mathbf{S}_u[\mathbf{P}[2]]$ is the second largest value in $\mathbf{S}_u$, and so on until $\mathbf{S}_u[\mathbf{P}[N_d]]$ is the smallest value in $\mathbf{S}_u$. We use indices instead of similarity values as we're interested in finding the closest database images and not their similarity scores. We then take the first k values to get the indices $\mathbf{V}_{\text{top-k}} \in \mathbb{N}^k$ of the top-k most similar database images.

1.2.3.2 Creating Image Descriptors

Typically, image descriptors are constructed from local features, as this approach enables focusing on informative regions within an image. However, recent advancements in computational power have facilitated the use of the entire image content (all image patches) for descriptor creation.

Local Feature Methods locate and describe keypoints in an image. These could be manually defined conventional methods like SIFT [191], SURF [186], BRIEF [173], ORB [171], AKAZE [158, 162], etc. or features that are learned like FAST [188, 181], LIFT [143], DELF [139], SuperPoint [118], L2-Net [130], R2D2 [103], D2-Net [96], OriNet [99, 123, 122], etc. A thorough benchmarking of these algorithms towards image matching is presented in [87]. These methods take an image $\mathbf{I} \in \mathbb{R}^{C,H,W}$ and return keypoints $\mathbf{K} \in \mathbb{R}^{N_k,2}$ (a list of locations of interest $\{\ldots, (h_i, w_i), \ldots\}$) and descriptors $\mathbf{D}_k \in \mathbb{R}^{N_k,d_k}$, where N_k is the number of keypoints and d_k is the dimensionality of the descriptor (for example, $d_k = 128$ for SIFT).

Global Context through Image Patches divides the input image into a set of patches and processes them individually. It takes an image $\mathbf{I} \in \mathbb{R}^{C,H,W}$ and breaks it uniformly into patches of shape $\mathbf{G} \in \mathbb{R}^{N_p,C,s_h,s_w}$ where $N_p = n_h \times n_w$ is the number of patches (n_h along the height and n_w along the width), each patch is of shape s_h, s_w ($s_h = s_w = 16$ for ViT). If the patches are non-overlapping and uniform, then $H = s_h \times n_h$ and $W = s_w \times n_w$. $\mathbf{G}$ can be reshaped into $\mathbf{D}_p \in \mathbb{R}^{N_p,d_p}$ where $d_p = C \times s_h \times s_w$.

Bag of Visual Words (BoVW) [192] approach represents an image as a histogram of visual words. Given a database of images denoted by $\mathbf{I_{DB}}$, it first extracts local image features using methods like those mentioned previously. These features are then encoded into corresponding descriptors, denoted by $\mathbf{V_{DB}} \in \mathbb{R}^{N_{total},d}$, where N_{total} represents the total number of descriptors extracted from all N_d database images. These N_{total} descriptors are then clustered into k clusters using a technique like k-means clustering. The centers of these clusters form the *vocabulary*. For each image, the BoVW approach counts the number of features belonging to each cluster, essentially building a histogram. This histogram representation is typically limited to the top-n most frequent vocabulary terms. This image processing pipeline provides a histogram representation for each image in the database. When a new image is considered, its histogram is compared against the histograms of stored database images using metrics like chi-squared distance, histogram intersection, cosine distance, or other nearest neighbor approaches. This is the vision equivalent of Bag-of-Words (BOW) from NLP for language and document retrieval.

Building upon the standard BoW, the *spatial BoW* approach incorporates spatial information. It divides the image into smaller regions and independently applies BoW to each region. This creates a local vocabulary for each region, capturing the presence of visual words within specific image areas. The final image representation combines information from all regional BoW histograms, potentially including the location of each region within the image.

To improve efficiency for real-time applications, some approaches convert descriptors to *binary* strings using a thresholding technique on the histogram values. Alternatively, certain local feature extraction methods inherently produce binary descriptors. This binary representation enables faster comparisons during the matching stage.

Hierarchically structured vocabularies can also be constructed using clustering methods. [187] propose a method where each cluster in a hierarchical structure is further partitioned, and a clustering technique like k-means is applied recursively at each level. This approach facilitates a coarse-to-fine vocabulary creation suitable for long-term operations. DBoW2 [164] uses this technique with binary descriptors while also leveraging it for loop detection.

VLAD [159, 175] Vector of Locally Aggregated (VLAD) descriptor is constructed by following these steps:

1. *Feature Extraction*: Local features descriptors such as SIFT (or RootSIFT [163]) are extracted from the image.

2. *Cluster Assignment*: Each descriptor is assigned to its closest cluster in a pre-defined vocabulary of size k.

3. *Residual Accumulation*: For each cluster, the residuals are calculated. These residuals are vector differences between the assigned descriptors and the cluster centers. The residuals are then accumulated (added) to obtain a 128-d vector (same dimensionality as descriptor used) per cluster.

4. *VLAD Descriptor Formulation*: Finally, the $128 - d$ residuals across all k clusters are concatenated to form a single $k \times 128$ dimensional vector which is the VLAD vector.

We normalize the accumulated cluster residuals before concatenation (known as intra-norm) and normalize the final concatenated result to prevent bias towards a small subset of clusters and to effectively utilize the entire vocabulary. The vocabulary is constructed by extracting descriptors across all the database images and clustering them using a method like k-means. Since the cluster assignment of VLAD is non-differentiable, it cannot directly be used for training.

Mathematically, given an image $\mathbf{I} \in \mathbb{R}^{C,H,W}$, we extract its keypoints $\mathbf{K} \in \mathbb{R}^{N_k,2}$ and descriptors $\mathbf{D} \in \mathbb{R}^{N_k,d_k}$. We have $\mathbf{C} \in \mathbb{R}^{K,d_k}$ as the vocabulary (K cluster centers, each d_k dimensional). We compute a residual matrix $\mathbf{V} \in \mathbb{R}^{K,d_k}$ as follows

$$\mathbf{V}[k,j] = \sum_{i=1}^{N_k} \left[\alpha\left(\mathbf{D}[i],k\right)\left(\mathbf{D}[i,j] - \mathbf{C}[k,j]\right)\right] \tag{1.4}$$

Where $\mathbf{V}[k,j] \in \mathbb{R}$ is the residual for the k-th cluster center's j-th dimension, $\alpha : (\mathbb{R}^{d_k}, \mathbb{N}) \to \{0,1\}$ takes in the descriptor and cluster index; and returns 1 if the descriptor belongs to the cluster (the given cluster is the closest to the descriptor) and 0 otherwise. The matrix $\mathbf{V}$ is normalized along d_k (each of K rows, which is a d_k-dimensional vector, is normalized) for intra-normalization. It is then reshaped (flattened) to a vector $\mathbf{v} \in \mathbb{R}^{K \times d_k}$ and then normalized (same shape). This final vector $\hat{\mathbf{v}} = \|\mathbf{v}\|_2$ is the VLAD vector.

The cluster centers $\mathbf{C} \in \mathbb{R}^{K,d_k}$ are obtained by applying a clustering technique like k-means clustering to the descriptors from database images. We start with a set of database images $\mathbf{I_{DB}} = \{\mathbf{I}_1, \mathbf{I}_2, \cdots, \mathbf{I}_{N_d}\}$, where $\mathbf{I}_i \in \mathbb{R}^{C,H,W}$ is a database image, and extract its corresponding local feature descriptors $\mathbf{D_{kdb}} = \{\mathbf{D}_1, \mathbf{D}_2, \cdots, \mathbf{D}_{N_d}\}$ where $\mathbf{D}_i \in \mathbb{R}^{N_i,d_k}$ is the list of descriptors for image $\mathbf{I}_i$ (each image might have a different number of descriptors). These descriptors are stacked along N_i dimension to give $\mathbf{D}_{all} \in \mathbb{R}^{N_{total},d_k}$ (where $N_{total} = \sum_i N_i$). These N_{total} descriptors are clustered into K clusters giving the cluster centers $\mathbf{C} \in \mathbb{R}^{K,d_k}$.

NetVLAD [144] is a differentiable version of VLAD. VLAD has two non-differentiable steps: *clustering* for fitting on database images (descriptors) and cluster *assignment* during construction of VLAD, denoted as $\alpha(\bullet,\bullet)$ above. NetVLAD uses the following cluster assignment instead

$$\tilde{\alpha}\left(\mathbf{d}_i, \mathbf{c}_k\right) = \frac{e^{-\lambda\|\mathbf{d}_i - \mathbf{c}_k\|_2^2}}{\sum_{k'} e^{-\lambda\|\mathbf{d}_i - \mathbf{c}_{k'}\|_2^2}} \tag{1.5}$$

Where $\mathbf{d}_i = \mathbf{D}[i] \in \mathbb{R}^{d_k}$ is the i-th descriptor in the image and $\mathbf{c}_k = \mathbf{C}[k] \in \mathbb{R}^{d_k}$ is the k-th cluster center. The above equation can be simplified further

$$
\begin{aligned}
\|\mathbf{d}_i - \mathbf{c}_k\|_2^2 &= \sum_j (\mathbf{d}_i[j] - \mathbf{c}_k[j])^2 = \sum_j \left[\mathbf{d}_i[j]^2 + \mathbf{c}_k[j]^2 - 2\mathbf{d}_i[j]\mathbf{c}_k[j]\right] \\
&= \left\|\mathbf{d}_i^2\right\|_1 + \left\|\mathbf{c}_k^2\right\|_1 - 2\mathbf{d}_i \cdot \mathbf{c}_k \\
e^{-\lambda\|\mathbf{d}_i - \mathbf{c}_k\|_2^2} &= e^{2\lambda \mathbf{d}_i \cdot \mathbf{c}_k} \cdot e^{-\lambda\left\|\mathbf{d}_i^2\right\|_1} \cdot e^{-\lambda\left\|\mathbf{c}_k^2\right\|_1} \\
e^{-\lambda\|\mathbf{d}_i - \mathbf{c}_{k'}\|_2^2} &= e^{2\lambda \mathbf{d}_i \cdot \mathbf{c}_{k'}} \cdot e^{-\lambda\left\|\mathbf{d}_i^2\right\|_1} \cdot e^{-\lambda\left\|\mathbf{c}_{k'}^2\right\|_1} \\
\tilde{\alpha}\left(\mathbf{d}_i, \mathbf{c}_k\right) &= \frac{e^{-\lambda\|\mathbf{d}_i - \mathbf{c}_k\|_2^2}}{\sum_{k'} e^{-\lambda\|\mathbf{d}_i - \mathbf{c}_{k'}\|_2^2}} = \frac{e^{2\lambda \mathbf{d}_i \cdot \mathbf{c}_k} \cdot e^{-\lambda\left\|\mathbf{d}_i^2\right\|_1} \cdot e^{-\lambda\left\|\mathbf{c}_k^2\right\|_1}}{\sum_{k'} e^{2\lambda \mathbf{d}_i \cdot \mathbf{c}_{k'}} \cdot e^{-\lambda\left\|\mathbf{d}_i^2\right\|_1} \cdot e^{-\lambda\left\|\mathbf{c}_{k'}^2\right\|_1}} \\
&= \frac{e^{2\lambda \mathbf{d}_i \cdot \mathbf{c}_k - \lambda\left\|\mathbf{c}_k^2\right\|_1}}{\sum_{k'} e^{2\lambda \mathbf{d}_i \cdot \mathbf{c}_{k'} - \lambda\left\|\mathbf{c}_{k'}^2\right\|_1}} = \frac{e^{\mathbf{w}_k^T \mathbf{d}_i + b_k}}{\sum_{k'} e^{\mathbf{w}_{k'}^T \mathbf{d}_i + b_{k'}}}
\end{aligned}
$$

The final VLAD residual matrix becomes

$$
\mathbf{V}[k, j] = \sum_{i=1}^{N_k} \left[\frac{e^{\mathbf{w}_k^T \cdot \mathbf{d}_i + b_k}}{\sum_{k'} e^{\mathbf{w}_{k'}^T \cdot \mathbf{d}_i + b_{k'}}} \left(\mathbf{D}[i, j] - \mathbf{C}[k, j]\right) \right] \tag{1.6}
$$

Where $\{\mathbf{w}_k \in \mathbb{R}^{d_k}\}$, $\{b_k \in \mathbb{R}\}$, and $\mathbf{C} \in \mathbb{R}^{K, d_k}$ are trainable parameters. Conventionally, the descriptors $\mathbf{D} \in \mathbb{R}^{N_k, d_k}$ are obtained as dense features from a CNN backbone.

GeM Pooling [125] uses generalized mean pooling over image descriptors $\mathbf{D} \in \mathbb{R}^{N_k, d_k}$ and yields a vector $\mathbf{f}_G \in \mathbb{R}^{d_k}$ where each element is given by

$$
\mathbf{f}_G[j] = \left(\frac{1}{N_k} \sum_{i=1}^{N_k} \left(\mathbf{D}[i, j]\right)^{p_k} \right)^{\frac{1}{p_k}} \tag{1.7}
$$

An advantage of GeM pooling is that the dimension of the output descriptor is d_k, whereas it is $K \times d_k$ for VLAD. This facilitates storing more of them on limited memory.

Other Pooling Methods include max-pooling ,also called *MAC* (Maximum Activations of Convolutions) vector [157, 152] given by $\mathbf{f}_M \in \mathbb{R}^{d_k}$ and average-pooling, also called SPoC (Sum-Pooled Convolution) features [145] given by $\mathbf{f}_A \in \mathbb{R}^{d_k}$. These are defined below

$$
\mathbf{f}_M[j] = \max_i \mathbf{D}[i, j] \qquad\qquad \mathbf{f}_A[j] = \frac{1}{N_k} \sum_{i=1}^{N_k} \mathbf{D}[i, j] \tag{1.8}
$$

1.2.3.3 Other Approaches to VPR

While traditional approaches to VPR often rely on features extracted from trained CNN backbones [113], this thesis focuses on leveraging foundation models for this task.

Other areas of exploration in VPR include the use of sequential descriptors [54, 55, 116], low-resolution image training [37], and visual semantics [109]. Techniques like mesh-based localization [43],

direct 2D-3D matching [172], change-based VPR [84], hierarchical 3D descriptors [83], novel pooling methods (like SuperGlobal [22]), and cross-device queries [17] are also being investigated, along with the development of sequential SLAM pipelines [165].

Chapter 2

Foundation Models

Foundation models (FM) are a type of artificial intelligence (AI) model designed for learning generic representations from data. They typically employ unsupervised learning techniques and fall under the umbrella of representation learning. This chapter includes an overview of Vision Foundation Models (VFMs). Foundation Models, like all AI models, have the following components: model architecture, dataset, objective (training strategy), and optimizer.

- *Model architecture*: This is used to store the parameters and decides how operations are preformed on the input data. They are usually multi-layer perceptrons (MLPs), convolution [195], and/or transformer [82, 131] layers. Recent developments include MLP mixer [71], ConvNext [41], and some transformer variants (Swin Transformer [60, 61], CCT [57], etc).

- *Dataset*: This is the source of learning for the FM. Usually, these are large datasets that contain samples from a wide distribution. Some common publicly available datasets in vision are ImageNet [178] and Tencent ML-Images [107]. Some proprietary datasets include JFT-300M [126], JFT-3B [73] (Google), and IG-3.5B [111] (Facebook/Meta).

- *Objective, training strategy and loss function*: Since the datasets are non-labelled and manual annotation is difficult, the training strategy must include some form of self-supervision to facilitate representation learning. These include techniques like knowledge distillation [147], masked representation learning [58], and various contrastive learning techniques and losses that aim to align modalities. Some implementations include MoCo [97, 80], SwAV [77], SimCLR [78, 79], BYOL [85], etc.

- *Optimizer*: These are used to tune the model's parameters. Usually, optimizers like the Adam [156] and its recent development, AdamW [120], are well suited. However, since the number of parameters often run into hundreds of mullions (and often billions), it makes heavy use of parallelization techniques like data sharding and model splitting across GPUs. Common implementations of parallel optimizers include Fully Sharded Data Parallel (FSDP) [93] (part of FairScale and PyTorch), Zero Redundancy Optimizer (ZeRO) [23, 102], DeepSpeed [39], Megatron [65], etc.

This chapter gives an overview of model architectures and training strategies involved in DINO [51] and DINOv2 [21], which are the vision backbones used for this work. A thorough review on self-supervised learning (SSL) and foundation models can be found in [6, 12, 86].

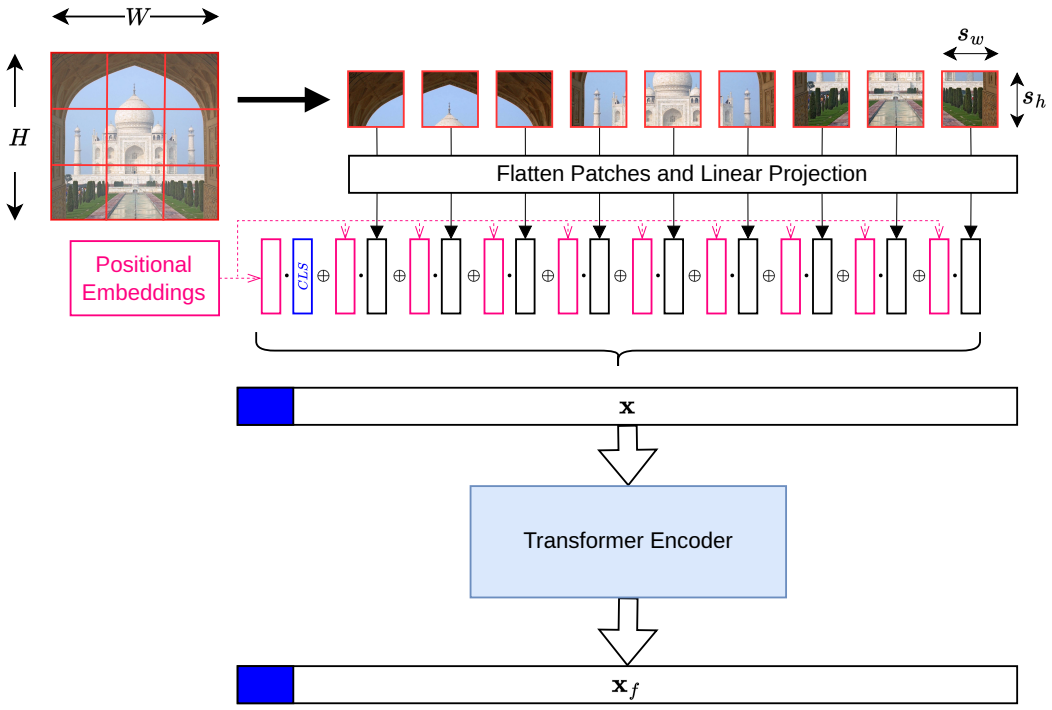


Figure 2.1 Data processing for Vision Transformers
Input image is first broken into patches. These patches are flattened and a (shared) linear projection is applied to all of them. A $[CLS]$ token is added for global context. Then, position embeddings are added to each element and the entire set in concatenated into a vector (with the $[CLS]$ - denoted in blue squared - usually going first). The $\cdot$ (dot) signifies addition of position embeddings and $\oplus$ signifies concatenation.
Image is best viewed in color.

2.1 Vision Transformers

2.1.1 ViT

Vision Transformers (ViTs) [82] are inspired from transformers in Natural Language Processing (NLP) [131], which in turn are inspired by *information retrieval* using queries in a key-value database. They operate in a manner described below (and shown in Figs. 2.1 and 2.2)

Step 1 - Patchification of the Input Image As shown in Fig. 2.1, an input image $\mathbf{I} \in \mathbb{R}^{C,H,W}$ is split into patches of shape s_h, s_w each, giving $n_h = H/s_h$ patches along height and $n_w = W/s_w$ patches along width. These $n = n_h \times n_w$ patches are then flattened into $d_m = C \times s_h \times s_w$ dimensional vectors. Each patch is multiplied by a *shared* square linear matrix, keeping the output dimensionality same. We obtain the transformed patch embeddings as $\left\{\mathbf{p}_i \in \mathbb{R}^{d_m}\right\}_{i=[1,n]}$. For utilizing global context, a learnable $\mathbf{p}_0 = [CLS]$ token (same shape as the patch vectors) is added to the set. These $n+1$ patches are then transformed by adding position embeddings (so that each element in the sequence gets a notion of its

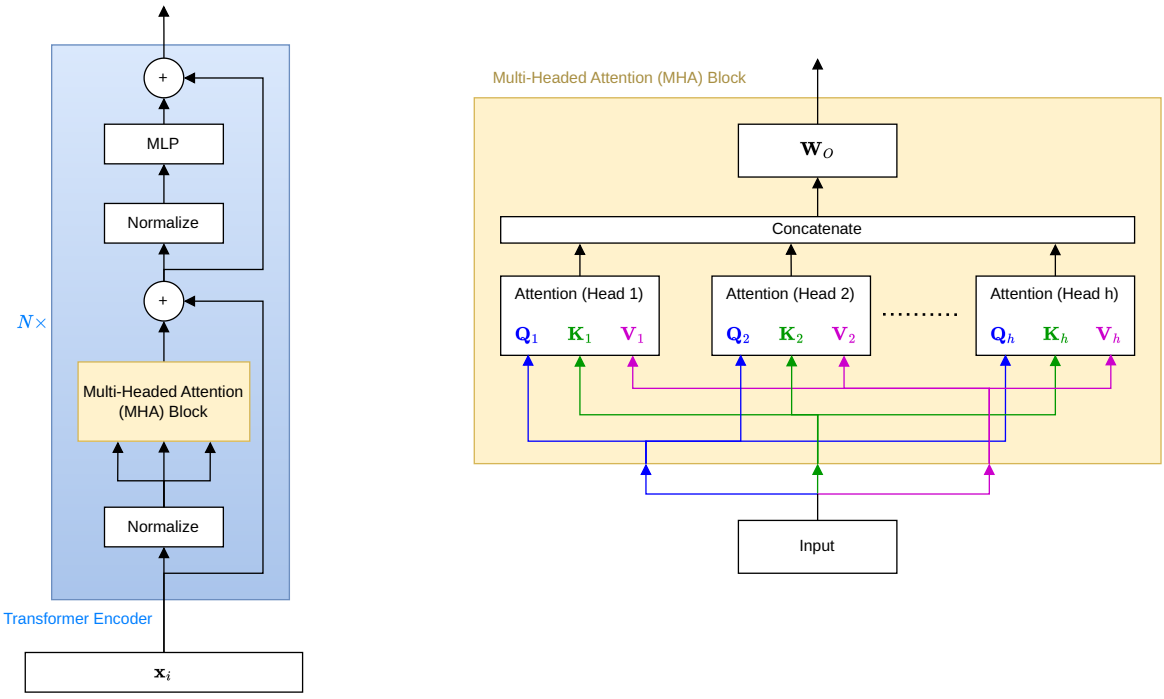


Figure 2.2 Transformer Encoder and Multi-Headed Attention
Each transformer encoder block contains multi-headed attention block. The + operations denote addition of residuals. Each head has its own query $\mathbf{Q}_i$, key $\mathbf{K}_i$ and value $\mathbf{V}_i$ vectors and applies self-attention. Image is best viewed in color.

position in the sequence). Finally, they're all concatenated into a vector. The final input to the model is given by the vector $\mathbf{x} \in \mathbb{R}^{(1+n)\times d_m}$ (with the first d_m values for the $[CLS]$ token). This vector $\mathbf{x}$ is the input to the transformer layers. For simplicity, we use $l = 1 + n$ for sequence length hereon.

Step 2 - Transformer Encoders with Multi-headed Attention As shown in figure Fig. 2.2, the input is given to N transformer encoder blocks, which are linked in sequence. Each block normalizes its input, passes it through a multi-headed attention (MHA) layer in residual, it finally passes it through an MLP block in the same normalization and residual manner. LayerNorm [133] is used instead of BatchNorm [148] because it is easier to parallelize and can deal with varying batch sizes. Each head (say head i) of MSA does the following

1. The input $\mathbf{x} \in \mathbb{R}^{l,d_m}$ (where $l = 1 + n$ is the sequence length and d_m is the feature dimension) is passed through LayerNorm, giving $\tilde{\mathbf{x}} = \text{LN}(\mathbf{x})$ (where $\tilde{\mathbf{x}} \in \mathbb{R}^{l,d_m}$). We then apply weights $\mathbf{W}_{Qi} \in \mathbb{R}^{d_m,d_k}$ for query, $\mathbf{W}_{Ki} \in \mathbb{R}^{d_m,d_k}$ for key, and $\mathbf{W}_{Vi} \in \mathbb{R}^{d_m,d_k}$ for value. Here, $d_k = d_m/h$, where h is the number of heads (h is chosen such that it divides d_m). We get the query $\mathbf{Q}_i \in \mathbb{R}^{l,d_k}$, key $\mathbf{K}_i \in \mathbb{R}^{l,d_k}$, and value $\mathbf{V}_i \in \mathbb{R}^{l,d_k}$ vectors using

$$\mathbf{Q}_i = \tilde{\mathbf{x}}\mathbf{W}_{Qi} \qquad \mathbf{K}_i = \tilde{\mathbf{x}}\mathbf{W}_{Ki} \qquad \mathbf{V}_i = \tilde{\mathbf{x}}\mathbf{W}_{Vi} \tag{2.1}$$

2. The output of the self-attention $\mathbf{A}_i \in \mathbb{R}^{l,d_k}$ is given by

$$\mathbf{A}_i = \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \text{softmax}\left(\frac{\mathbf{Q}_i \mathbf{K}_i^{\text{T}}}{\sqrt{d_k}}\right)\mathbf{V}_i \tag{2.2}$$

The above equation is similar to information retrieval where a query is matched with key using d_k dimensional embeddings to retrieve value (information). This formulates it in a differentiable and continuous manner. The softmax uses $\sqrt{d_k}$ for normalization and does row-wise normalization.

3. The output across all heads is concatenated (along the d_k dimension of each vector) into a single vector $\mathbf{A} \in \mathbb{R}^{l,d_m}$ using

$$\mathbf{A} = [\mathbf{A}_1 \oplus \mathbf{A}_2 \oplus \cdots \oplus \mathbf{A}_i \oplus \cdots \mathbf{A}_h] \tag{2.3}$$

4. The final output $\mathbf{y} \in \mathbb{R}^{l,d_m}$ of the MHA block is given by applying a linear transformation through weights $\mathbf{W}_O \in \mathbb{R}^{d_m,d_m}$, given by

$$\mathbf{y} = \mathbf{A}\mathbf{W}_O \tag{2.4}$$

The weights $\mathbf{W}_O$ mix the outputs across all heads.

Step 3 - Normalize and Linear Transform The output of MHA block $\mathbf{y} \in \mathbb{R}^{l,d_m}$ is then added with the input $\mathbf{x} \in \mathbb{R}^{l,d_m}$ (residual connection). The result is then passed through LayerNorm [133] (along the d_m dimension) and then a 2-layer MLP where first layer expands the dimension from d_m to $4d_m$, and the second layer reduces the dimension form $4d_m$ to d_m. The result is added with a residual connection, shown in Fig. 2.2. This gives the output $\mathbf{z} \in \mathbb{R}^{l,d_m}$ of the current layer (which is then the input for the next layer).

$$\mathbf{z}_{out} = \text{MLP}(\text{LN}(\mathbf{y} + \mathbf{x})) + (\mathbf{y} + \mathbf{x}) \tag{2.5}$$

Downstream Tasks The downstream tasks take the $[CLS]$ token (first d_m values) and use it for classification (by adding an additional MLP) [82]. We explore the representations learned by latent layers for visual place recognition.

Configurations The ViT proposed by Google has variants described in Table 2.1. The input image size is $(224, 224)$. With patch size $(16, 16)$, it gives $d_m = 3 \times 16 \times 16 = 768$ dimensional patch embeddings. Architectures that have a different embedding dimension (like ViT-L) use a non-square linear projection at the input stage (as shown in Fig. 2.1); these are usually used to increase dimensionality. All entries are from google-research/vision_transformer on GitHub.

2.1.2 DeIT

Data efficient Image Transformers (DeIT) [91] proposes minor modifications towards training transformers. Instead of training on JFT-300M (like ViT), DeIT is trained only on ImageNet. It uses the same architecture and improvements from the `timm` (PyTorch Image Models) library [106]. Compared to ViT, DeIT also uses a lower batch size (1024 vs. 4096), learning rate adjusted with batch size [119],

Family	Model	Params	Patch s	Emb. d_m	Heads h	Layers N	MLP Block
ViT [82]	ViT-B	86M	16	768	12	12	(*-3072-*)
	ViT-L	307M	16	1024	16	24	(*-4096-*)
	ViT-H	632M	14	1280	16	32	(*-5120-*)
DeIT [91]	DeIT-Ti	5M	16	192	3	12	(*-768-*)
	DeIT-S	22M	16	384	6	12	(*-1536-*)
	DeIT-B	86M	16	768	12	12	(*-3072-*)
CCT [57]	ViT-Lite-7/8	3.74M	8	256	4	7	(*-512-*)
	ViT-Lite-7/4	3.72M	4	256	4	7	(*-512-*)
	CVT-7/4	3.72M	4	256	4	7	(*-256-*)
	CCT-2/3x2	0.28M	c(3x3)*2	128	2	2	(*-128-*)
	CCT-7/3x1	3.76M	c(3x3)	256	4	7	(*-512-*)

Table 2.1 Transformer Variants and Specifications
Comparisons of different transformer configurations. Params signifies the number of parameters. The patch size is given by $s = s_h = s_w$ (square patches), $c(\cdot)$ means convolution tokenizer. MLP block specifications are given as $(input - hidden - output)$, a '*' here means 'same as d_m'.

and RandAugment data augmentation [81]. Various DeIT models are mentioned in Table 2.1 (taken from [91] and facebookresearch/deit GitHub).

Distillation Models DeIT also proposes a distillation strategy for vision transformers where an extra distillation token is added (like the class token), but is trained for a distillation loss (like the KL divergence). Instead of using a *soft distillation* approach (modeling the softmax of teacher), they use a hard-label distillation strategy where they use cross-entropy (CE) loss on the teacher's label. More on knowledge distillation is mentioned in Section 2.2.1.

2.1.3 Other Model Architectures

Swin Transformers [61] Shifted Windows (Swin) transformers constrains the self-attention to local windows. This way, in a particular layer, only the patches within a local window exchange information. This avoids the quadratic complexity of applying global attention. Consecutive layers have shifted windows so that information exchange happens through patches that enter or leave windows. It also proposes merging patches to form hierarchical representations.

Swin Transformer V2 [60] improves the Swin Transformer further by changing the pre-normalization to post-normalization, replacing the dot-product to a scaled cosine function in the attention formulation, and using log-space in positional embeddings.

CCT [57] aims to train vision transformers on small datasets. It proposes ViT-Lite, Compact Vision Transformers (CVT), and Compact Convolution Transformers (CCT). Its variants are presented in Table 2.1.

ViT-Lite is a more suitable ViT architecture, nearly identical to the original ViT, for small-scale learning. It achieves better results using only 7 layers and a $4, 4$ patch size, while having fewer parameters.

CVT removes the conventional $[CLS]$ token and instead uses Sequence Pooling (SeqPool) to get the output. It uses a linear layer with $\mathrm{softmax}$ to generate weights for pooling the output tokens.

CCT uses a convolutional tokenizer in CVT. It replaces the input embeddings (like in Fig. 2.1) with a conv layer followed by $\mathrm{MaxPool}$. This adds inductive biases of a convolution network in the transformer model.

2.2 SSL Concepts

2.2.1 Knowledge Distillation

In the field of deep learning, knowledge distillation refers to the technique of transferring knowledge acquired by a pre-trained model (teacher) to a new model (student) [147]. This process aims to empower the student model to achieve performance comparable to the teacher, often with a smaller and more efficient architecture. During knowledge distillation, the student learns to mimic the teacher's behavior, effectively leveraging the teacher's knowledge to improve its own performance. Usually, a dataset split, called the "transfer set", is used for this purpose. It is also common to train the student model on the dataset after distillation or concurrently. Most commonly, it is of three types

1. Teacher-Student with different architectures: This scenario is commonly used when the teacher network is a large network trained on a significant amount of data, while the student network is a simpler model designed for deployment constraints. The student network is trained to mimic the softmax outputs (often just before the output layer) of the teacher on the transfer set.

2. Teacher-Student with same architecture: This strategy aims for the student to learn robust representations from the training dataset. Here, the teacher, often initialized with the same architecture as the student, is trained using an exponentially moving average (EMA) of the student gradients. Such techniques enable a fast-learning student to benefit from a teacher that has access to a more robust representation of the training data (through the EMA).

3. Mixture-of-Experts teacher: This approach is typically used for very large datasets with many categories. A global generalist model is trained to identify categories within the data, and an expert model is trained on each category. The generalist model serves two key purposes: identifying sub-classes (gating mechanism) for a given sample and acting as a catch-all class for samples where no expert has a strong specialization. Samples from a transfer set are used to train a student model that must model the probability distribution of both the generalist and the relevant expert(s) for the category of the sample.

2.2.1.1 Loss for Learning Targets

In knowledge distillation, the loss function plays a crucial role in guiding the student model to mimic the knowledge of the teacher model. There are two primary approaches for defining the target distribution that the student should learn: soft targets and hard targets. Soft target modeling is when the concerned probability distribution is a softmax distribution over all output classes of the teacher. On the other hand,

hard target is when only the single output class (per sample of the transfer set) is considered. Usually, losses like cross entropy and Kullback-Leibler (KL) Divergence are used for this alignment. These are derived from information theory, but fund use in aligning probability distributions (like softmax outputs of a student to align with the output of teacher).

Entropy is a measure of useful information for a probability of an event. For example, let x be a random variable modeling distribution $P(X)$, that is $x \sim P(X)$. The entropy (information) of $x = E$, basically $P(X = E) = P(E)$ is given by

$$I_{x \sim P}(E) = \log\left(\frac{1}{P(E)}\right) = -\log\left(P(E)\right) \tag{2.6}$$

The expected entropy over all states/events of x is the entropy of the distribution P. It is given by

$$\mathrm{H}_{x \sim P}(X) = \mathrm{E}_{x \sim P}\left(I(X)\right) = \sum_{x \in X} P(X = x) I(X = x) = -\sum_{x \in X} p(x) \log(p(x)) \tag{2.7}$$

where $p(x) = P(X = x) \in \mathbb{R}$ is used to denote the probability of the event $X = x$.

Kullback-Leibler Divergence or KL Divergence is the excess information required to model X using distribution $Q(X)$, when the actual (underlying) distribution is $P(X)$. It is used to measure the deviation when using $x \sim Q$, if the actual distribution is supposed to be $x \sim P$; it is a directed difference between two distributions. It is modeled as

$$\begin{aligned}\mathrm{D}_{KL}(P \parallel Q) &= \mathrm{E}_{x \sim P}\left(I_{x \sim Q}(X = x) - I_{x \sim P}(X = x)\right) = \sum_{x \in X} P(X = x) \log\left(\frac{P(X = x)}{Q(X = x)}\right) \\ &= \sum_{x \in X} p(x) \log\left(\frac{p(x)}{q(x)}\right) = -\sum_{x \in X} p(x) \log\left(\frac{q(x)}{p(x)}\right)\end{aligned} \tag{2.8}$$

where $q(x) = Q(X = x) \in \mathbb{R}$ is used to denote the probability of the event $X = x$ (using distribution $Q(X)$). The above is easy to differentiate with respect to $q(x)$ and it is ideal to have $q(x)$ for the student and $p(x)$ for the teacher.

Cross Entropy is defined as the difference between two distributions for a specific event. It is the excess information needed for encoding Q, when the underlying distribution is P. It is formulated as

$$\begin{aligned}\mathrm{H}(p, q) &= \mathrm{H}(p) + \mathrm{D}_{KL}(p \parallel q) = \sum_{x \in X} p(x) \log\left(\frac{1}{p(x)}\right) + \sum_{x \in X} p(x) \log\left(\frac{p(x)}{q(x)}\right) \\ &= \sum_{x \in X} p(x) \log\left(\frac{1}{q(x)}\right) = -\sum_{x \in X} p(x) \log\left(q(x)\right)\end{aligned} \tag{2.9}$$

This is more directly related to the entropy of Q and the likelihood of $P(X = x)$ and is more commonly used to align a student (as $q(x)$) to a teacher (as $p(x)$).

2.3 DINO and DINOv2 Models

2.3.1 DINO

DINO [51], distillation with no labels, is a method of knowledge distillation where the student and the teacher have the same architecture and where the *cross entropy loss* is used for aligning the student to the teacher and where the teacher is the EMA of the student network. Using the multi-crop strategy [77], a set of views V is generated from an image. Two of these views are large global crops x_1^g, x_2^g and others are smaller resolution local views. The student network is given by P_s and the teacher is given by P_t (both have the same architecture). The training loss is minimized as follows

$$\min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{\substack{x' \in V \\ x' \neq x}} H(P_t(x), P_s(x')) \tag{2.10}$$

Note that the teacher gets no gradient. In PyTorch, it's implemented using a `detach` call to the output. The teacher is updated using an exponentially moving average (momentum encoder) [97] of the student weights, given by $\theta_t \leftarrow \lambda\theta_t + (1 - \lambda)\theta_s$. To avoid collapse of outputs (preventing one dimension to dominate over the others or the output becoming a uniform distribution), the teacher's output is centered (with the center in-turn being updated as an EMA of teacher's outputs) and sharpened using temperature softmax.

The architecture is DeIT from Section 2.1.2, with patch size 8 and the transformer having a pre-norm layer normalization. The main work of DINO uses a $[CLS]$ token and applies a linear projection head or employs k-NN (k nearest neighbor) for classification. However, we are more interested in features learned by the latent (intermediate) transformer layers. Especially for the context of VPR.

DINO demonstrates that self-supervised learning using the ViT architecture could form a strong vision foundation model that can be used for downstream tasks.

2.3.2 Further Developments

2.3.2.1 iBOT

Image BERT pre-training with Online Tokenizer (iBOT) [74] takes inspiration from Masked Language Modeling (MLM) in BERT [95] to perform Masked Image Modeling (MIM). It formulates MIM task as a knowledge distillation (KD) task with an online tokenizer (used to transform the patches) trained simultaneously with the target model. The MIM task is also solved in BEiT (BERT Pre-Training of Image Transformers) [50], albeit with an offline tokenizer learned from visual vocabulary.

During training, block-wise masking is performed on the input image and two augmented views $\mathbf{u}$ and $\mathbf{v}$ are generated with their corresponding masked views $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$. The student and the teacher have a transformer backbone and a shared projection head (which is an MLP) for patch tokens and $[CLS]$. The losses contain distillation across the same view and cross-view, for the MIM objective and the $[CLS]$ objective. For the MIM task, the objective is to align the predicted softmax of the masked tokens (where $m_i = 1$). The cross-entropy (CE) objectives are given as

$$\mathrm{L}_{MIM}^{u_t \to \hat{u}_s} = -\sum_{i=1}^{N} m_i \mathrm{P}_{\theta'}^{t} \left(\mathbf{u}_t^{patch}\right)^{\top} \log\left(\mathrm{P}_{\theta}^{s}\left(\hat{\mathbf{u}}_s^{patch}\right)\right) \quad \mathrm{L}_{[CLS]}^{u_t \to \hat{v}_s} = -\mathrm{P}_{\theta'}^{t}\left(\mathbf{u}_t^{[CLS]}\right) \log\left(\mathrm{P}_{\theta}^{s}\left(\hat{\mathbf{v}}_s^{[CLS]}\right)\right)$$
$$\mathrm{L}_{MIM}^{v_t \to \hat{v}_s} = -\sum_{i=1}^{N} m_i \mathrm{P}_{\theta'}^{t} \left(\mathbf{v}_t^{patch}\right)^{\top} \log\left(\mathrm{P}_{\theta}^{s}\left(\hat{\mathbf{v}}_s^{patch}\right)\right) \quad \mathrm{L}_{[CLS]}^{v_t \to \hat{u}_s} = -\mathrm{P}_{\theta'}^{t}\left(\mathbf{v}_t^{[CLS]}\right) \log\left(\mathrm{P}_{\theta}^{s}\left(\hat{\mathbf{u}}_s^{[CLS]}\right)\right) \tag{2.11}$$

Where $\mathrm{L}^{a \to b}$ denotes distillation from teacher a to student b, and L_{MIM} and $\mathrm{L}_{[CLS]}$ are MIM tokens and CLS token losses respectively. As in DINO, the teacher is an EMA of student, updated through momentum of student weights. iBOT is able to form robust representations suitable for many downstream tasks such as object detection, panoptic segmentation, and transfer learning. The patch tokens learn semantically meaningful information that contains global context (because of the shared head with $[CLS]$ and the online tokenizer).

2.3.2.2 DINOv2

DINOv2 [21] contains an automatic pipeline to build a rich and diverse training dataset, and a distillation stage from a large model. It performs significantly better than CLIP [66] on benchmarks needing local and global semantic information.

Following [44], a curated dataset is first built by starting with little curated data. Embeddings are extracted from curated and un-curated data. Duplicates in the un-curated data (having close embeddings) are removed. Those samples that can be retrieved from the remaining un-curated set by querying embeddings from the curated set are added to the augmented curated set. The dataset source includes ImageNet, Google Landmarks, and many fine-grained datasets like ADE-20K, Pascal VOC, CityScapes, etc. The final dataset is called LVD-142M.

DINOv2 uses the same image-level objective from DINO (in Section 2.3.1) and patch-level objective from iBOT (in Section 2.3.2.1): cross entropy losses, knowledge distillation setting, teaching being the EMA of student. Other than this, DINOv2 has several improvements

1. Unlike the shared projection heads in iBOT, it *unties* (separates) the heads for patch and image-level objective.

2. Instead of applying softmax and centering to the teacher's output, *Sinkhorn-Knopp centering* is applied for 3 iterations. During training, the activations for each mini-batch are L2-normalized and a pair-wise similarity matrix is computed. This similarity matrix is steered to a template. The student goes through only softmax-norm. This is inspired from SwAV [77].

3. Uses *KoLeo Regularizer* [112]: Given a set of vectors (activations for a minibatch) $\{x_1, x_2, \cdots, x_n\}$, it defines a loss $L_{koleo} = -\frac{1}{n}\sum_{i=1}^{n} \log(d_{n,i})$ where $d_{n,i} = \min_{j \neq i} \|x_i - x_j\|$ is the minimum distance between samples in the minibatch. This aims to spread out samples in each batch.

4. Using Swish Gated Linear Unit (SwiGLU) activations [89] for the FFN in the transformer layers.

5. Using *Stochastic Depth* [136] to shrink the depth of a network during training (by randomly dropping entire residual blocks and replacing it with a bypass) while keeping it the same during inference. It helps the network learn sophisticated representations in earlier layers.

6. Using *LayerScale* [72] to train deep transformer models (with many layers). It adds a learnable diagonal matrix (multiplication) to the output of the residual blocks. This has been shown to increase model capacity.

7. Training a large ViT-g network (with 1.1B parameters) from scratch and then training smaller models through knowledge distillation [147]. It trains a ViT-g/14 from scratch on LVD-142M and uses this as the frozen teacher for the initial stages of training smaller ViT models (like ViT-L/14, ViT-B/14, and ViT-S/14).

8. Using higher-resolution images (518, 518) during the end of pre-training. Additionally, using a patch size of 14 seems to give better results.

Additionally, efficient implementation with `xFormers` [38], `timm` [106], and `FairScale` [52] libraries bring the following changes

1. Improved computational efficiency and resource utilization using FlashAttention [35]: it breaks the attention matrix operation into smaller chunks (called tiles) and optimizes the memory R/W operations on the GPU (between HBM and SRAM).

2. Nested Tensors allow modeling tensors with varying sequence length more efficiently (by containerizing each element as a Tensor).

3. Using PyTorch FSDP (Fully Sharded Data Parallel) [29] to split the model and data across GPUs and to reduce cross-GPU communication. This relates to data sharding (splitting across databases) in large database management systems, but it splits the model's parameters into shards (splits), usually along feature dimensions.

The above improvements lead to better representations learned by DINOv2 and make its training more efficient. It performs much better than DINO [51], MAE [58], and iBOT [74] on ImageNet, and has better linear evaluation over frozen features (linear probing) on fine-grained benchmarks. It also performs better on tasks like semantic segmentation, depth estimation, and image retrieval.

We are interested in exploiting latent features learned by intermediate transformer layers of DINOv2 for the purpose of visual place recognition (VPR). We show that these learn more meaningful representations and can be clubbed with existing feature aggregation techniques.

Chapter 3

AnyLoc: Foundation Model features for VPR

Prior Visual Place Recognition (VPR) methods have shown great performance when tackling task-specific challenges in isolation. A generic out-of-the-box model will need to perform adequately well under a large set of challenges, including cases where there isn't enough data for training or when training on different domains is complex. This chapter introduces AnyLoc [18] which is the first step towards a universal VPR system.

VPR task is defined in Section 1.2.3. AnyLoc aims to solve image retrieval over a wide range of dataset settings using an off-the-shelf foundation model without any VPR-specific fine-tuning.

3.1 Prior Knowledge

3.1.1 VPR Baselines

Global image descriptors are already described in Section 1.2.3. We choose to compare with the following prominent baselines that are widely used in the VPR community.

NetVLAD [144] is a smooth (differentiable) version of VLAD (Vector of Locally Aggregated Descriptor). It is a weakly supervised contrastive learning method where the positive and negatives are mined by proximity (geographical distance). It is trained on the Pitts-250k dataset [161]. A major drawback is the negative mining technique that is infeasible in large scale settings and the high dimensional embeddings that lose significant performance when dimensionality reduction techniques like PCA are applied.

CosPlace [33] proposes a city-wide visual geo-localization solution by casting the training process as a classification problem. Itt partitions the dataset into classes based on the location (geographical coordinates) and the heading, and groups neighboring classes into groups. It then applies the Large Margin Cosine Loss (LCML), also known as CosFace [115], sequentially over each group; this loss introduces a cosine margin (through dot product) in the normalized version of the Softmax loss. CosPlace uses GeM pooling with output dimension 512 (much lesser than NetVLAD). It also proposes the San Francisco extra large (SF-XL) dataset for benchmarking city-wide VPR.

MixVPR [5] proposes a feature aggregation technique without self-attention or regional pooling, inspired by MLP-Mixer [71]. MLP-Mixer contains channel-mixing and token-mixing MLP layers. Given

a tokenized input (like in the first step of the transformer) $\mathbf{X} \in \mathbb{R}^{S,C}$, the mixer does channel-mixing to get $\mathbf{U} \in \mathbb{R}^{S,C}$ and token-mixing to get $\mathbf{V} \in \mathbb{R}^{S,C}$, given by

$$\mathbf{U}[:, i] = \mathbf{X}[:, i] + \mathbf{W}_2\sigma\left(\mathbf{W}_1\mathrm{LN}\left(\mathbf{X}[:, i]\right)\right), \qquad \text{for } i = 1 \dots C \tag{3.1}$$

$$\mathbf{Y}[j, :] = \mathbf{U}[j, :] + \mathbf{W}_4\sigma\left(\mathbf{W}_3\mathrm{LN}\left(\mathbf{U}[j, :]\right)\right), \qquad \text{for } j = 1 \dots S \tag{3.2}$$

This gives the network linear complexity in terms of number of patches (unlike the quadratic complexity of the attention mechanism) and the number of pixels in the input image. MixVPR first extract feature maps from intermediate layers of a CNN backbone, it then feeds these to a feature mixing block (consecutive MLP blocks) to incorporate global relationship in each feature map. MLPs are used to reduce the dimensionality (both row-wise and column-wise) and the final output is flattened and L2-normalized, giving the global descriptor for the input image. It is trained using Multi-Similarity Loss [105] using the GSV-Cities framework [31].

3.1.2 Foundation Models

This is discussed in detail in Chapter 2. We benchmark the capability of the following vision foundation models.

CLIP [66] is an image captioning model that is trained to align embeddings from a text and an image backbone. A dataset is distilled from YFCC100M [151] (filter images with English titles and descriptions) to get around 15M images (about the size of ImageNet). The text encoder is a modified transformer with some modifications [101, 131]. The vision encoder is either a ResNet [146] or ViT [82]. A minibatch contains a set of $(image, text)$ pairs, the distance between the corresponding pairs should be minimized while the non-corresponding pairs should be maximized. This type of contrastive setting is carried out over a very large batch size (of over 32,000). We ultimately get a trained visual encoder that can understand the contents of an image by language supervision.

DINO [51] and DINOv2 [21] are described in Section 2.3.1 and Section 2.3.2.2 respectively. They use knowledge distillation formulation (student-teacher with the same ViT architecture) with the cross entropy loss to learn representations of an image. DINO uses a simple DeIT implementation of ViT and ImageNet dataset, while DINOv2 uses a custom ViT implementation with many changes (to accommodate for the much larger model size) and a curated dataset LVD-142M.

Prior work [49] shows that DINO features (from the intermediate layers of the transformer) can be used to solve a range of tasks that require semantic understanding of the image, for example: co-segmentation of foreground, part co-segmentation, and image part correspondences.

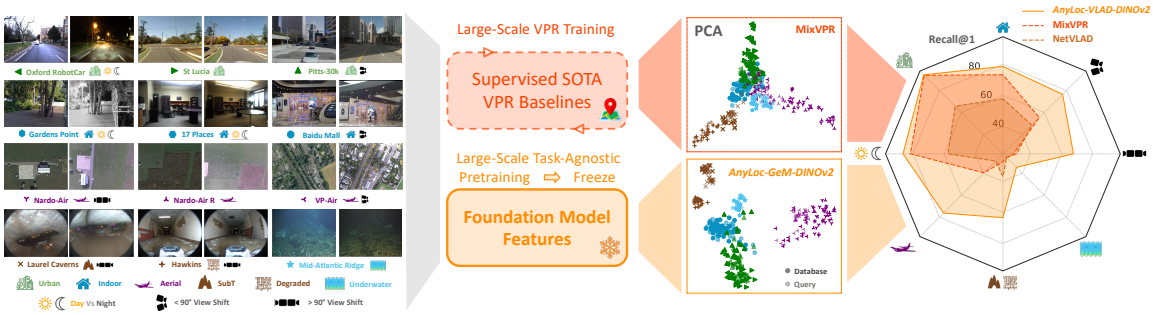


Figure 3.1 AnyLoc: Anywhere, Anytime, and Anyview VPR
Left: Image samples from various datasets used (and the legend for symbols and colors). *Center*: Comparing the PCA projections of MixVPR (a supervised baseline) and AnyLoc (foundation model features). *Right*: VPR performance by the test setting. Image from [18] (original paper).

3.2 AnyLoc: Towards Universal VPR

AnyLoc aims to perform good image retrieval (VPR) in diverse places (anywhere), under environmental changes (anytime), and under high viewpoint changes (anyview). For this, we extract the representations learned by the intermediate, often penultimate, transformer layers of DINO and DINOv2. We find that conventional feature aggregation techniques like VLAD and GeM, described in Section 1.2.3, bring significant benefit over using only the output $[CLS]$ token of these models. Our method, when projected to 2D, also discovers unique features that distinguish and group datasets of similar domain, which is lacking in most recent VPR methods (see center of Fig. 3.1 for comparison with MixVPR). The process and steps for AnyLoc are described below

3.2.1 Constructing AnyLoc Pipeline

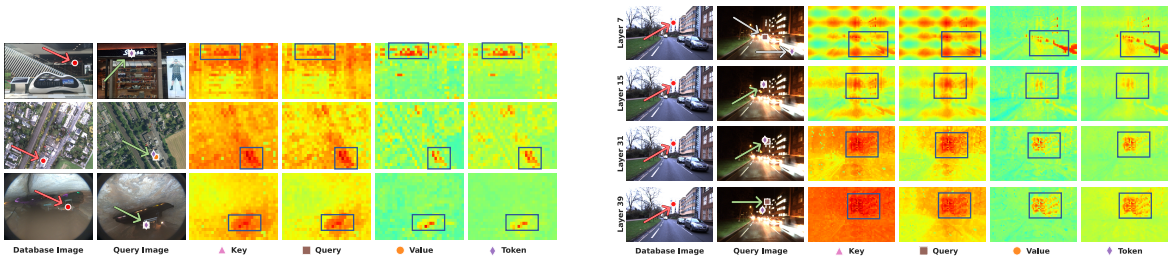
3.2.1.1 Choice of Foundation Model

From CLIP [66], MAE [58], DINO [51], and DINOv2 [21], AnyLoc employs DINO and DINOv2 ViTs for extracting features. This is because DINO and DINOv2 are trained for representation learning (through knowledge distillation - as described in Section 2.3.1 and Section 2.3.2.2). CLIP is trained on alignment and is used as a benchmarking method. MAE is trained to in-fill masked tokens (like the MIM objective in iBOT and DINOv2) and without any special dataset curation. We find CLIP and DINO methods perform better than pure masked modeling of MAE.

We use *DINO* and *DINOv2* models for extracting intermediate features and CLIP for benchmarking.

3.2.1.2 Choosing Layer and Facet

A vision transformer, described in Section 2.1.1, patchifies an input image and passes it through layers of transformer encoder. The transformer encoder consists of multi-headed self-attention (MHA) modules (alongside the normalization, residual, and FFN blocks). Each MHA has a softmax self-attention mechanism. The input to attention are the `query`, `key`, and `value` facets. The output of each transformer encoder (layer) is called the `value` facet.



(a) Facet **(b)** Layer

Figure 3.2 Facet and Layer-wise Attention Maps for DINOv2
Left: Given a query point in the database image, various facets show different levels of contrast (with the most similar marker on the query image). Top has high scale change, middle has rotation change, and bottom has high perspective change. *Right*: A similar comparison across layers shows varying contrast. Notice that layer 31 shows the building (for the queried point in database image) with the best contrast.

Description Let the input to the model be $\mathbf{x}^{(0)} \in \mathbb{R}^{n,d_m}$ and the input to the l-th transformer encoder layer be $\mathbf{x}^{(l-1)} \in \mathbb{R}^{n,d_m}$ (also the output of the previous layer). This goes through a normalization layer as $\bar{\mathbf{x}}^{(l-1)} = \text{LN}(\mathbf{x}^{(l-1)}) \in \mathbb{R}^{n,d_m}$ and projected to the query, key, and value facets as follows

$$\mathbf{q}^l = \bar{\mathbf{x}}^{(l-1)}\mathbf{W}_{ql} \qquad \mathbf{k}^l = \bar{\mathbf{x}}^{(l-1)}\mathbf{W}_{kl} \qquad \mathbf{v}^l = \bar{\mathbf{x}}^{(l-1)}\mathbf{W}_{vl} \tag{3.3}$$

Where $\mathbf{W}_{ql}, \mathbf{W}_{kl}, \mathbf{W}_{vl} \in \mathbb{R}^{d_m,d_m}$ are the projection weights. These facets are split into h heads and attention is applied to each head. The output of the attention's softmax is added with $\mathbf{x}^{(l-1)}$. This goes through another layer normalization and FFN (in a residual) and the final output of layer l is given as $\mathbf{x}^{(l)}$.

We say $\mathbf{q}^l$ is query, $\mathbf{k}^l$ is key, $\mathbf{v}^l$ is value, and $\mathbf{x}^l$ is token facet of the l-th layer. Each is of type $\mathbb{R}^{n,d_m}$ (n tokens, d_m dimension each) and is a candidate for feature aggregation.

Process Given a dataset containing database and query images, we pick a representative sample and pick a few corresponding points. As seen in Fig. 3.2, a point in the left database image is chosen and all patches (across all facets) are extracted from the query image (on the right) and matched. A map highlighting the similarity of each patch to the queried point is visualized across facets and layers. The most similar patch is highlighted in the query image. On inspecting the similarity maps, we can arrive at a decision on using a particular facet and layer. In the case of Fig. 3.2, we choose the `value` facet of the 31st layer (because the maps are most contrastive).

3.2.1.3 Choosing Aggregation Method

The facet features correspond to the individual patches of the image (mostly local features with some semantic image information imbibed due to training). These local features have to be aggregated to obtain image-level features (also called global descriptors). These techniques are described in Section 1.2.3. We explore the following methods

VLAD Vector of Locally Aggregated Descriptor is formed by computing a vocabulary from a database of images, like the cluster centers by clustering the local features. Given an image, the features are

assigned to the clusters and a residual is calculated (distance to the cluster centers). The residuals are then aggregated for each cluster, normalized, concatenated into one vector, and normalized again.

Given features $\mathbf{f} \in \mathbb{R}^{n,d_m}$ (which could come from any facet of any layer mentioned earlier), we first get the residual matrix $\mathbf{V} \in \mathbb{R}^{K,d_m}$

$$\mathbf{V}[k,:] = \sum_{i=1}^{n} m_{i,k}(\mathbf{f}[i,:] - \mathbf{c}[k,:]) \tag{3.4}$$

where $\mathbf{c} \in \mathbb{R}^{K,d_m}$ is the vocabulary of K cluster centers, each d_m dimensional. And $m_{i,k} = 1$ if feature i is assigned to cluster k (closest to the cluster) and 0 otherwise.

Soft VLAD replaces the $m_{i,k}$ with a softmax on the distance between the feature i and every cluster center k. This gives distance-proportional assignment of every descriptor to every cluster center. This is similar to the formulation of NetVLAD.

GeM applies a mean pooling over the descriptors $\mathbf{f} \in \mathbb{R}^{n,d_m}$ and gives pooled descriptor $\mathbf{f}_G \in \mathbb{R}^{n,d_m}$ using

$$\mathbf{f}_G = \left(\frac{1}{n} \sum_{i=1}^{n} (\mathbf{f}[i,:])^p \right)^{\frac{1}{p}} \tag{3.5}$$

3.2.1.4 Choosing Vocabulary

Algorithms like VLAD require a set of database images to form cluster centers (vocabulary). We aim to characterize distinctive semantic properties in a diverse set of environments. From Fig. 3.1, it is apparent that projecting the global descriptors of different datasets highlights distinct domains in the latent space, allowing us to characterize and group datasets having similar properties.

The vocabulary is a superset (a collection of datasets) of images we use to build database descriptors (for aggregation technique VLAD). We define the following types of vocabulary

- *Global*: Where database images from all the datasets are included when clustering.
- *Structured*: Where only the database images from structured environments are included for clustering. These are outdoor and indoor datasets. These are most widely benchmarked in the VPR community and are from mostly human maintained environments.
- *Unstructured*: Where database images from unstructured environments are included. These are subterranean, degraded, aerial, and underwater datasets. These aren't frequently benchmarked and remain less explored, but are important in scenarios like remote monitoring.
- *Map-Specific* or dataset-specific: Where the database images from only the single dataset (which is being tested on) is used. For example, if the testing is on Oxford cars (an outdoor dataset), then the database images only from Oxford car are used to get the cluster centers for VLAD.

Structured					Unstructured				
Dataset	N_{Db}	N_q	Loc.	Type	Dataset	N_{Db}	N_q	Loc.	Type
Baidu [127]	689	2292	10 m		Hawkins [28]	65	101	8 m	
Gardens [56, 150]	200	200	5 fr		Laurel [28]	141	112	8 m	
17 Places [140]	406	406	5 fr		Nardo-Air [14]	102	71	60 m	
Pitts-30k [161]	10k	6816	25 m		VP-Air [45]	2.7k	2.7k	3 fr	
St. Lucia [177]	1549	1464	25 m		Mid-Atlantic [8]	65	101	0.3 m	
Oxford [121]	191	191	25 m						

Table 3.1 Datasets Used for Benchmarking
Baidu stands for "Baidu Mall", Gardens stands for "Gardens Point", Laurel stands for "Laurel Caverns", Mid-Atlantic stands for "Mid-Atlantic Ridge" or "Eiffel Tower" (underwater hydrothermal vent in the atlantic). Localization radius is in meters (m) or number of frames (fm).

- *Domain-Specific*: A domain is a collection of datasets with similar properties. The latent representation of AnyLoc descriptors allows us to discover these groups by clustering image descriptors from various datasets (as shown in Fig. 3.1). For example, for the outdoor domain, we use all the datasets captured in the outdoor setting.

After getting global descriptors (using aggregation techniques), we store the descriptors from database images into a local cache and run similarity search for the descriptors of query images using Cosine similarity. Note that the database cache is built offline whereas the query's nearest neighbor lookup can be done online.

3.3 Experimental Setup

3.3.1 Datasets

All datasets used for benchmarking are mentioned in Table 3.1, along with the type (see Fig. 3.1 for legend), number of database images $\mathbf{N_{Db}}$, number of query images $\mathbf{N_q}$, and the localization radius used for evaluation purposes. We choose a diverse set of structured and unstructured datasets. Structured datasets include commonly used indoor and outdoor datasets whereas unstructured datasets are less commonly used, but are gaining traction in VPR because of their challenging nature and the requirements of robotics in the near future (subterranean and underwater datasets, for example).

To create an aerial dataset with no rotation shift between database and query images, we create `Nardo-Air R`, which is created by rotating the Nardo-Air images so that they all point in the same direction.

3.3.2 Benchmarking

Baselines We evaluate AnyLoc against three state of the art VPR baselines: NetVLAD [144], CosPlace [33], and MixVPR [5]. To evaluate against other foundation models, we additionally propose benchmarking CLIP [66], DINO [51], and DINOv2 [21] using their $[CLS]$ token as the global image descriptor

and performing retrieval against it. For CLIP, we use the OpenCLIP implementation trained on the LAION-5B dataset [59, 9, 66, 46], specifically, we use the `ViT-BigG/14` model. We use `ViT-S/8` (DeIT implementation) for DINO and `ViT-G/14` for DINOv2.

Model Specifications For *AnyLoc-VLAD-DINO*, we perform VLAD aggregation over DINO features extracted from layer 9, `key` facet, of the `ViT-S/8` (DeIT) model. For *AnyLoc-VLAD-DINOv2*, we perform VLAD over DINOv2 features from layer 31, `value` facet, of the `ViT-G/14` model. For *AnyLoc-GeM-DINOv2*, we extract the same features as in the VLAD version, but perform GeM pooling instead of VLAD aggregation.

For *DINO*, the descriptor dimension for ViT-S (DeIT) is $d_m = 384$ (for each patch), and VLAD clustering is used with $K = 128$ clusters, giving a total output dimension of $384 \times 128 = 49152$. For *DINOv2*, the descriptor dimension for ViT-G/14 is $d_m = 1536$ and we use $K = 32$ clusters, giving the VLAD output dimension $1536 \times 32 = 49152$. We additionally down-sample these global descriptors using principal component analysis (PCA). We fit the PCA (estimate the parameters) using only the database images, which is done offline. The same (cached) projection parameters are used on the query images (which are fed online). We project the dimensionality from 49152 to 512 in *AnyLoc-VLAD-DINO-PCA* and *AnyLoc-VLAD-DINOv2-PCA* methods.

3.3.3 Evaluation

We use the widely used `Recall@N` metric to evaluate image retrieval (VPR) performance on each dataset. Recall@N (or R@N) is defined as the percentage of queries that have a correct retrieval in the top-N closest retrieved database images. A good VPR method should have high recall at low 'N' value. As 'N' increases, the recall of a particular method also increases (as it is more likely to get a correct retrieval if it's matching against more retrieved images). We use 'R@1' (top-1 retrieved) and 'R@5' (top-5 retrieved) to compare all methods.

3.4 Results

3.4.1 Overall

The results for indoor, outdoor, aerial, and remaining unstructured datasets are presented in Table 3.2, Table 3.3, Table 3.4, and Table 3.5 respectively.

It is interesting to note that even with a $96\times$ reduction descriptor dimension through PCA, AnyLoc methods hardly loose any performance. This robustness to dimensionality reduction could be attributed to regularization techniques during training these foundation models, promoting the learning of efficient representations across channels. This robustness could also be a reason why AnyLoc can easily separate and group datasets of different domains in the 2D PCA projection in Fig. 3.1 whereas MixVPR struggles to separate the data in such a low dimension.

AnyLoc methods perform much better than the benchmark for the indoor domain. Interestingly, the `CLS` baselines of foundation models are competitive with conventional VPR baselines out-of-the-box.

Methods	Baidu Mall		Gardens Point		17 Places		*Average*	
	R@1	R@5	R@1	R@5	R@1	R@5	R@1	R@5
NetVLAD [144]	53.1	70.5	58.5	85.0	61.6	77.8	57.73	77.76
CosPlace [33]	41.6	55.0	74.0	94.5	61.1	76.1	58.9	75.2
MixVPR [5]	64.4	80.3	91.5	96.0	63.8	78.8	73.23	85.03
CLIP-CLS [66]	56.0	71.6	42.5	74.5	59.4	77.6	52.63	74.56
DINO-CLS [51]	48.3	65.1	78.5	95.0	61.8	76.4	62.86	78.83
DINOv2-CLS [21]	49.2	64.6	71.5	96.0	61.8	78.8	60.83	79.8
AnyLoc-GeM-DINOv2	50.1	70.6	88.0	97.5	63.6	79.6	67.23	82.56
AnyLoc-VLAD-DINO	61.2	78.3	95.0	98.5	63.8	78.8	73.33	85.2
AnyLoc-VLAD-DINO-PCA	62.3	81.2	91.5	**99.5**	63.3	78.8	72.36	86.5
AnyLoc-VLAD-DINOv2	**75.2**	87.6	95.5	**99.5**	**65.0**	80.5	**78.56**	89.2
AnyLoc-VLAD-DINOv2-PCA	74.9	**89.4**	**96.0**	**99.5**	64.8	**81.0**	**78.56**	**89.96**

Table 3.2 AnyLoc Benchmarking on Indoor Datasets
Performance of methods on indoor datasets. *Top*: Common VPR baselines, *middle*: foundation model global descriptors (using CLS token), *bottom*: our AnyLoc methods. The best is shown in **bold**, and the second best is shown in underline.

Methods	Pitts-30k		St. Lucia		Oxford		*Average*	
	R@1	R@5	R@1	R@5	R@1	R@5	R@1	R@5
NetVLAD [144]	86.1	92.7	57.9	73.0	57.6	79.1	67.2	81.6
CosPlace [33]	90.4	**95.7**	99.6	99.9	95.3	99.5	**95.1**	98.36
MixVPR [5]	**91.5**	95.5	**99.7**	**100**	92.7	99.5	94.63	98.33
CLIP-CLS [66]	55.0	77.2	62.7	80.7	46.6	60.7	54.76	72.86
DINO-CLS [51]	70.1	86.4	45.2	64.0	20.4	46.6	42.23	65.66
DINOv2-CLS [21]	78.3	91.1	78.6	89.7	47.1	58.1	68	79.63
AnyLoc-GeM-DINOv2	77.0	87.3	76.9	89.3	92.2	97.9	82.03	91.5
AnyLoc-VLAD-DINO	83.4	92.0	88.5	94.9	82.2	99.0	84.7	95.3
AnyLoc-VLAD-DINO-PCA	82.8	90.8	87.6	94.3	82.7	96.3	84.36	93.8
AnyLoc-VLAD-DINOv2	87.7	94.7	96.2	98.8	**99.5**	**100**	94.46	97.83
AnyLoc-VLAD-DINOv2-PCA	86.9	93.8	96.4	99.5	96.9	**100**	93.4	97.76

Table 3.3 AnyLoc Benchmarking on Outdoor Datasets
Performance of methods on outdoor datasets. *Top*: Common VPR baselines, *middle*: foundation model global descriptors (using CLS token), *bottom*: our AnyLoc methods. The best is shown in **bold**, and the second best is shown in underline.

Methods	Nardo-Air		Nardo-Air R		VP-Air		*Average*	
	R@1	R@5	R@1	R@5	R@1	R@5	R@1	R@5
NetVLAD [144]	19.7	39.4	60.6	85.9	6.4	17.7	28.9	47.66
CosPlace [33]	0	1.4	91.6	**100**	8.1	14.2	33.23	38.53
MixVPR [5]	32.4	42.2	76.1	98.6	10.3	18.3	39.6	53.03
CLIP-CLS [66]	42.2	70.4	62.0	97.2	36.6	52.8	46.93	73.46
DINO-CLS [51]	57.8	90.1	84.5	**100**	24.0	38.4	55.43	76.16
DINOv2-CLS [21]	73.2	88.7	71.8	91.6	45.2	59.9	63.4	80.06
AnyLoc-GeM-DINOv2	**76.1**	83.1	57.8	97.2	38.3	53.8	57.4	78.03
AnyLoc-VLAD-DINO	43.7	54.9	**94.4**	**100**	17.8	28.7	51.96	61.2
AnyLoc-VLAD-DINOv2	**76.1**	**94.4**	85.9	**100**	**66.7**	**79.2**	**76.23**	**91.2**

Table 3.4 AnyLoc Benchmarking on Aerial Datasets
Performance of methods on aerial datasets. *Top*: Common VPR baselines, *middle*: foundation model global descriptors (using CLS token), *bottom*: our AnyLoc methods. The best is shown in **bold**, and the second best is shown in underline. Note that aerial datasets are also included in our classification of unstructured datasets. However, they also form a domain of their own, hence these results are in a separate table.

Methods	Hawkins		Laurel Caverns		Mid-Atlantic Ridge	
	R@1	R@5	R@1	R@5	R@1	R@5
NetVLAD [144]	34.8	71.2	39.3	71.4	25.7	53.5
CosPlace [33]	31.4	59.3	24.1	47.3	20.8	40.6
MixVPR [5]	25.4	60.2	29.5	67.0	25.7	60.4
CLIP-CLS [66]	33.0	67.0	36.6	66.1	25.7	51.5
DINO-CLS [51]	46.6	84.8	41.1	57.1	27.7	49.5
DINOv2-CLS [21]	28.0	62.7	40.2	65.2	24.8	48.5
AnyLoc-GeM-DINOv2	53.4	83.9	58.9	86.6	14.8	49.5
AnyLoc-VLAD-DINO	48.3	84.8	57.1	79.5	**41.6**	**66.3**
AnyLoc-VLAD-DINOv2	**65.2**	**94.1**	**61.6**	90.2	34.6	61.4

Table 3.5 AnyLoc Benchmarking on Unstructured Datasets
Performance of methods on unstructured datasets (outside aerial domain). *Top*: Common VPR baselines, *middle*: foundation model global descriptors (using CLS token), *bottom*: our AnyLoc methods. The best is shown in **bold**, and the second best is shown in underline.

Vocabulary Type	Indoor	Outdoor	Aerial
Global	77.0	93.9	57.1
Structured	77.0	93.3	56.4
Unstructured	74.8	89.0	75.8
Map-Specific	78.0	92.3	62.9
Domain-Specific	**78.6**	**94.4**	**76.2**

Table 3.6 Vocabulary Analysis using AnyLoc-VLAD-DINOv2
Different vocabularies (first column) are tried against different dataset domains (heading of next three columns). The 'R@1' is calculated and averaged across the domains. The best results are in **bold**

Simple aggregation techniques over DINO and DINOv2 give a new state-of-the-art model in this category.

Most `CLS` baselines fail in outdoor settings, due to not being explicitly trained on outdoor data; only DINOv2 has Google Landmarks so it performs relatively better than DINO and CLIP. However, adding conventional aggregation techniques to foundation model baselines bring them at-par with the VPR baselines that are explicitly trained for outdoor VPR. AnyLoc DINOv2 methods, perform better than VPR baselines in the presence of day-vs-night change (Oxford).

Traditional VPR methods, though being trained on ourdoor data, perform poorly on aerial data. This could be because they fail to generalize on top-view images. Upon visual inspection of retrievals, we find that CosPlace retrieves a particular fixed set of reference images for almost all query images (giving poor results on Nardo-Air).

Foundation models and AnyLoc methods are significantly better than VPR baselines on aerial datasets. They set a new state-of-the-art, with the worst performer (CLIP-`CLS`) beating the best traditional VPR method (MixVPR) by more than 7%. AnyLoc-VLAD methods, especially with DINOv2, yield robust results that are unseen on previous benchmarks.

The performance of AnyLoc-VLAD-DINOv2 on Laurel Caverns and Mid-Atlantic Ridge is significantly better than VPR baselines (and even foundation models), showcasing the utility of the methods even under conditions where distinct keypoints are few and when there is a high viewpoint change involved.

3.4.2 Vocabulary Domain Ablation

We study the effect of using different vocabularies (collection of datasets) for different deployment scenarios. The overall results are shown in Table 3.6. It is evident that using the domain-specific vocabulary for each domain gives the best result; that is, using database images across various indoor datasets is the best option to test on an indoor dataset (even better than using database images only from the single dataset being tested). This shows good knowledge transfer abilities of AnyLoc. Interestingly, using unstructured vocabulary for structured domains (indoor and outdoor) degrades performance by only 4%, but gives significant boost to evaluations on unstructured datasets (aerial).

Vocabulary Dataset	Evaluation Dataset	Map-Specific R@1	Vocab-Specific R@1
Baidu Mall (0.7k)	17 Places (0.4k)	**64.5**	63.8
	Gardens Point (0.2k)	**98.0**	94.5
VP-Air (2.7k)	Nardo-Air (0.1k)	57.8	**64.8**
	Nardo-Air R (0.1k)	70.4	**88.7**
Pitts-30k (10k)	Oxford (0.2k)	94.8	**99.0**

Table 3.7 Vocabulary Transfer for AnyLoc-VLAD-DINOv2
Using a large vocabulary dataset, we test how well the cluster centers trained on the testing dataset (map-specific) perform w.r.t. those transferred form the given dataset (vocab-specific). The first column is the large dataset (for vocab testing), second column is the dataset for evaluation, third column is results using the evaluation dataset alone (no transfer), fourth column is results using transfer of vocab. Top row is for indoor domain, middle is for aerial, and bottom is for outdoor.

3.4.2.1 Vocabulary Transfer Behavior

We additionally diagnose the vocabulary transfer abilities of our model by using one large dataset for building vocabulary, and testing on other datasets of the same domain. The results for this are shown in Table 3.7. It is clear that, in case of using vocabulary transfer from another dataset, the vocabulary dataset should be large. Using cluster centers obtained from Pitts-30k to evaluate Oxford gives a significant boost of more than 4% (over using cluster centers from only Oxford).

This shows that we can save the vocabulary (cluster centers) in long-term storage and use them based on the deployment environment. One way to estimate the deployment environment could be to project sample images on a template projection map containing images from different domains, like the PCA projections by AnyLoc-GeM-DINOv2 in Fig. 3.1.

3.4.2.2 Visual Analysis of Vocabulary

We visualize the cluster assignments in Fig. 3.3. This is done by color coding the cluster numbers and projecting the assigned numbers (closest cluster center) to each patch (since descriptors are patch-wise). We notice that most clusters learn (latch onto) semantically meaningful information needed for good VPR. For example, the buildings in aerial images (green), furniture in indoor images (purple), and road, buildings, and horizon in outdoor images (orange, pink, and blue respectively). For the purpose of this experiment, we use $K = 8$ clusters.

3.4.3 Design Ablation

To better justify our choice of model parameters like layer, facet, and pooling techniques, we run thorough ablations in Fig. 3.4 (tested on Baidu Mall and Oxford datasets).

We see that using ViT-G (with 1.1B parameters) gives the best results on the test datasets. ViT-L (with 300M parameters) gives the closest best result, followed by ViT-B (with 86M parameters) and ViT-S (with 21M parameters). The low change when downgrading from ViT-G to ViT-L shows that DINOv2 retains performance even after a large reduction in parameters. This could be a result of the distillation strategy of DINOv2 (smaller models are distilled from larger models) discussed in Section 2.3.2.2.

On average, using the `key` facet for DINO and `value` facet for DINOv2 gives the best results

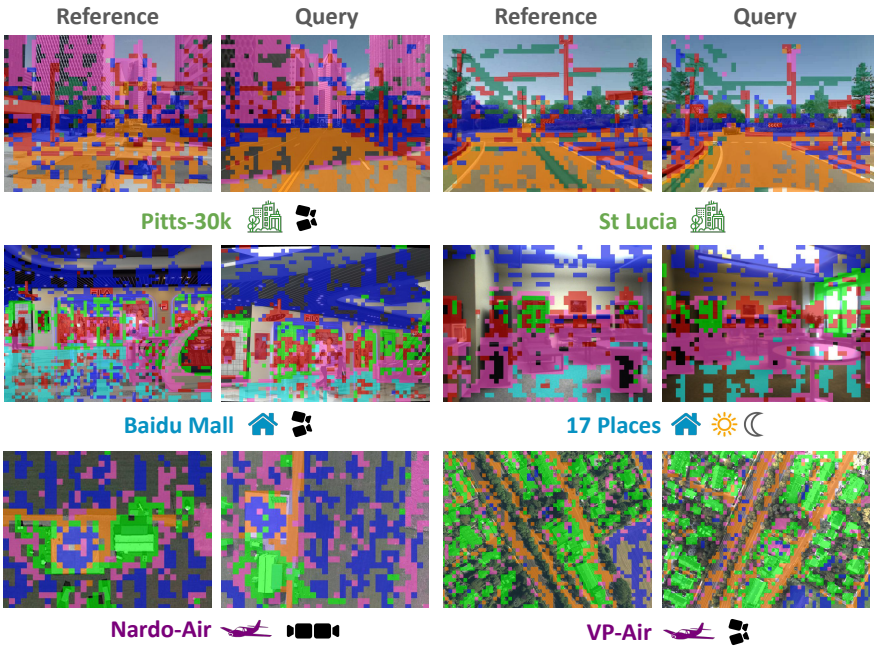
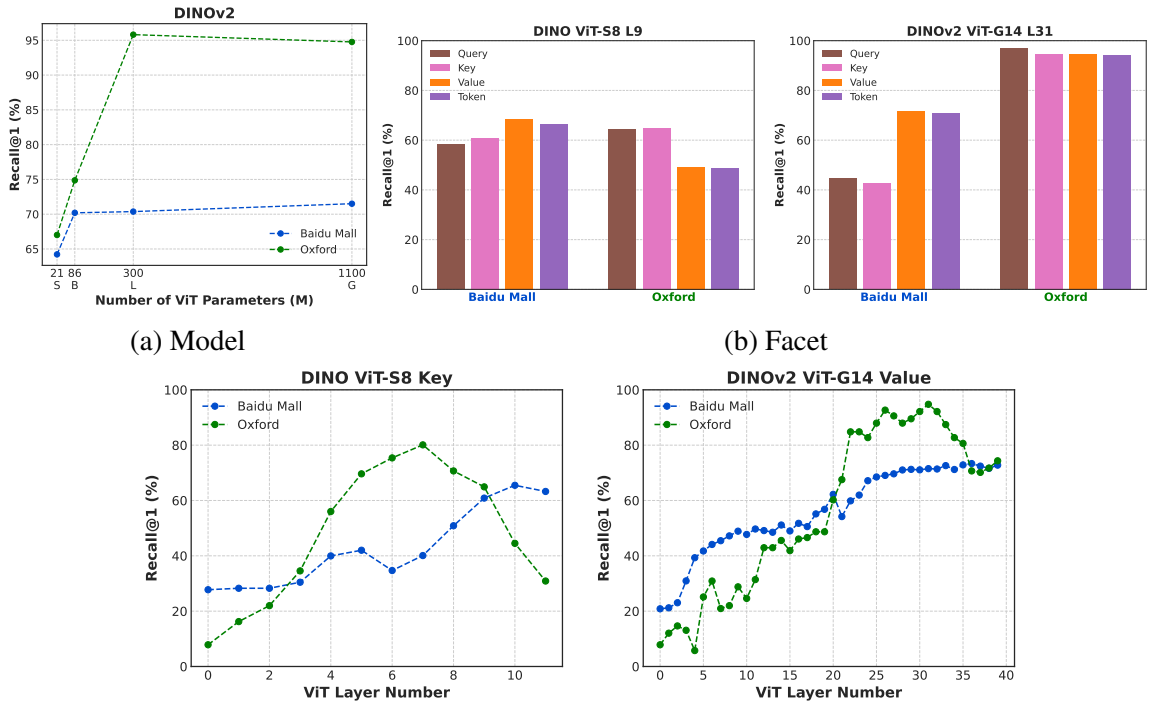


Figure 3.3 VLAD Cluster Visualizations
Visualizing the clusters different patches of the image belong to. We use domain vocabulary (clusters) for this purpose. In each row, patches belonging to the same cluster have the same color. Noisy clusters are not visualized for clarity.



(a) Model (b) Facet

(c) Layer

Figure 3.4 AnyLoc Design Ablations
(a) Performance varying by model size. (b) On average, `key` and `value` perform best for DINO and DINOv2 respectively. (c) Performance peaks at penultimate layers of DINO and DINOv2.

Method	Indoor	Urban	Aerial	SubT & D	Underwater
ViT-B CosPlace	62.9	80.7	26.3	26.5	18.8
ViT-B CosPlace VLAD	68.5	82.9	38.4	37.5	23.8
ViT-S AnyLoc-VLAD-DINO	72.9	79.6	47.8	52.7	**41.6**
ViT-B AnyLoc-VLAD-DINOv2	77.0	82.6	53.6	60.2	35.6
ViT-G AnyLoc-VLAD-DINOv2	**78.0**	**92.3**	**62.9**	**63.4**	34.6

Table 3.8 VPR Trained ViTs Vs. Self Supervised ViTs
Analysis of ViTs trained for VPR (top rows) versus self-supervised ViTs (bottom rows). Best results are in **bold**. All numbers are Recall@1 percentages.

Aggregation Methods	DINO			DINOv2		
	Baidu ↑	Oxford ↑	Dim ↓	Baidu ↑	Oxford ↑	Dim ↓
Global Average Pooling (GAP)	29.6	28.8	**384**	41.6	78.5	**1536**
Global Max Pooling (GMP)	34.9	38.2	**384**	64.4	74.9	**1536**
Generalized Mean Pooling (GeM)	34.7	47.6	**384**	50.1	92.2	**1536**
Soft Assignment VLAD	33.8	28.3	49152	40.3	82.2	49152
Hard Assignment VLAD	**60.9**	**64.9**	49152	**71.5**	**94.8**	49152

Table 3.9 AnyLoc Aggregation Methods
Testing various feature aggregation methods for AnyLoc using DINO and DINOv2 features (same facet and layer as AnyLoc). Note that for DINOv2, through VLAD has high dimension, the PCA projected version has only 512 dimension. It is not included here for fairness. The numbers in `Dim` column are integers (lower is better) and the numbers in other columns are Recall@1 percentages (higher is better).

across dataset domains. The layer ablations show that performance is low at the initial layers, peaks at the penultimate layers, and then gradually stagnates or goes down. We believe this is because of unique representations learned per layer. It is possible that later layers learn more meaningful global representations at the cost of local information. As a good middle ground, we choose `layer 9` for `DINO ViT-S/8` and `layer 31` for `DINOv2 ViTT-G/14`.

3.4.3.1 Architecture

We additionally investigate the affect of the architecture on recalls in Table 3.8. We use the author's implementation of GeM pooling on ViT-B's output [33]. We also test this against VLAD aggregation ($K = 128$ clusters) over the 6th layer's values for CosPlace ViT-B (we found this performs best).

We find that AnyLoc gives better results than CosPlace ViT, despite using a smaller ViT (ViT-S). On the urban setting, since CosPlace is trained on it, it performs slightly better. However, we achieve the best performance using ViT-G AnyLoc-VLAD-DINOv2.

3.4.3.2 Pooling Methods

We try different pooling methods in Table 3.9. We find that VLAD pooling with hard cluster assignments gives the best results, followed by GeM pooling. Surprisingly, these methods do not perform well when using soft cluster assignments in VLAD (like in NetVLAD). This could be because these models aren't trained for VLAD.

3.5 Conclusion

This work introduces AnyLoc, a novel general-purpose visual place recognition (image retrieval) system achieving state-of-the-art performance across diverse domains (indoor, aerial, unstructured) under varying conditions (day/night, viewpoints). By leveraging features from foundation models and unsupervised aggregation techniques, AnyLoc outperforms existing VPR systems by up to 5%, 35%, and 30% in these domains, respectively. Notably, it demonstrates competitive results even on outdoor environments without specific supervision or training data, highlighting its generalizability. Developments like AnyLoc are crucial for enabling robust robot navigation and perception in real-world scenarios with limited or unavailable domain-specific data.

Chapter 4

Conclusions

Our introduction (Chapter 1) established the field of mobile robotics as the primary domain of this work (Section 1.2.1). It also provided an overview of simultaneous localization and mapping (SLAM) systems (Section 1.2.2), visual place recognition (VPR) systems (Section 1.2.3), and image retrieval (IR) techniques using global image descriptors.

Chapter 2 delves into Vision Foundation Models (VFMs). It discusses their core architecture, the Vision Transformer, focusing on both ViT (Section 2.1.1) and DeIT (Section 2.1.2). The chapter also explores self-supervised learning concepts crucial for this thesis, such as knowledge distillation (Section 2.2.1).

Furthermore, Chapter 2 introduces the specific foundation models employed in this work: DINO (Section 2.3.1) and DINOv2 (Section 2.3.2.2). It's important to note that DINOv2 relies on iBOT (Section 2.3.2.1) as an underlying dependency.

Leveraging the power of vision foundation models and insights from traditional VPR systems, Chapter 3 introduced AnyLoc, our novel image retrieval system for VPR. AnyLoc demonstrates the significant promise that foundation models hold for tackling complex visual place recognition tasks, particularly in scenarios with limited data availability.

This chapter concludes this thesis with the limitations of our system and some possible solutions to them. Following on latest developments, it also includes notes for improvements in the future.

4.1 Limitations and Suggestions

Descriptor Dimensionality While AnyLoc achieves state-of-the-art performance across diverse domains (see Tables 3.2 to 3.5), its high-dimensional descriptors (49,152 elements) pose challenges for deployment in memory-constrained environments. To address this, we employ Principal Component Analysis (PCA) for efficient dimensionality reduction to 512 elements, with minimal to no performance impact. Exploring alternative feature aggregation and dimensionality reduction techniques holds promise for further improvements in this area.

Size and Latency The backbone of our most efficient model is ViT-G, which has 1.1B parameters and requires 6 GB VRAM under low-power hardware settings. This large model size presents challenges for deployment in real-world environments with limited memory resources.

ViT-G also exhibits significant latency during loading and forward passes (AnyLoc takes about 0.2 seconds for some images). This latency can hinder real-time applications that require fast turnaround times.

Some potential solutions to the above problems could include

- *Efficient Transformer Architectures*: Investigating foundation models trained using more efficient transformer architectures, such as CCT, could lead to smaller models that maintain a good balance between memory footprint, accuracy, and speed.
- *Knowledge Distillation and VPR-Specific Supervision*: Exploring a combination of knowledge distillation and VPR-specific supervision techniques holds significant potential for creating more efficient VFMs specifically tailored for visual place recognition tasks.
- *Implementation Optimizations*: Utilizing lower-level code optimizations like those offered by NVIDIA TensorRT [48] or specialized hardware like FPGAs [47] could further unlock the potential of the ViT architecture in terms of speed and throughput.

Other Foundation Models The recent emergence of large language models (LLMs) equipped with vision capabilities, such as GPT-4V [26] and Google Gemini [11], opens exciting possibilities for visual place recognition (VPR). While our experiments with MAE [58] and CLIP [59] under our specific settings did not yield significant results, these models remain largely unexplored in the context of VPR tasks.

Other Concurrent Work Includes using foundation models for autonomous driving [24] and generative models to create worlds (for training robust models) [7] (MUVO), [16] (GAIA). These models could also possibly be learning world representations necessary for robust VPR.

4.2 Future Scope

Recent advancements in the Vision Transformer (ViT) field, such as register tokens [10], hold promise for further improvements in AnyLoc. These tokens can potentially help ViTs learn more effective latent maps, which could translate to enhanced performance in visual place recognition tasks. Similarly, token reorganization techniques like those explored in [40] (using specific tokens instead of all) and architectural changes like variations in normalization position [92] could further optimize AnyLoc's efficiency and accuracy. By integrating these advancements and potentially exploring their synergistic effects, future iterations of AnyLoc could achieve even greater performance and pave the way for seamless integration into a full SLAM system, aiding mobile robots in tasks like real-time localization and environment understanding.

Related Publications

1. Keetha, N.V., *Mishra, A.*, Karhade, J., Jatavallabhula, K., Scherer, S.A., Krishna, M., & Garg, S. (2023). AnyLoc: Towards Universal Visual Place Recognition. *IEEE Robotics and Automation Letters*, 9, 1286-1293. doi: 10.1109/LRA.2023.3343602 (arXiv: 2308.00688)

Bibliography

[1] Marcus Abate et al. “Kimera2: Robust and Accurate Metric-Semantic SLAM in the Real World”. In: *ArXiv* abs/2401.06323 (2024). URL: https://api.semanticscholar.org/CorpusID:266977414.

[2] Lukas Schmid et al. “Khronos: A Unified Approach for Spatio-Temporal Metric-Semantic SLAM in Dynamic Environments”. In: 2024. URL: https://api.semanticscholar.org/CorpusID:267770462.

[3] Fabio Tosi et al. “How NeRFs and 3D Gaussian Splatting are Reshaping SLAM: a Survey”. In: 2024. URL: https://api.semanticscholar.org/CorpusID:267759760.

[4] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. *Ceres Solver*. Version 2.2. Oct. 2023. URL: https://github.com/ceres-solver/ceres-solver.

[5] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. “MixVPR: Feature Mixing for Visual Place Recognition”. In: *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2023), pp. 2997–3006. URL: https://api.semanticscholar.org/CorpusID:256646815.

[6] Randall Balestriero et al. “A Cookbook of Self-Supervised Learning”. In: *ArXiv* abs/2304.12210 (2023). URL: https://api.semanticscholar.org/CorpusID:258298825.

[7] Daniel Bogdoll, Yitian Yang, and J. Marius Zollner. “MUVO: A Multimodal Generative World Model for Autonomous Driving with Geometric Representations”. In: *ArXiv* abs/2311.11762 (2023). URL: https://api.semanticscholar.org/CorpusID:265295410.

[8] Clémentin Boittiaux et al. “Eiffel Tower: A deep-sea underwater dataset for long-term visual localization”. In: *The International Journal of Robotics Research* 42 (2023), pp. 689–699. URL: https://api.semanticscholar.org/CorpusID:258564327.

[9] Mehdi Cherti et al. “Reproducible scaling laws for contrastive language-image learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 2818–2829.

[10] Timoth’ee Darcet et al. “Vision Transformers Need Registers”. In: *ArXiv* abs/2309.16588 (2023). URL: https://api.semanticscholar.org/CorpusID:263134283.

[11] Gemini Team Google. “Gemini: A Family of Highly Capable Multimodal Models”. In: *ArXiv* abs/2312.11805 (2023). URL: https://api.semanticscholar.org/CorpusID:266361876.

[12] Jie Gui et al. “A Survey of Self-Supervised Learning from Multiple Perspectives: Algorithms, Theory, Applications and Future Trends”. In: *ArXiv* abs/2301.05712 (2023). URL: https://api.semanticscholar.org/CorpusID:255941551.

[13] Xiaoye Han et al. “RO-MAP: Real-Time Multi-Object Mapping With Neural Radiance Fields”. In: *IEEE Robotics and Automation Letters* 8 (2023), pp. 5950–5957. URL: https://api.semanticscholar.org/CorpusID:258079281.

[14] Yao He et al. “FoundLoc: Vision-based Onboard Aerial Localization in the Wild”. In: *ArXiv* abs/2310.16299 (2023). URL: https://api.semanticscholar.org/CorpusID:264487194.

[15] Thorsten Hermes et al. “Content-based image retrieval”. In: *Multimedia Tools and Applications* 82 (2023), pp. 37903–37903. URL: https://api.semanticscholar.org/CorpusID:225264.

[16] Anthony Hu et al. “GAIA-1: A Generative World Model for Autonomous Driving”. In: *ArXiv* abs/2309.17080 (2023). URL: https://api.semanticscholar.org/CorpusID:263310665.

[17] Huajian Huang et al. “360Loc: A Dataset and Benchmark for Omnidirectional Visual Localization with Cross-device Queries”. In: *ArXiv* abs/2311.17389 (2023). URL: https://api.semanticscholar.org/CorpusID:265498473.

[18] Nikhil Varma Keetha et al. “AnyLoc: Towards Universal Visual Place Recognition”. In: *IEEE Robotics and Automation Letters* 9 (2023), pp. 1286–1293. URL: https://api.semanticscholar.org/CorpusID:260351368.

[19] Nikhil Varma Keetha et al. “SplaTAM: Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM”. In: *ArXiv* abs/2312.02126 (2023). URL: https://api.semanticscholar.org/CorpusID:265609060.

[20] Bernhard Kerbl et al. “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Transactions on Graphics (TOG)* 42 (2023), pp. 1–14. URL: https://api.semanticscholar.org/CorpusID:259267917.

[21] Maxime Oquab et al. “DINOv2: Learning Robust Visual Features without Supervision”. In: *ArXiv* abs/2304.07193 (2023). URL: https://api.semanticscholar.org/CorpusID:258170077.

[22] Shihao Shao et al. “Global Features are All You Need for Image Retrieval and Reranking”. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)* (2023), pp. 11002–11012. URL: https://api.semanticscholar.org/CorpusID:260887397.

[23] Guanhua Wang et al. “ZeRO++: Extremely Efficient Collective Communication for Giant Model Training”. In: *ArXiv* abs/2306.10209 (2023). URL: https://api.semanticscholar.org/CorpusID:259203634.

[24] Tsun-Hsuan Wang et al. “Drive Anywhere: Generalizable End-to-end Autonomous Driving with Multi-modal Foundation Models”. In: *ArXiv* abs/2310.17642 (2023). URL: https://api.semanticscholar.org/CorpusID:264490392.

[25] Chi Yan et al. “GS-SLAM: Dense Visual SLAM with 3D Gaussian Splatting”. In: *ArXiv* abs/2311.11700 (2023). URL: https://api.semanticscholar.org/CorpusID:265294572.

[26] Zhengyuan Yang et al. “The Dawn of LMMs: Preliminary Explorations with GPT-4V(ision)”. In: *ArXiv* abs/2309.17421 (2023). URL: https://api.semanticscholar.org/CorpusID:263310951.

[27] Vladimir V. Yugay et al. “Gaussian-SLAM: Photo-realistic Dense SLAM with Gaussian Splatting”. In: *ArXiv* abs/2312.10070 (2023). URL: https://api.semanticscholar.org/CorpusID:266348788.

[28] Shibo Zhao et al. “SubT-MRS Dataset: Pushing SLAM Towards All-weather Environments”. In: 2023. URL: https://api.semanticscholar.org/CorpusID:259937795.

[29] Yanli Zhao et al. “PyTorch FSDP: Experiences on Scaling Fully Sharded Data Parallel”. In: *Proc. VLDB Endow.* 16 (2023), pp. 3848–3860. URL: https://api.semanticscholar.org/CorpusID:258297871.

[30] Zihan Zhu et al. “NICER-SLAM: Neural Implicit Scene Encoding for RGB SLAM”. In: *ArXiv* abs/2302.03594 (2023). URL: https://api.semanticscholar.org/CorpusID:256627303.

[31] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. “GSV-Cities: Toward Appropriate Supervised Visual Place Recognition”. In: *ArXiv* abs/2210.10239 (2022). URL: https://api.semanticscholar.org/CorpusID:252510544.

[32] Andréa Macario Barros et al. “A Comprehensive Survey of Visual SLAM Algorithms”. In: *Robotics* 11 (2022), p. 24. URL: https://api.semanticscholar.org/CorpusID:246794074.

[33] Gabriele Berton, Carlos German Masone, and Barbara Caputo. “Rethinking Visual Geo-localization for Large-Scale Applications”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 4868–4878. URL: https://api.semanticscholar.org/CorpusID:247958415.

[34] Gabriele Berton et al. “Deep Visual Geo-localization Benchmark”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 5386–5397. URL: https://api.semanticscholar.org/CorpusID:248006142.

[35] Tri Dao et al. “FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness”. In: *ArXiv* abs/2205.14135 (2022). URL: https://api.semanticscholar.org/CorpusID:249151871.

[36] M. Humenberger et al. “Investigating the Role of Image Retrieval for Visual Localization”. In: *International Journal of Computer Vision* 130 (2022), pp. 1811–1836. URL: https://api.semanticscholar.org/CorpusID:249081030.

[37] Ahmad Khaliq, Michael Milford, and Sourav Garg. “MultiRes-NetVLAD: Augmenting Place Recognition Training With Low-Resolution Imagery”. In: *IEEE Robotics and Automation Letters* 7 (2022), pp. 3882–3889. URL: https://api.semanticscholar.org/CorpusID:246540788.

[38] Benjamin Lefaudeux et al. *xFormers: A modular and hackable Transformer modelling library*. https://github.com/facebookresearch/xformers. 2022.

[39] Conglong Li et al. “DeepSpeed Data Efficiency: Improving Deep Learning Model Quality and Training Efficiency via Efficient Data Sampling and Routing”. In: *ArXiv* abs/2212.03597 (2022). URL: https://api.semanticscholar.org/CorpusID:254366532.

[40] Youwei Liang et al. “Not All Patches are What You Need: Expediting Vision Transformers via Token Reorganizations”. In: *ArXiv* abs/2202.07800 (2022). URL: https://api.semanticscholar.org/CorpusID:246867285.

[41] Zhuang Liu et al. “A ConvNet for the 2020s”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 11966–11976. URL: https://api.semanticscholar.org/CorpusID:245837420.

[42] Steve Macenski et al. “Robot Operating System 2: Design, architecture, and uses in the wild”. In: *Science Robotics* 7 (2022). URL: https://api.semanticscholar.org/CorpusID:248736816.

[43] Vojtech Panek, Zuzana Kukelova, and Torsten Sattler. “MeshLoc: Mesh-Based Visual Localization”. In: *European Conference on Computer Vision*. 2022. URL: https://api.semanticscholar.org/CorpusID:251018193.

[44] Ed Pizzi et al. “A Self-Supervised Descriptor for Image Copy Detection”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 14512–14522. URL: https://api.semanticscholar.org/CorpusID:247011159.

[45] Michael Schleiss, Fahmi Rouatbi, and Daniel Cremers. “VPAIR - Aerial Visual Place Recognition and Localization in Large-scale Outdoor Environments”. In: *ArXiv* abs/2205.11567 (2022). URL: https://api.semanticscholar.org/CorpusID:249017910.

[46] Christoph Schuhmann et al. “LAION-5B: An open large-scale dataset for training next generation image-text models”. In: *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2022. URL: https://openreview.net/forum?id=M3Y74vmsMcY.

[47] Teng Wang et al. “ViA: A Novel Vision-Transformer Accelerator Based on FPGA”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41 (2022), pp. 4088–4099. URL: https://api.semanticscholar.org/CorpusID:253123249.

[48] Yuxiao Zhou and Kecheng Yang. "Exploring TensorRT to Improve Real-Time Inference for Deep Learning". In: *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)* (2022), pp. 2011–2018. URL: https://api.semanticscholar.org/CorpusID:253882601.

[49] Shirzad Amir et al. "Deep ViT Features as Dense Visual Descriptors". In: *ArXiv* abs/2112.05814 (2021). URL: https://api.semanticscholar.org/CorpusID:245123845.

[50] Hangbo Bao, Li Dong, and Furu Wei. "BEiT: BERT Pre-Training of Image Transformers". In: *ArXiv* abs/2106.08254 (2021). URL: https://api.semanticscholar.org/CorpusID:235436185.

[51] Mathilde Caron et al. "Emerging Properties in Self-Supervised Vision Transformers". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 9630–9640. URL: https://api.semanticscholar.org/CorpusID:233444273.

[52] FairScale authors. *FairScale: A general purpose modular PyTorch library for high performance and large scale training*. https://github.com/facebookresearch/fairscale. 2021.

[53] Sourav Garg, Tobias Fischer, and Michael Milford. "Where is your place, Visual Place Recognition?" In: *ArXiv* abs/2103.06443 (2021). URL: https://api.semanticscholar.org/CorpusID:232185215.

[54] Sourav Garg and Michael Milford. "SeqNet: Learning Descriptors for Sequence-Based Hierarchical Place Recognition". In: *IEEE Robotics and Automation Letters* 6 (2021), pp. 4305–4312. URL: https://api.semanticscholar.org/CorpusID:232013538.

[55] Sourav Garg, Madhu Babu Vankadari, and Michael Milford. "SeqMatchNet: Contrastive Learning with Sequence Matching for Place Recognition & Relocalization". In: *Conference on Robot Learning*. 2021. URL: https://api.semanticscholar.org/CorpusID:237455214.

[56] A. Glover. *Day and night with lateral pose change datasets*. Zenodo, May 2021. DOI: 10.5281/zenodo.4745641. URL: https://doi.org/10.5281/zenodo.4745641.

[57] Ali Hassani et al. "Escaping the Big Data Paradigm with Compact Transformers". In: *ArXiv* abs/2104.05704 (2021). URL: https://api.semanticscholar.org/CorpusID:233210459.

[58] Kaiming He et al. "Masked Autoencoders Are Scalable Vision Learners". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 15979–15988. URL: https://api.semanticscholar.org/CorpusID:243985980.

[59] Gabriel Ilharco et al. *OpenCLIP*. Version 0.1. If you use this software, please cite it as below. July 2021. DOI: 10.5281/zenodo.5143773. URL: https://doi.org/10.5281/zenodo.5143773.

[60] Ze Liu et al. “Swin Transformer V2: Scaling Up Capacity and Resolution”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 11999–12009. URL: https://api.semanticscholar.org/CorpusID:244346076.

[61] Ze Liu et al. “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 9992–10002. URL: https://api.semanticscholar.org/CorpusID:232352874.

[62] Steve Macenski and Ivona Jambrecic. “SLAM Toolbox: SLAM for the dynamic world”. In: *J. Open Source Softw.* 6 (2021), p. 2783. URL: https://api.semanticscholar.org/CorpusID:235351236.

[63] Ferdinand Maiwald, Christoph U. Lehmann, and Taras Lazariv. “Fully Automated Pose Estimation of Historical Images in the Context of 4D Geographic Information Systems Utilizing Machine Learning Methods”. In: *ISPRS Int. J. Geo Inf.* 10 (2021), p. 748. URL: https://api.semanticscholar.org/CorpusID:243780743.

[64] Carlo Masone and Barbara Caputo. “A Survey on Deep Visual Place Recognition”. In: *IEEE Access* 9 (2021), pp. 19516–19547. URL: https://api.semanticscholar.org/CorpusID:231824266.

[65] Deepak Narayanan et al. “Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM”. In: *SC21: International Conference for High Performance Computing, Networking, Storage and Analysis* (2021), pp. 1–14. URL: https://api.semanticscholar.org/CorpusID:236635565.

[66] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *International Conference on Machine Learning*. 2021. URL: https://api.semanticscholar.org/CorpusID:231591445.

[67] Antoni Rosinol et al. “Kimera: From SLAM to spatial perception with 3D dynamic scene graphs”. In: *The International Journal of Robotics Research* 40 (2021), pp. 1510–1546. URL: https://api.semanticscholar.org/CorpusID:231632715.

[68] Edgar Sucar et al. “iMAP: Implicit Mapping and Positioning in Real-Time”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 6209–6218. URL: https://api.semanticscholar.org/CorpusID:232320654.

[69] Zachary Teed and Jia Deng. “DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras”. In: *Neural Information Processing Systems*. 2021. URL: https://api.semanticscholar.org/CorpusID:237278112.

[70] Yulun Tian et al. “Kimera-Multi: Robust, Distributed, Dense Metric-Semantic SLAM for Multi-Robot Systems”. In: *IEEE Transactions on Robotics* 38 (2021), pp. 2022–2038. URL: https://api.semanticscholar.org/CorpusID:235658543.

[71] Ilya O. Tolstikhin et al. “MLP-Mixer: An all-MLP Architecture for Vision”. In: *Neural Information Processing Systems*. 2021. URL: https://api.semanticscholar.org/CorpusID:233714958.

[72] Hugo Touvron et al. “Going deeper with Image Transformers”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 32–42. URL: https://api.semanticscholar.org/CorpusID:232428161.

[73] Xiaohua Zhai et al. “Scaling Vision Transformers”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 1204–1213. URL: https://api.semanticscholar.org/CorpusID:235367962.

[74] Jinghao Zhou et al. “iBOT: Image BERT Pre-Training with Online Tokenizer”. In: *ArXiv* abs/2111.07832 (2021). URL: https://api.semanticscholar.org/CorpusID:244117494.

[75] Zihan Zhu et al. “NICE-SLAM: Neural Implicit Scalable Encoding for SLAM”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 12776–12786. URL: https://api.semanticscholar.org/CorpusID:245385791.

[76] Carlos Campos et al. “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM”. In: *IEEE Transactions on Robotics* 37 (2020), pp. 1874–1890. URL: https://api.semanticscholar.org/CorpusID:220713377.

[77] Mathilde Caron et al. “Unsupervised Learning of Visual Features by Contrasting Cluster Assignments”. In: *ArXiv* abs/2006.09882 (2020). URL: https://api.semanticscholar.org/CorpusID:219721240.

[78] Ting Chen et al. “A Simple Framework for Contrastive Learning of Visual Representations”. In: *ArXiv* abs/2002.05709 (2020). URL: https://api.semanticscholar.org/CorpusID:211096730.

[79] Ting Chen et al. “Big Self-Supervised Models are Strong Semi-Supervised Learners”. In: *ArXiv* abs/2006.10029 (2020). URL: https://api.semanticscholar.org/CorpusID:219721239.

[80] Xinlei Chen et al. “Improved Baselines with Momentum Contrastive Learning”. In: *ArXiv* abs/2003.04297 (2020). URL: https://api.semanticscholar.org/CorpusID:212633993.

[81] Ekin D Cubuk et al. “Randaugment: Practical automated data augmentation with a reduced search space”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 702–703.

[82] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *ArXiv* abs/2010.11929 (2020). URL: https://api.semanticscholar.org/CorpusID:225039882.

[83] Juan Du, Rui Wang, and Daniel Cremers. “DH3D: Deep Hierarchical 3D Descriptors for Robust Large-Scale 6DoF Relocalization”. In: *ArXiv* abs/2007.09217 (2020). URL: https://api.semanticscholar.org/CorpusID:220647014.

[84] Sourav Garg et al. “Delta Descriptors: Change-Based Place Representation for Robust Visual Localization”. In: *IEEE Robotics and Automation Letters* 5 (2020), pp. 5120–5127. URL: https://api.semanticscholar.org/CorpusID:219558350.

[85] Jean-Bastien Grill et al. “Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning”. In: *ArXiv* abs/2006.07733 (2020). URL: https://api.semanticscholar.org/CorpusID:219687798.

[86] Ashish Jaiswal et al. “A Survey on Contrastive Self-supervised Learning”. In: *ArXiv* abs/2011.00362 (2020). URL: https://api.semanticscholar.org/CorpusID:226227230.

[87] Yuhe Jin et al. “Image Matching Across Wide Baselines: From Paper to Practice”. In: *International Journal of Computer Vision* 129 (2020), pp. 517–547. URL: https://api.semanticscholar.org/CorpusID:211817874.

[88] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *Communications of the ACM* 65 (2020), pp. 99–106. URL: https://api.semanticscholar.org/CorpusID:213175590.

[89] Noam M. Shazeer. “GLU Variants Improve Transformer”. In: *ArXiv* abs/2002.05202 (2020). URL: https://api.semanticscholar.org/CorpusID:211096588.

[90] Zachary Teed and Jia Deng. “RAFT: Recurrent All-Pairs Field Transforms for Optical Flow”. In: *European Conference on Computer Vision*. 2020. URL: https://api.semanticscholar.org/CorpusID:214667893.

[91] Hugo Touvron et al. “Training data-efficient image transformers & distillation through attention”. In: *International Conference on Machine Learning*. 2020. URL: https://api.semanticscholar.org/CorpusID:229363322.

[92] Ruibin Xiong et al. “On Layer Normalization in the Transformer Architecture”. In: *ArXiv* abs/2002.04745 (2020). URL: https://api.semanticscholar.org/CorpusID:211082816.

[93] Yuanzhong Xu et al. “Automatic Cross-Replica Sharding of Weight Update in Data-Parallel Training”. In: *ArXiv* abs/2004.13336 (2020). URL: https://api.semanticscholar.org/CorpusID:216562234.

[94] Mubariz Zaffar et al. “VPR-Bench: An Open-Source Visual Place Recognition Evaluation Framework with Quantifiable Viewpoint and Appearance Change”. In: *International Journal of Computer Vision* 129 (2020), pp. 2136–2174. URL: https://api.semanticscholar.org/CorpusID:218674095.

[95] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *North American Chapter of the Association for Computational Linguistics*. 2019. URL: https://api.semanticscholar.org/CorpusID:52967399.

[96] Mihai Dusmanu et al. “D2-Net: A Trainable CNN for Joint Description and Detection of Local Features”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 8084–8093. URL: https://api.semanticscholar.org/CorpusID:204241691.

[97] Kaiming He et al. “Momentum Contrast for Unsupervised Visual Representation Learning”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 9726–9735. URL: https://api.semanticscholar.org/CorpusID:207930212.

[98] Krishna Murthy Jatavallabhula, Ganesh Iyer, and Liam Paull. “∇SLAM: Dense SLAM meets Automatic Differentiation”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)* (2019), pp. 2130–2137. URL: https://api.semanticscholar.org/CorpusID:204837922.

[99] Axel Barroso Laguna et al. “Key.Net: Keypoint Detection by Handcrafted and Learned CNN Filters”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 5835–5843. URL: https://api.semanticscholar.org/CorpusID:90262949.

[100] B. K. Patle et al. “A review: On path planning strategies for navigation of mobile robot”. In: *Defence Technology* (2019). URL: https://api.semanticscholar.org/CorpusID:149886594.

[101] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: 2019. URL: https://api.semanticscholar.org/CorpusID:160025533.

[102] Samyam Rajbhandari et al. “ZeRO: Memory Optimization Towards Training A Trillion Parameter Models”. In: *ArXiv* abs/1910.02054 (2019). URL: https://api.semanticscholar.org/CorpusID:203736482.

[103] Jérôme Revaud et al. “R2D2: Repeatable and Reliable Detector and Descriptor”. In: *ArXiv* abs/1906.06195 (2019). URL: https://api.semanticscholar.org/CorpusID:189927786.

[104] Francisco Rubio, Francisco Valero, and Carlos Llopis-Albert. “A review of mobile robots: Concepts, methods, theoretical framework, and applications”. In: *International Journal of Advanced Robotic Systems* 16 (2019). URL: https://api.semanticscholar.org/CorpusID:146005566.

[105] Xun Wang et al. “Multi-Similarity Loss With General Pair Weighting for Deep Metric Learning”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 5017–5025. URL: https://api.semanticscholar.org/CorpusID:118646482.

[106] Ross Wightman. *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. 2019. DOI: 10.5281/zenodo.4414861.

[107] Baoyuan Wu et al. “Tencent ML-Images: A Large-Scale Multi-Label Image Database for Visual Representation Learning”. In: *IEEE Access* 7 (2019), pp. 172683–172693. URL: https://api.semanticscholar.org/CorpusID:57573693.

[108] Heng Yang et al. “Graduated Non-Convexity for Robust Spatial Perception: From Non-Minimal Solvers to Global Outlier Rejection”. In: *IEEE Robotics and Automation Letters* 5 (2019), pp. 1127–1134. URL: https://api.semanticscholar.org/CorpusID:202660784.

[109] Sourav Garg, Niko Sünderhauf, and Michael Milford. “LoST? Appearance-Invariant Place Recognition for Opposite Viewpoints using Visual Semantics”. In: *ArXiv* abs/1804.05526 (2018). URL: https://api.semanticscholar.org/CorpusID:4897390.

[110] Shing Yan Loo et al. “CNN-SVO: Improving the Mapping in Semi-Direct Visual Odometry Using Single-Image Depth Prediction”. In: *2019 International Conference on Robotics and Automation (ICRA)* (2018), pp. 5218–5223. URL: https://api.semanticscholar.org/CorpusID:52912195.

[111] Dhruv Kumar Mahajan et al. “Exploring the Limits of Weakly Supervised Pretraining”. In: *ArXiv* abs/1805.00932 (2018). URL: https://api.semanticscholar.org/CorpusID:13751202.

[112] Alexandre Sablayrolles et al. “Spreading vectors for similarity search”. In: *arXiv: Machine Learning* (2018). URL: https://api.semanticscholar.org/CorpusID:62841605.

[113] Paul-Edouard Sarlin et al. “From Coarse to Fine: Robust Hierarchical Localization at Large Scale”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 12708–12717. URL: https://api.semanticscholar.org/CorpusID:54460261.

[114] Torsten Sattler et al. “Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 8601–8610. URL: https://api.semanticscholar.org/CorpusID:4644084.

[115] H. Wang et al. “CosFace: Large Margin Cosine Loss for Deep Face Recognition”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 5265–5274. URL: https://api.semanticscholar.org/CorpusID:68589.

[116] Youjiang Xu et al. “Sequential Video VLAD: Training the Aggregation Locally and Temporally”. In: *IEEE Transactions on Image Processing* 27 (2018), pp. 4933–4944. URL: https://api.semanticscholar.org/CorpusID:49656316.

[117] P. S. Anish et al. “Mobile Robot Navigation and Obstacle Avoidance Techniques: A Review”. In: *IEEE International Conference on Robotics and Automation*. 2017. URL: https://api.semanticscholar.org/CorpusID:36907925.

[118] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “SuperPoint: Self-Supervised Interest Point Detection and Description”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2017), pp. 337–33712. URL: https://api.semanticscholar.org/CorpusID:4918026.

[119] Priya Goyal et al. “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour”. In: *ArXiv* abs/1706.02677 (2017). URL: https://api.semanticscholar.org/CorpusID:13905106.

[120] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization”. In: *International Conference on Learning Representations*. 2017. URL: https://api.semanticscholar.org/CorpusID:53592270.

[121] William P. Maddern et al. “1 year, 1000 km: The Oxford RobotCar dataset”. In: *The International Journal of Robotics Research* 36 (2017), pp. 15–3. URL: https://api.semanticscholar.org/CorpusID:22556995.

[122] Anastasiya Mishchuk et al. “Working hard to know your neighbor’s margins: Local descriptor learning loss”. In: *Neural Information Processing Systems*. 2017. URL: https://api.semanticscholar.org/CorpusID:22589592.

[123] Dmytro Mishkin, Filip Radenovic, and Jiri Matas. “Repeatability Is Not Enough: Learning Affine Regions via Discriminability”. In: *European Conference on Computer Vision*. 2017. URL: https://api.semanticscholar.org/CorpusID:50782240.

[124] Adnan Qayyum et al. “Medical image retrieval using deep convolutional neural network”. In: *ArXiv* abs/1703.08472 (2017). URL: https://api.semanticscholar.org/CorpusID:13995550.

[125] Filip Radenovic, Giorgos Tolias, and Ondřej Chum. “Fine-Tuning CNN Image Retrieval with No Human Annotation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41 (2017), pp. 1655–1668. URL: https://api.semanticscholar.org/CorpusID:6426453.

[126] Chen Sun et al. “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 843–852. URL: https://api.semanticscholar.org/CorpusID:6842201.

[127] Xun Sun et al. “A Dataset for Benchmarking Image-Based Localization”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 5641–5649. URL: https://api.semanticscholar.org/CorpusID:20531893.

[128] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. “Visual SLAM algorithms: a survey from 2010 to 2016”. In: *IPSJ Transactions on Computer Vision and Applications* 9 (2017), pp. 1–11. URL: https://api.semanticscholar.org/CorpusID:7981750.

[129] Keisuke Tateno et al. “CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 6565–6574. URL: https://api.semanticscholar.org/CorpusID:206596482.

[130] Yurun Tian, Bin Fan, and Fuchao Wu. “L2-Net: Deep Learning of Discriminative Patch Descriptor in Euclidean Space”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 6128–6136. URL: https://api.semanticscholar.org/CorpusID:206596352.

[131] Ashish Vaswani et al. “Attention is All you Need”. In: *Neural Information Processing Systems*. 2017. URL: https://api.semanticscholar.org/CorpusID:13756489.

[132] Jingwei Zhang et al. “Neural SLAM”. In: *ArXiv* abs/1706.09520 (2017). URL: https://api.semanticscholar.org/CorpusID:39039010.

[133] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. “Layer Normalization”. In: *ArXiv* abs/1607.06450 (2016). URL: https://api.semanticscholar.org/CorpusID:8236317.

[134] César Cadena et al. “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age”. In: *IEEE Transactions on Robotics* 32 (2016), pp. 1309–1332. URL: https://api.semanticscholar.org/CorpusID:2596787.

[135] Wolfgang Hess et al. “Real-time loop closure in 2D LIDAR SLAM”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)* (2016), pp. 1271–1278. URL: https://api.semanticscholar.org/CorpusID:11892586.

[136] Gao Huang et al. “Deep Networks with Stochastic Depth”. In: *European Conference on Computer Vision*. 2016. URL: https://api.semanticscholar.org/CorpusID:6773885.

[137] Stephanie M. Lowry et al. “Visual Place Recognition: A Survey”. In: *IEEE Transactions on Robotics* 32 (2016), pp. 1–19. URL: https://api.semanticscholar.org/CorpusID:253204.

[138] Raul Mur-Artal and Juan D. Tardós. “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras”. In: *IEEE Transactions on Robotics* 33 (2016), pp. 1255–1262. URL: https://api.semanticscholar.org/CorpusID:206775640.

[139] Hyeonwoo Noh et al. “Large-Scale Image Retrieval with Attentive Deep Local Features”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2016), pp. 3476–3485. URL: https://api.semanticscholar.org/CorpusID:18376539.

[140] Raghavender Sahdev and John K. Tsotsos. “Indoor Place Recognition System for Localization of Mobile Robots”. In: *2016 13th Conference on Computer and Robot Vision (CRV)* (2016), pp. 53–60. URL: https://api.semanticscholar.org/CorpusID:10225397.

[141] Johannes L. Schönberger and Jan-Michael Frahm. “Structure-from-Motion Revisited”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 4104–4113. URL: https://api.semanticscholar.org/CorpusID:1728538.

[142] Johannes L. Schönberger et al. “Pixelwise View Selection for Unstructured Multi-View Stereo”. In: *European Conference on Computer Vision*. 2016. URL: https://api.semanticscholar.org/CorpusID:977535.

[143] Kwang Moo Yi et al. “LIFT: Learned Invariant Feature Transform”. In: *European Conference on Computer Vision*. 2016. URL: https://api.semanticscholar.org/CorpusID:602850.

[144] Relja Arandjelović et al. “NetVLAD: CNN Architecture for Weakly Supervised Place Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 5297–5307. URL: https://api.semanticscholar.org/CorpusID:44604205.

[145] Artem Babenko and Victor S. Lempitsky. “Aggregating Local Deep Features for Image Retrieval”. In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1269–1277. URL: https://api.semanticscholar.org/CorpusID:5225966.

[146] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 770–778. URL: https://api.semanticscholar.org/CorpusID:206594692.

[147] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. “Distilling the Knowledge in a Neural Network”. In: *ArXiv* abs/1503.02531 (2015). URL: https://api.semanticscholar.org/CorpusID:7200347.

[148] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *ArXiv* abs/1502.03167 (2015). URL: https://api.semanticscholar.org/CorpusID:5808102.

[149] Raul Mur-Artal, José M. M. Montiel, and Juan D. Tardós. “ORB-SLAM: A Versatile and Accurate Monocular SLAM System”. In: *IEEE Transactions on Robotics* 31 (2015), pp. 1147–1163. URL: https://api.semanticscholar.org/CorpusID:206775100.

[150] Niko Sünderhauf et al. “On the performance of ConvNet features for place recognition”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2015), pp. 4297–4304. URL: https://api.semanticscholar.org/CorpusID:9202348.

[151] Bart Thomee et al. “YFCC100M: The new data in multimedia research”. In: *Communications of the ACM* 59 (2015), pp. 64–73. URL: https://api.semanticscholar.org/CorpusID:207230134.

[152] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. “Particular object retrieval with integral max-pooling of CNN activations”. In: *CoRR* abs/1511.05879 (2015). URL: https://api.semanticscholar.org/CorpusID:786898.

[153] Jakob J. Engel, Thomas Schöps, and Daniel Cremers. “LSD-SLAM: Large-Scale Direct Monocular SLAM”. In: *European Conference on Computer Vision*. 2014. URL: https://api.semanticscholar.org/CorpusID:14547347.

[154] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. “SVO: Fast semi-direct monocular visual odometry”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), pp. 15–22. URL: https://api.semanticscholar.org/CorpusID:206850490.

[155] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural Turing Machines”. In: *ArXiv* abs/1410.5401 (2014). URL: https://api.semanticscholar.org/CorpusID:15299054.

[156] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR* abs/1412.6980 (2014). URL: https://api.semanticscholar.org/CorpusID:6628106.

[157] Ali Sharif Razavian et al. “Visual Instance Retrieval with Deep Convolutional Networks”. In: *CoRR* abs/1412.6574 (2014). URL: https://api.semanticscholar.org/CorpusID:14082107.

[158] Pablo Fernández Alcantarilla, Jesús Nuevo, and Adrien Bartoli. “Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces”. In: *British Machine Vision Conference*. 2013. URL: https://api.semanticscholar.org/CorpusID:8488231.

[159] Relja Arandjelović and Andrew Zisserman. “All About VLAD”. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 1578–1585. URL: https://api.semanticscholar.org/CorpusID:9496819.

[160] Maik Keller et al. “Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion”. In: *2013 International Conference on 3D Vision* (2013), pp. 1–8. URL: https://api.semanticscholar.org/CorpusID:17260589.

[161] Akihiko Torii et al. “Visual Place Recognition with Repetitive Structures”. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 883–890. URL: https://api.semanticscholar.org/CorpusID:556419.

[162] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. “KAZE Features”. In: *European Conference on Computer Vision*. 2012. URL: https://api.semanticscholar.org/CorpusID:5720491.

[163] Relja Arandjelović and Andrew Zisserman. “Three things everyone should know to improve object retrieval”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 2911–2918. URL: https://api.semanticscholar.org/CorpusID:14678946.

[164] Dorian Gálvez-López and Juan D. Tardós. “Bags of Binary Words for Fast Place Recognition in Image Sequences”. In: *IEEE Transactions on Robotics* 28 (2012), pp. 1188–1197. URL: https://api.semanticscholar.org/CorpusID:710586.

[165] Michael Milford and Gordon F. Wyeth. “SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights”. In: *2012 IEEE International Conference on Robotics and Automation* (2012), pp. 1643–1649. URL: https://api.semanticscholar.org/CorpusID:14700600.

[166] Jürgen Sturm et al. “A benchmark for the evaluation of RGB-D SLAM systems”. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), pp. 573–580. URL: https://api.semanticscholar.org/CorpusID:206942855.

[167] Mark Joseph Cummins and Paul Newman. “Appearance-only SLAM at large scale with FAB-MAP 2.0”. In: *The International Journal of Robotics Research* 30 (2011), pp. 1100–1123. URL: https://api.semanticscholar.org/CorpusID:18537011.

[168] Stefan Kohlbrecher et al. “A flexible and scalable SLAM system with full 3D motion estimation”. In: *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics* (2011), pp. 155–160. URL: https://api.semanticscholar.org/CorpusID:14113246.

[169] Richard A. Newcombe, S. Lovegrove, and Andrew J. Davison. “DTAM: Dense tracking and mapping in real-time”. In: *2011 International Conference on Computer Vision* (2011), pp. 2320–2327. URL: https://api.semanticscholar.org/CorpusID:1336659.

[170] Richard A. Newcombe et al. “KinectFusion: Real-time dense surface mapping and tracking”. In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality* (2011), pp. 127–136. URL: https://api.semanticscholar.org/CorpusID:11830123.

[171] Ethan Rublee et al. “ORB: An efficient alternative to SIFT or SURF”. In: *2011 International Conference on Computer Vision* (2011), pp. 2564–2571. URL: https://api.semanticscholar.org/CorpusID:206769866.

[172] Torsten Sattler, B. Leibe, and Leif Kobbelt. “Fast image-based localization using direct 2D-to-3D matching”. In: *2011 International Conference on Computer Vision* (2011), pp. 667–674. URL: https://api.semanticscholar.org/CorpusID:11836057.

[173] Michael Calonder et al. “BRIEF: Binary Robust Independent Elementary Features”. In: *European Conference on Computer Vision*. 2010. URL: https://api.semanticscholar.org/CorpusID:1815530.

[174] Giorgio Grisetti et al. “A Tutorial on Graph-Based SLAM”. In: *IEEE Intelligent Transportation Systems Magazine* 2 (2010), pp. 31–43. URL: https://api.semanticscholar.org/CorpusID:5970112.

[175] Hervé Jégou et al. “Aggregating local descriptors into a compact image representation”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), pp. 3304–3311. URL: https://api.semanticscholar.org/CorpusID:1912782.

[176] Kurt Konolige et al. “Efficient Sparse Pose Adjustment for 2D mapping”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2010), pp. 22–29. URL: https://api.semanticscholar.org/CorpusID:8132160.

[177] Michael Warren et al. “Unaided stereo vision based pose estimation”. In: *IEEE International Conference on Robotics and Automation*. 2010. URL: https://api.semanticscholar.org/CorpusID:7349245.

[178] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 248–255. URL: https://api.semanticscholar.org/CorpusID:57246310.

[179] Morgan Quigley. “ROS: an open-source Robot Operating System”. In: *IEEE International Conference on Robotics and Automation*. 2009. URL: https://api.semanticscholar.org/CorpusID:6324125.

[180] Mark Joseph Cummins and Paul Newman. “FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance”. In: *The International Journal of Robotics Research* 27 (2008), pp. 647–665. URL: https://api.semanticscholar.org/CorpusID:17969052.

[181] Edward Rosten, Reid B. Porter, and Tom Drummond. "Faster and Better: A Machine Learning Approach to Corner Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (2008), pp. 105–119. URL: https://api.semanticscholar.org/CorpusID:206764370.

[182] Howie Choset et al. "Principles of Robot Motion: Theory, Algorithms, and Implementation". In: 2007. URL: https://api.semanticscholar.org/CorpusID:61848621.

[183] Andrew J. Davison et al. "MonoSLAM: Real-Time Single Camera SLAM". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (2007), pp. 1052–1067. URL: https://api.semanticscholar.org/CorpusID:206764185.

[184] Giorgio Grisetti, C. Stachniss, and Wolfram Burgard. "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters". In: *IEEE Transactions on Robotics* 23 (2007), pp. 34–46. URL: https://api.semanticscholar.org/CorpusID:321256.

[185] Georg S. W. Klein and David William Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality* (2007), pp. 225–234. URL: https://api.semanticscholar.org/CorpusID:206986664.

[186] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *European Conference on Computer Vision*. 2006. URL: https://api.semanticscholar.org/CorpusID:461853.

[187] David Nistér and Henrik Stewénius. "Scalable Recognition with a Vocabulary Tree". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* 2 (2006), pp. 2161–2168. URL: https://api.semanticscholar.org/CorpusID:1654266.

[188] Edward Rosten and Tom Drummond. "Machine Learning for High-Speed Corner Detection". In: *European Conference on Computer Vision*. 2006. URL: https://api.semanticscholar.org/CorpusID:1388140.

[189] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. "Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)". In: 2005. URL: https://api.semanticscholar.org/CorpusID:59864784.

[190] Thomas Martin Lehmann et al. "Content-based Image Retrieval in Medical Applications". In: *Methods of Information in Medicine* 43 (2004), pp. 354–361. URL: https://api.semanticscholar.org/CorpusID:15584417.

[191] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60 (2004), pp. 91–110. URL: https://api.semanticscholar.org/CorpusID:174065.

[192] Josef Sivic and Andrew Zisserman. "Video Google: a text retrieval approach to object matching in videos". In: *Proceedings Ninth IEEE International Conference on Computer Vision* (2003), 1470–1477 vol.2. URL: https://api.semanticscholar.org/CorpusID:14457153.

[193] Richard Hartley and Andrew Zisserman. “Multiple View Geometry in Computer Vision”. In: 2001. URL: https://api.semanticscholar.org/CorpusID:264666721.

[194] Roy Featherstone and David E. Orin. “Robot dynamics: equations and algorithms”. In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)* 1 (2000), 826–834 vol.1. URL: https://api.semanticscholar.org/CorpusID:499047.

[195] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proc. IEEE* 86 (1998), pp. 2278–2324. URL: https://api.semanticscholar.org/CorpusID:14542261.