IEEE Robotics and Automation Letters (RAL) paper presented at
2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Design Space Exploration of Low-Bit Quantized Neural Networks for Visual Place Recognition

Oliver Grainge[1], Michael Milford[2], Indu Bodala[1], Sarvapali D. Ramchurn[1] and Shoaib Ehsan[1,3]

***Abstract*— Visual Place Recognition (VPR) is a critical task for performing global re-localization in visual perception systems, requiring the ability to recognize a previously visited location under variations such as illumination, occlusion, appearance and viewpoint. In the case of robotics, the target devices for deployment are usually embedded systems. Therefore whilst the accuracy of VPR systems is important so too is memory consumption and latency. Recently new works have focused on the Recall@1 metric paying limited attention to resource utilization, resulting in methods that use complex models unsuitable for edge deployment. We hypothesize that these methods can be optimized to satisfy the constraints of low powered embedded systems whilst maintaining high recall performance. Our work studies the impact of compact architectural design in combination with full-precision and mixed-precision post-training quantization on VPR performance. Importantly we not only measure performance via the Recall@1 score but also measure memory consumption and latency. We characterize the design implications on memory, latency and recall scores and provide a number of design recommendations for VPR systems under these limitations.**

Visual-Place-Recognition, Quantization, Localization

I. Introduction

Visual Place Recognition (VPR) is the task of recognising a previously visited place. It forms a critical part of the global re-localization module of visual simultaneous localization and mapping systems (VSLAM) [1], [2]. VPR performs loop closures which reduce the uncertainty in perceptions [3], improving the robustness of downstream tasks. To achieve these benefits a VPR system must be able to recognise when two images are taken from the same place. This seemingly simple task is complicated by natural variations in images of the same place taken at different times, including changes in illumination, occlusion, viewpoint, and appearance.

Manuscript received: January, 29, 2024; Revised Febuary, 15, 2024; Accepted March, 25, 2024.

This paper was recommended for publication by Editor Pascal Vasseur upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the UK Engineering and Physical Sciences Research Council through grants EP/Y009800/1 and EP/V00784X/1

[1]O. Grainge, I. Bodala. S. D. Ramchurn and S. Ehsan are with the school of Electronics and Computer Science, University of Southampton, United Kingdon (email: oeg1n18@soton.ac.uk; i.p.bodala@soton.ac.uk; sdr1@soton.ac.uk; s.ehsan@soton.ac.uk).

[2]M. Milford is with the School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia (email: michael.milford@qut.edu.au).

[3]S. Ehsan is also with the school of Computer Science and Electronic Engineering, University of Essex, United Kingdom, (email: sehsan@essex.ac.uk).

Digital Object Identifier (DOI): see top of this page.

Copyright ©2024 IEEE

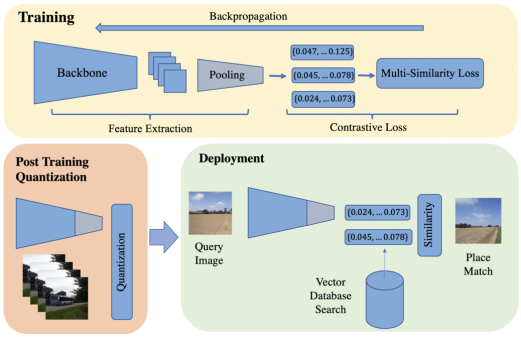

Fig. 1. Block Diagram of our Visual Place Recognition Training, Quantization and Deployment Pipeline

VPR is a representation problem. To achieve high performance a method must extract low dimensional features that are close in vector space for images of the same place whilst being distant for images of different places. A process depicted by the deployment section in Fig I. Previously this has been achieved by lightweight handcrafted feature extraction techniques such as those utilizing keypoint descriptors [4], [**?**], [5] or whole image holistic descriptors [6], [7]. Handcrafted features, whilst highly efficient do not perform robustly under appearance or viewpoint changes [8]. This has however been achieved through the design of specialized pooling layers in combination with deep neural networks [9], [10], [11], [12], [13]. Whilst the network backbone as shown in Fig I provides the capacity to learn representations, the pooling layers shown in grey act on features with different principles to obtain features with specific invariance properties [10], [12], [11].

Recent studies have demonstrated that enhancing the scale of both the network backbone and the training dataset substantially improves recall scores [14], [15], [16]. However, this scaling also leads to increased latency and memory usage in VPR. The latest state-of-the-art VSLAM systems, such as ORB-SLAM3 [1] and Kimera [2], process frames at up to 40Hz, posing a real-time bottleneck on the VPR inference, particularly if they are to be combined with sequential consistency checks. Our experiments demonstrate that on an embedded platform, the feature extraction latency for even the most efficient state-of-the-art VPR techniques, such as CosPlace [17], EigenPlaces [15], and MixVPR [13], can exceed 29ms. They hence create a bottleneck in Visual SLAM systems reducing real-time performance. Additionally the recent increases in model size [16] limits the range of

devices on which these systems can be deployed as they must have suitable memory capacity. For instance, AnyLoc [16], based on our experiments, requires 4.3GB just to store the model, rendering it unusable for lightweight devices such as drones and small ground robots.

Embedded systems, the primary target devices for VPR have limited memory, throughput, and power. However, with the proliferation of vision and machine learning applications, there is a growing trend toward enabling these systems to handle parallel computations, specifically with integer arithmetic, as evidenced by circuits such as systolic array compute units [18]. Such advancements are driven by the benefits of reduced latency, energy, and memory consumption [19]. They are realized through the reduced complexity of integer Multiply Add Accumulates (MACs) and a reduction in the required memory bandwidth for reading and writing layer weights and activations [20]. In essence, quantized neural networks reduce latency and memory consumption significantly and hence are a focus of our work for embedded VPR.

While previous studies have explored the design of VPR systems with a focus on recall performance [21], [8] there has been little coverage regarding the design of quantized VPR networks with a focus on efficiency. To address this gap in the literature our work thoroughly examines the design of low-bit-width VPR networks in four dimensions: network backbone, pooling method, quantization scheme, and descriptor size. We explore the interdependencies between these design points and hypothesize that by combining optimizations from each of these aspects, much of the recall performance of larger, more powerful networks can be preserved while making the computation lightweight enough to be usable in real-time embedded perception systems. Our quantization exploration includes both single and mixed-precision methods, with the latter's precision levels found through a genetic search. As such, our work identifies the specific design configurations required to maximize VPR recall performance under embedded resource constraints, thus providing significant value to the VPR research and robotics communities.

II. Related Work

Forming image descriptors robust to appearance and viewpoint changes is an ongoing challenge for the VPR community. Early attempts used handcrafted features to identify key points. The Scale Invariant Feature Transform (SIFT) [22] identifies key points using the Difference Of Gaussians blob detector and describes them using a histogram of oriented gradients (HOG). SIFT descriptions are speeded up in SURF [23] by utilizing integral images. Each of these keypoint feature description methods can be aggregated with a Bag Of Visual Words (BOVW) or Vector of Locally Aggregated Descriptors (VLAD) technique [24], [25]. Whilst effective, handcrafted features have struggled with viewpoint and appearance changes leading to the widespread use of CNNs [8]. Excellent robustness to appearance change can be achieved with NetVLAD [11] an end-to-end trainable differentiable relaxation of the VLAD descriptor. SPoC [9] a simpler pooling method utilises spatial mean pooling and MAC [10] spatial max pooling. GeM [12] is a parameterized generalization between average and max pooling that can learn to focus its activation distribution on salient image regions. Recently, MixVPR [13] achieved state-of-the-art performance by employing an MLP-Mixer architecture. However, it has been surpassed by the advanced representations of AnyLoc [16], which utilizes a DINOv2 transformer backbone [26] trained in a self-supervised manner on billions of images.

Previous VPR research, as outlined in [21], has primarily concentrated on recall performance on desktop hardware, neglecting memory and latency implications critical for embedded deployment. The exploration of compact models, especially those incorporating grouped convolutions and inverted bottlenecks, is lacking in VPR literature. Similarly, the exploration of quantization is limited. Binary neural networks offer significant memory and latency savings [27] but are unable to capture the complex representations required for large-scale VPR [28]. Although significant advancements have been made in integer quantization for classification and other vision tasks [29], its application in VPR is still largely uninvestigated.

III. Experimental Setup

A. Datasets

We train our models using the GSV-Cities Dataset [14], a diverse collection of over 570k images depicting 67k different places. For testing, the Pittsburgh30k (Pitts30k), Mapillary Street Level Sequences (MSLS) and Nordland [11], [30], [31] datasets are utilized as each assesses a different aspect of the VPR system. The Nordland dataset offers seasonal appearance changes, Pitts30k displays extreme viewpoint shifts, and MSLS contains a large range of urban conditions. Together they validate VPR methods against a diverse set of challenges.

B. Training

We train our models following the protocol outlined in [13], as it is shown to be the most effective [14]. This protocol uses the multi-similarity loss function with online mining. During training, we set the image resolutions to 320x320, which, although smaller than the conventional 480x640, enhances efficiency without sacrificing recall performance [21]. For gradient descent, we utilize the Adam optimizer, starting with an initial learning rate of 1e-3 and a learning rate step decay of 0.3, which is applied at the 5th, 10th, and 15th epochs.

C. Backbones

In this work, we investigate the effect of the backbone architecture on the embedded system performance for VPR. To this end, we test 8 different backbones including the traditional VGG-16 and ResNet models [32], [33]. VGG-16

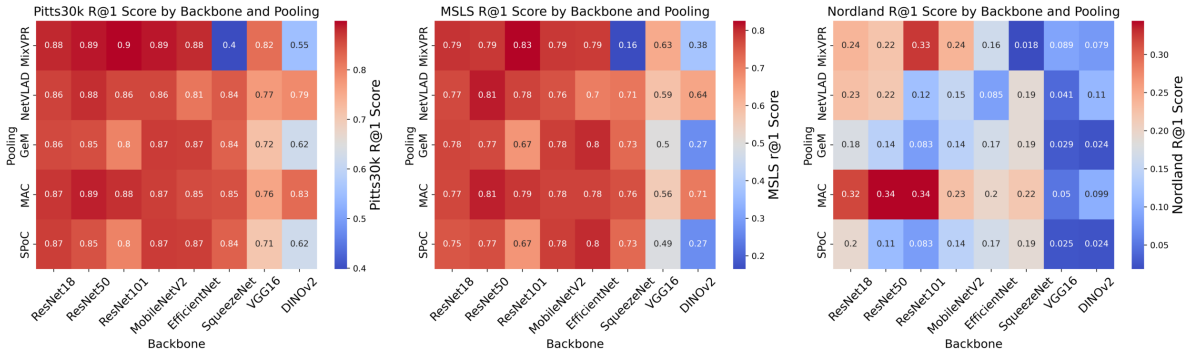

Fig. 2. Network Recall@1 performance for different datasets, backbones, and pooling methods under fp16 quantization. As can be seen from the rows of the figure, the pooling method is most important for Recall@1 performance.

is a traditional network using large kernel sizes without any batch normalization. ResNets conversely are deeper networks with smaller kernels, employing residual connections and batch normalization to facilitate gradient propagation and enable deeper architecture design.

We additionally investigate the use of networks optimized for mobile inference including MobileNetV2, SqueezeNet and EfficientNet B0 [34], [35], [36]. All three use factorized convolutions to increase parameter efficiency and inverted bottlenecks to maintain expression capacity. Inverted Bottlenecks expand the channel dimensions with 1x1 convolutions and subsequently manipulate information with efficient depth-wise convolution. MobileNetV2 [34] utilizes the inverted bottle-necks with a final linear projection to preserve information. SqueezeNet [35] uses Fire Modules which make use of both 1x1 and 3x3 depth-wise convolutions to capture more spatial features. EfficientNet [36], a model found by forming scaling laws for depth, resolution and width also uses a squeeze and excite mechanism to weight channels by their relative importance.Due to the success of AnyLoc [16], we have additionally incorporated the DINOv2 [26] backbone into our experiments. To enhance efficiency and allow for an effective comparison among efficient CNNs, we employ the most compact version of the distilled DINOv2 which uses a patch size of 14 with 12 transformer blocks and 6 attention heads each. The inclusion of this backbone enables a comprehensive analysis of CNNs versus more general transformer representations.

D. Pooling

The five pooling methods we investigated are SPoC, MAC, GeM, NetVLAD and MixVPR[9], [10], [12], [11], [13]. NetVLAD given a set of d dimensional descriptors $X = \{x_0, x_1, \cdots, x_N\}$ and K learnt codes $C = \{c_0, c_1, \cdots, c_K\}$ computes the weighted sum of residuals between the closest codes and descriptors as shown in Equation 1. The weights a_{nk} are computed via a differentiable relaxation of the hard cluster assignment given by Equation 2.

$$v_k = \sum_{n=1}^{N} a_{nk}(x_n - c_k) \tag{1}$$

$$a_{nk} = \frac{\exp(\langle x_n, c_k \rangle)}{\sum_{j=1}^{K} \exp(\langle x_n, c_j \rangle)} \tag{2}$$

SPoC [9] is a spatial weighted average pooling. Given a convolutional feature map of dimensions DxHxW. The spatial dimensions HxW are averaged producing a D dimensional descriptor. In this work, we adopt uniform weighting making SPoC akin to global average pooling. The MAC [10] descriptor is very similar and implemented as global max pooling. We also investigate GeM which is a learnable generalization between SPoC and MAC [9], [12]. As shown in Equation 3 given D feature maps F with spatial dimensions HxW the generalization is performed with the learnable parameter p. The MixVPR [13] pooling method applies an MLP-Mixer on the D dimensional descriptors to incorporate both channel-wise and row-wise information into the final representation.

$$v_d = \left(\frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} F_{ijd}^{p} \right)^{\frac{1}{p}} \tag{3}$$

Considering that all these pooling methods convert the spatial feature maps of CNNs into D, HxW dimensional vectors, each mapping to a specific receptive field in the input images, they can be similarly applied to the output tokens of a transformer, which also map to patches in the input image. Utilizing this principle, we apply all pooling methods to the token outputs of the DINOv2 backbone.

E. Quantization

As shown in Fig I we perform post-training quantization (PTQ) using a data sample to reduce memory, energy consumption and minimize latency. PTQ to floating point 16 (fp16) requires only a type cast. As fp16 has a 5-bit exponent and 10-bit mantissa as opposed to fp32's 8-bit exponent and 23-bit mantissa it has a significantly smaller dynamic range and numeric precision.

Quantizing the network for integer inference is more involved as continuous values can no longer be represented.

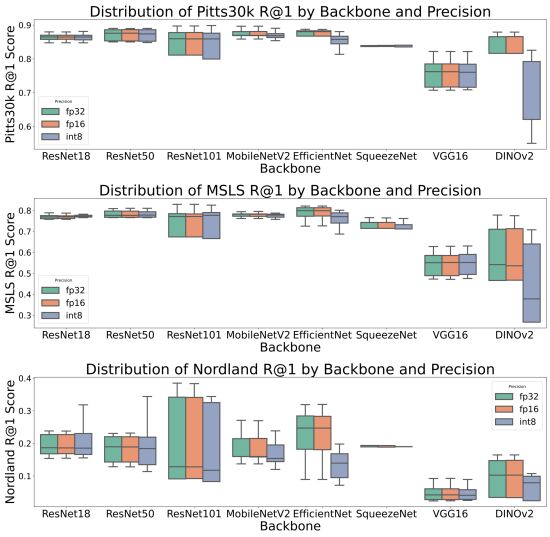

Fig. 3. Distribution of backbone Recall@1 performance under full-network quantization. The figure shows no loss in performance under fp16 precision.

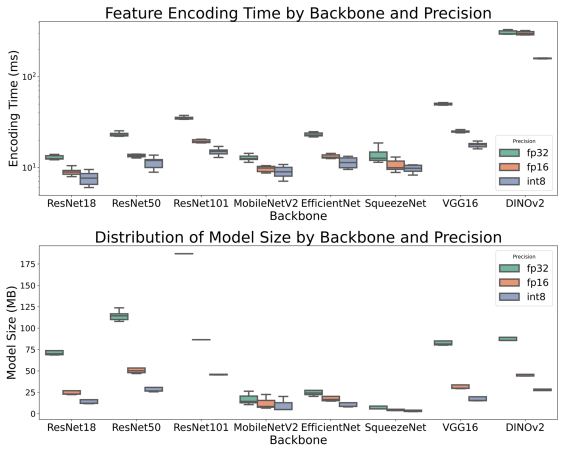

Fig. 4. Distribution of latency and memory cost for backbone architectures across different pooling methods. As seen in the top graph, DINOv2 has significantly higher latency than the convolution models

In this work, we quantize weights to a signed 8-bit or 4-bit integer data type for weights and accumulate in int32 for activations. We utilize linear symmetric quantization over linear asymmetric as it removes an additional zero-point bias and improves implementation efficiency in hardware. The quantization is performed by Equation 4 with b the integer bit width and s the floating point scaler. Dequantization is performed by Equation 5. For convolution weights, we use per-channel granularity with a scaler s for every channel. For activations, per-tensor granularity is used with just a single scale parameter. Per-channel granularity for weights has been chosen as the factorized convolutions used by MobileNetV2, SqueezeNet and EfficientNet [34], [35], [36] have significantly varying weight distributions across channels [37]. To reduce the information loss under quantization we use entropy calibration by selecting a scale s that minimizes the Kullbeck-Leibler divergence between the quantized and unquantized weights.

$$w_{int} = \text{clamp}(\lfloor \frac{w_{float}}{s} \rceil, 0, 2^{b-1}) \tag{4}$$

$$w_{float} = s \circ w_{int} \tag{5}$$

IV. Architectural Performance

In this section, we evaluate the embedded performance impact of compact architectural design. For each design, we evaluate the Recall@1 accuracy in addition to the memory and latency cost. All evaluations are performed with a compact 1024 dimension descriptor size. For hardware we use the Nvidia Xavier Nx embedded system with the TensorRT back-end for inference.

A. Backbone

Fig 3 displays backbone recall performance across datasets, pooling layers, and quantization precisions. Notably, quantizing to fp16 preserves recall performance for all architectures, while halving memory and significantly reducing latency (Fig 4). Quantization to int8 further reduces memory consumption, reducing the footprint of ResNet50 by 75%. However, the information loss caused by PTQ often causes small drops in Recall@1. The most robust backbones under int8 precision are the traditional architectures including the ResNets and VGG-16 due to their increased information redundancy.

Among all the models, MobileNetV2 demonstrates the highest recall rates across all evaluation datasets and as shown by Fig 4 exhibits a reduced latency when compared to the conventional ResNet50. Closely following in terms of recall performance is SqueezeNet, which display further reductions in latency and memory usage. Interestingly, both MobileNetV2 and SqueezeNet exhibit enhanced robustness under integer quantization when compared to EfficientNet. This is attributable to the sensitivity of EfficientNet's squeeze-and-excite mechanism to quantization noise. Overall, these results suggest that factorized convolutions, as utilized by efficient CNN designs, are indeed suitable for VPR representations.

In the case of DINOv2, although there is an improvement in recall over the VGG-16 architecture, its recall scores are not competitive with the other CNNs. Moreover, it significantly suffers from quantization noise and has a latency exceeding 300ms, which is substantially higher than the 23ms latency of EfficientNet rendering real-time VPR on the Xavier Nx infeasible.

B. Pooling Method

As demonstrated by Fig 5, MixVPR achieves the highest recall scores on the Pitts30k and MSLS datasets. These datasets, both originating from urban environments, closely match the distribution of the training data, indicating that

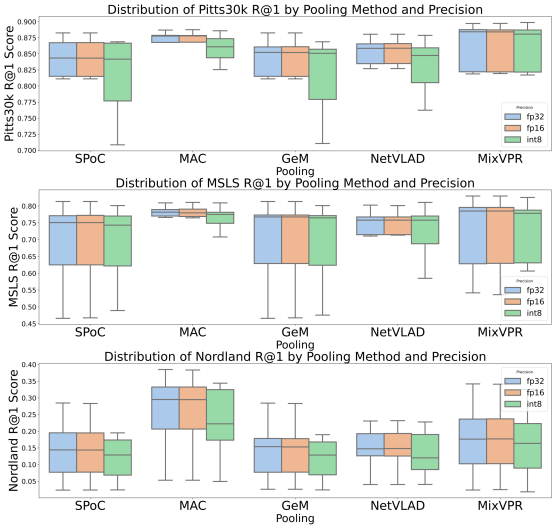

Fig. 5. Distribution of pooling layer Recall@1 performance under full network quantization to different precisions. MixVPR shows the highest recall on urban environments and the most robustness under quantization.

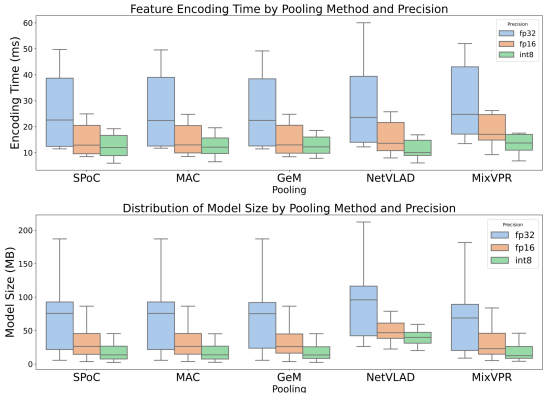

Fig. 6. Latency and memory usage across various backbones with consistent pooling methods at different precision levels. SPOC, MAC and GeM pooling show reduced latency and memory consumption over NetVLAD and MixVPR

MixVPR may be overfitting to these types of environments. However, it proves to be the most robust against quantization noise, thanks to the redundancy in its dense linear layers. In contrast, MAC and SPoC are the least robust to quantization, exhibiting significant recall decreases on the Pitts30k dataset. This is likely due to their nature as parameterless pooling methods, where the convolution weights previously used to generate the representations have less redundancy. Despite this, in full precision, MAC demonstrates the smallest variance in recall performance across different backbones. On the other hand, MixVPR shows higher sensitivity to the choice of backbone compared to MAC, indicated by a greater variance in its recall performance.

Regarding efficiency, Fig 6 illustrates that SPoC, MAC, and GeM incur the lowest memory and latency costs, attributable to the simplicity of these methods. Conversely, MixVPR exhibits increases in both latency and memory usage, a consequence of the multiple dense matrix multiplications required by its MLP heads. NetVLAD also displays slight increases in latency and memory usage compared to SPoC, MAC, and GeM due to the summation across its soft-assigned cluster residuals.

V. Descriptor Size

In Sections IV-A, IV-B, and Fig 2, we observed that MobileNetV2 and EfficientNet backbones, combined with MAC, MixVPR, or NetVLAD pooling, provide the optimal balance between memory usage, latency, and recall performance. Consequently we examine this subset of designs with varying descriptor sizes to find the optimal trade-off between recall accuracy and resource efficiency. To compare with the state-of-the-art, we include baseline CNN architectures as well as the DINOv2 encoder.

Table I presents the performance of the VPR system on the Pitts30K dataset across four descriptor sizes, alongside the feature extraction latency τ_e measured in milliseconds per image. It also includes the total VPR latency τ_{total} for each descriptor size, which is the sum of both τ_e and the retrieval latency τ_r, where τ_r denotes the time taken to match a place in a database of 100k descriptors. The table shows that increasing the descriptor dimension from 512 to 4096 on average improves recall scores on the Pitts30k dataset by 1%. This however comes at a significant latency cost. For the convolutional networks, this increase from 512 to 4096 increases the total VPR latency τ_{total} on average by 49%. However, for DINOv2, this increase is just 5% due to its significant extraction time. Conversely, for an efficient network such as MobileNetV2, the increase is 104%, highlighting the substantial impact that descriptor dimension has on VPR latency τ_{total}.

Among all pooling methods, MAC experienced the most significant improvement in recall with an increase in descriptor size, showing an average increase of 2.3% when the descriptor size was increased from 512 to 4096. However, this enhancement in performance not only adds to the latency but also incurs a significant memory cost. Increasing the descriptors from 512 to 4096 dimensions leads to a linear increase in memory consumption by a factor of 8x. For example, a map containing 100k descriptors of 512 dimensions requires merely 196MB, whereas for 4096 dimensions, the memory requirement escalates to 1.5GB.

VI. Mixed Precision

While integer post-training quantization significantly reduces memory consumption by an average of 69% and latency by 49%, it also leads to a decrease in recall performance by an average of 7%. With that in mind, a balance can be struck between preserving recall performance and reducing precision on a per-layer basis to improve efficiency. To achieve this a search can be conducted to identify the optimal precision for each layer that maximizes recall, subject to a budget for the average bit-width, denoted by B. Here, B is defined as the scalar value that represents the average number of bits allocated to represent the weights

TABLE I

IMPACT OF DESCRIPTOR SIZE ON RECALL PERFORMANCE AND LATENCY. ALL RESULTS ARE COMPUTED AT FLOATING POINT 16 PRECISION.

Backbone+Pooling	**Pitts30k R@1 at Descriptor Size**				**Latency (ms)**				
	512	**1024**	**2048**	**4096**	τ_e	$\tau_{\text{total}}^{512}$	$\tau_{\text{total}}^{1024}$	$\tau_{\text{total}}^{2048}$	$\tau_{\text{total}}^{4096}$
MobileNetV2+MAC	87.1	87.2	87.9	88.0	12.3	13.9	15.5	19.2	28.4
MobileNetV2+MixVPR	88.0	**89.6**	89.0	**90.0**	13.7	15.3	16.9	20.6	29.8
MobileNetV2+NetVLAD	84.1	86.8	86.1	84.8	14.0	15.6	17.2	22.5	30.1
EfficientNet+MAC	87.9	87.9	87.9	88.0	22.8	24.4	27.6	29.7	38.9
EfficientNet+MixVPR	86.9	88.8	88.5	88.3	25.4	27.0	30.2	32.3	43.1
EfficientNet+NetVLAD	81.0	83.5	81.5	81.3	24.1	25.7	28.9	31.8	41.0
ResNet50+MAC	87.1	88.6	88.1	88.8	22.9	24.5	27.7	29.7	39.0
ResNet50+MixVPR	**88.7**	89.0	**89.5**	88.9	27.0	28.6	30.2	32.3	43.1
ResNet50+NetVLAD	85.6	88.8	86.4	86.6	24.1	25.7	28.9	31.0	41.8
VGG-16+MAC	69.9	76.0	76.3	76.3	49.0	50.6	52.3	55.9	65.1
VGG-16+MixVPR	81.0	82.2	81.6	81.8	51.0	52.6	54.2	57.9	67.1
VGG-16+NetVLAD	75.0	76.8	73.9	74.0	60.1	61.7	63.3	66.9	76.2
DINOv2+MAC	87.4	88.1	87.6	87.9	297.4	299.0	300.6	304.4	313.51
DINOv2+MixVPR	68.3	69.9	68.6	69.1	327.5	329.0	332.3	334.3	343.6
DINOv2+NetVLAD	85.5	86.0	86.2	85.2	315.2	316.8	318.4	322.1	331.3

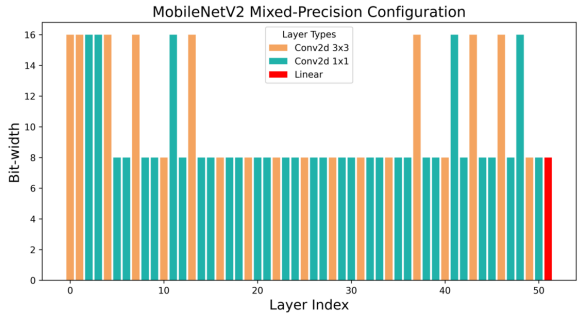

Fig. 7. Mixed-Precision network configuration found under evolutionary search with bit-width budget of 10. All batch-norm layers have been folded into the previous convolution

in each network layer. As this creates a large configuration space it makes a brute force search intractable so we use a genetic algorithm (GA) shown in Algorithm 1 to find the optimal per-layer precisions.

The GA's population P is represented by N mixed-precision candidates I in which I_i represents the bit precision of layer i. We limit the available precisions to int4, int8 and fp16 as those are the precisions currently supported for acceleration in commodity hardware. Given a full precision model M and model Q quantized under configuration I we define the fitness function in Equation 6. Given a limited data sample $\mathbf{x}$ the objective function favours quantization configurations that minimize the difference between the full precision model descriptors and the quantized model descriptors. This however is subject to the constraint in Equation 6 which ensures the average per-layer bit-width precision is below the budget B where T represents the number of layers in Q. The hyperparameter B therefore controls the trade-off between efficiency and accuracy. As in [38] our mutation function randomly alters the precision of a random layer with a probability p_m, where precisions are sampled from a categorical distribution skewed by the sensitivity of the layer to quantization noise. Additionally, we utilize crossover between the two most fit individuals from the population by concatenating random sections of the two parents as we find it improves convergence.

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & f(I) = -\frac{1}{L}\sum_{i=0}^{L}(Q(I;x_i) - M(x_i))^2 \\ \text{subject to} \quad & B \geq \frac{1}{T}\sum_{i=0}^{T} I_i \end{aligned} \tag{6}$$

Fig 7 shows a mixed-precision configuration found by the evolutionary algorithm with a bit-width budget B of 10. It can be seen that higher precisions are favoured in the earlier layers of the network. This may be because the early convolution layers that perform low-level pattern recognition and feature extraction are sensitive to noise. Additionally, quantization of early layers would inject noise into the network that may be amplified by later layers creating a larger objective loss of Equation 6. It is also seen empirically that some higher precision convolution layers are also required to preserve performance later in the network. If using a final linear layer for projection, a lower precision is adequate due to a high level of redundancy in its dense connections.

In Fig 8 we additionally investigate the trade-off between the recall and the average bit-width across the layers of the network $\bar{I}$. Whilst reducing the average bit-width decreases the latency and VRAM memory consumption of the network it also results in a reduction in recall score. A noticeable trend seen across datasets is the significant drop in performance when imposing an average bit-width budget B from Equation 6 below a value of 10.

VII. DESIGN RECOMMENDATIONS AND DISCUSSION

Our work has conducted a comprehensive analysis of the design of resource efficient VPR networks under quantiza-

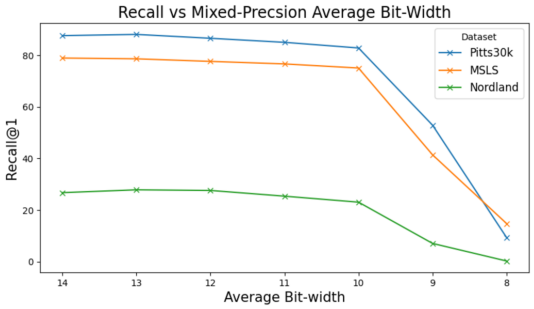

Fig. 8. Recall@1 performance under Mixed-Precision network configurations found with an evolutionary search.

Algorithm 1 Genetic Algorithm Mixed-Precision Search

Input: Objective function $f(I)$, Population size N, Mutation rate p_m, Sample size C, Average Bitwidth B
Output: Best solution found I^*

Initialize population P with N random mixed-precision candidate solutions
while termination condition not met **do**
    Sample C individuals from P to form P^s
    Select parents x_1, x_2 from P^s based on fitness f
    Remove worst individual from P^s based on f
    $offspring \leftarrow$ Crossover(x_1, x_2)
    $offspring \leftarrow$ Mutate$(offspring, p_m)$ subject to B
    Push $offspring$ into P
end while
return the solution in P with the highest $f(I)$

tion. Through experiments across four design points, we've identified crucial insights for developing advanced VPR systems with embedded resource limitations. Our findings provide actionable guidelines for optimizing VPR network efficiency and performance, marking a significant advancement in the field and enhancing practical applications in resource-constrained environments. Our insights are as follows;

Backbone: The choice of backbone architecture plays a crucial role in determining latency and memory efficiency. While DINOv2 is not ideal for real-time embedded perception systems due to its high resource requirements, factorized convolutional models such as MobileNetV2 offer greater resource efficiency. Compared to standard CNN models like VGG-16 or ResNets, MobileNetV2 significantly reduces latency without sacrificing capacity. Therefore, it is recommended wherever possible. However, if latency still poses a bottleneck during deployment, quantization can be effectively utilized due to MobileNetV2s robustness under quantization noise.

Pooling: The impact of pooling methods on extraction latency and memory consumption is minimal. Therefore, the method that offers the highest recall should always be selected. Our results in table I and Fig 5 indicate that MixVPR is an excellent choice, offering the added advantage of enhanced robustness to quantization. However, network designers should be cautious to prevent MixVPR from overfitting to the training data distribution.

Quantization: Quantizing the model weights to fp16 halves the memory footprint and reduces the feature encoding time for convolutional models on average by 40%. Hence fp16 should always be used in a deployment scenario provided there is hardware support. If further reductions are still required, int8 quantization reduces memory consumption by 69% and latency by 52% at inference. However, if quantization noise degrades performance unsatisfactorily and there is still some headroom in terms of memory and latency, a balance can be struck with mixed-precision quantization. The mixed-precision quantization configuration for VPR should leave the initial convolutional layers in higher precision and perform low-bit quantization of the intermediate backbone layers. The final layers excluding any dense fully connected layers should also be kept at higher precision. This configuration however should always be subject to the constraint that the average layer bit-width of the network $\bar{I}$ is above 10 otherwise significant recall performance will be lost.

$$D = \frac{\tau^*_{total} - k_2 \cdot V}{k_1} + \tau_e \tag{7}$$

$$M_{memory} > V \cdot D \cdot 4 \tag{8}$$

Descriptor dimension: The dimension of the descriptor is chosen depending on a multitude of factors. One such factor is that a larger descriptor improves recall. However, as the dimension affects memory consumption it limits the size of the map that can be stored. It also significantly affects the total VPR system latency τ_{total} by increasing the retrieval time τ_r. Provided after query feature extraction the VPR system optimally searches all features in the database, its complexity is linear with descriptor dimension and database size. Thus given the scale of the environment in terms of the number of map images V and the latency target τ^*_{total}, the optimal descriptor dimension can be chosen according to Equation 7 where k_1 is the time taken per unit increase in descriptor dimension and k_2 is the per unit time taken per increase in the number of map images V. This selection must be made subject to the constraints in 8 where M_{memory} is the number of bytes allocated for the database stored in $fp32$.

VIII. Conclusion

Our study investigates four aspects of visual VPR optimization: backbone architecture, pooling methods, descriptor dimensions, and quantization schemes. The findings illuminate key strategies for crafting efficient VPR systems, tailored to specific resource constraints. Essential insights include the impact of pooling on model generalization, the strategic selection of backbone architecture based on device capabilities, the robustness of various architectures to quantization noise, and a new design rule for selecting the optimal descriptor dimension. These guidelines aid in developing

VPR algorithms that are not only effective but also mindful of operational constraints, ensuring optimal performance within given resource limitations. This contributes significantly to the advancement of VPR technology, optimizing computational efficiency and recognition accuracy.

REFERENCES

[1] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[2] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020. [Online]. Available: https://github.com/MIT-SPARK/Kimera

[3] J. J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European Conference on Computer Vision*, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:14547347

[4] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157 vol.2.

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. Springer, 2010, pp. 778–792.

[6] A. Oliva and A. Torralba, "Torralba, a.: Building the gist of a scene: The role of global image features in recognition. progress in brain research 155, 23-36," *Progress in brain research*, vol. 155, pp. 23–36, 02 2006.

[7] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "Brief: Computing a local binary descriptor very fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, 2012.

[8] M. Zaffar, S. Garg, M. Milford, J. Kooij, D. Flynn, K. McDonald-Maier, and S. Ehsan, "Vpr-bench: An open-source visual place recognition evaluation framework with quantifiable viewpoint and appearance change," *International Journal of Computer Vision*, vol. 129, no. 7, pp. 2136–2174, 2021.

[9] A. B. Yandex and V. Lempitsky, "Aggregating local deep features for image retrieval," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1269–1277.

[10] A. S. Razavian, J. Sullivan, S. Carlsson, and A. Maki, "Visual instance retrieval with deep convolutional networks," *ITE Transactions on Media Technology and Applications*, vol. 4, no. 3, pp. 251–258, 2016.

[11] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[12] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning cnn image retrieval with no human annotation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1655–1668, 2019.

[13] A. Ali-bey, B. Chaib-draa, and P. Giguère, "Mixvpr: Feature mixing for visual place recognition," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 2998–3007.

[14] A. Ali-bey, B. Chaib-draa, and P. Giguère, "Gsv-cities: Toward appropriate supervised visual place recognition," *Neurocomputing*, vol. 513, 2022.

[15] G. Berton, G. Trivigno, B. Caputo, and C. Masone, "Eigenplaces: Training viewpoint robust models for visual place recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 11 080–11 090.

[16] N. Keetha, A. Mishra, J. Karhade, K. M. Jatavallabhula, S. Scherer, M. Krishna, and S. Garg, "Anyloc: Towards universal visual place recognition," *IEEE Robotics and Automation Letters*, vol. 9, 2024.

[17] G. Berton, C. Masone, and B. Caputo, "Rethinking visual geo-localization for large-scale applications," vol. 2022-June, 2022.

[18] S. Markidis, S. W. D. Chien, E. Laure, I. B. Peng, and J. S. Vetter, "Nvidia tensor core programmability, performance & precision," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2018, pp. 522–531.

[19] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[20] S. Kim, G. Park, and Y. Yi, "Performance evaluation of int8 quantized inference on mobile gpus," *IEEE Access*, vol. 9, pp. 164 245–164 255, 2021.

[21] G. Berton, R. Mereu, G. Trivigno, C. Masone, G. Csurka, T. Sattler, and B. Caputo, "Deep visual geo-localization benchmark," in *CVPR*, June 2022.

[22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, 2004.

[23] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[24] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22. Prague, 2004, pp. 1–2.

[25] R. Arandjelovic and A. Zisserman, "All about vlad," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2013, pp. 1578–1585.

[26] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," 2023.

[27] B. Ferrarini, M. Milford, K. D. McDonald-Maier, and S. Ehsan, "Highly-efficient binary neural networks for visual place recognition," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 5493–5500.

[28] J. Bethge, H. Yang, M. Bornstein, and C. Meinel, "Binarydensenet: Developing an architecture for binary neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[29] S. I. Young, W. Zhe, D. Taubman, and B. Girod, "Transform quantization for cnn compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5700–5714, 2021.

[30] F. Warburg, S. Hauberg, M. Lopez-Antequera, P. Gargallo, Y. Kuang, and J. Civera, "Mapillary street-level sequences: A dataset for lifelong place recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[31] D. Olid, J. M. Fácil, and J. Civera, "Single-view place recognition under seasonal changes," in *PPNIV Workshop at IROS 2018*, 2018.

[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[34] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2018.

[35] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size," 2016.

[36] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6105–6114. [Online]. Available: https://proceedings.mlr.press/v97/tan19a.html

[37] S. Yun and A. Wong, "Do all mobilenets quantize poorly? gaining insights into the effect of quantization on depthwise separable convolutional networks through the eyes of multi-scale distributional dynamics," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021, pp. 2447–2456.

[38] Y. Yuan, C. Chen, X. Hu, and S. Peng, "Evoq: Mixed precision quantization of dnns via sensitivity guided evolutionary search," in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.