SqueezeNetVLAD: High-speed power and memory efficient GPS less accurate network model for visual place recognition on the edge

Chandrajit Pal, Pratibha Verma *, Himanshu Rohit †, Dubacharla Gyaneshwar ‡,
Sumohana S. Channappayya, Amit Acharyya §
* Engineering Physics, Department of Electrical Engineering, Indian Institute of Technology (IIT), Hyderabad, India.
{chandrajit.pal, sumohana, amit_acharyya}@ee.iith.ac.in
* ee21resch01007, † ep20btech11008, ‡ ee21pdf16}@iith.ac.in, § corresponding author

***Abstract*—Visual place recognition (VPR) provides the ability for a machine to recognise any given place if it has been visited before by utilising visual graphics data. Classical handcrafted feature extraction doesn't seem to perform well under varying environmental conditions, which is where the convolutional neural networks with their state-of-the-art performance act as a substitute however at the cost of an increased computational complexity making it difficult to port into resource-constrained edge devices like tiny robots, drones etc. This motivated us to propose an optimal NetVLAD network algorithm consisting of an optimised and efficient feature extraction base network performing feature extraction while satisfying both a given constraint of accuracy and the resource budget of a target implementation platform. Both the accuracy, which is marginally better, the achieved model size reduction is better by 17% and a nominal power reduction by 5% w.r.t the SOTA models. Also developed a customised CUDA kernel that dynamically compares the pair of query image feature vectors to the descriptor vectors of the already stored gallery image datasets in parallel utilising the maximum available tensor cores achieving hard-real time performance at 62.5 fps with input videos tested on the Nvidia Jetson Xavier NX platform.**
[1]

Keywords - NetVLAD Network with Vector of Locally Aggregated Descriptors, Visual place recognition (VPR), SqueezeNetVLAD, Convolutional Neural Network (CNN), Pittsburgh250K, TokyoTM

I. Introduction

Visual place recognition (VPR) based vision applications provide the ability to a machine to recognise a location or a given place if it has visited before using visual information. In computer vision applications VPR is a fundamental and a very important task for autonomous navigation systems. This technology assists any mobile avionics system such as a drone, UCAV etc to accurately navigate and position itself or to the target where positional tracking like GPS systems fails due to errors arising out of unstable infrastructure in emergency conditions. This technology faces various challenges that can creep in due to variations in weather, surrounding illumination, seasonal changes etc. Classical handcrafted feature extraction doesn't seem to perform well under varying environmental conditions, which is where the convolutional neural networks with their state-of-the-art feature extraction performance act as a substitute. However, it comes at the cost of an increased computational complexity resulting in an increased model size and execution time making it difficult to port into resource-constrained edge devices like tiny robots, and drones.

Lightweight drones, tiny avionic robots in space, defence and next-generation warfare are generally fitted with less complex hardware platforms that become a bottleneck to hosting heavy-weight complex algorithms with satisfying performance. Enhancing the efficiency while reducing memory occupancy footprint and computational effort to execute a model without disturbing the performance is challenging for these kinds of tiny avionic devices. Generally, mission-critical applications like defence and industrial automation demand hard real-time completion requirements and excellent performance accuracy. This necessitates the need to design a low-complex VPR algorithm.

A few pieces of literature related to our study have been highlighted below. Authors in [3] performed a comparative study of the SOTA local features descriptors from accuracy and computational perspective for VPR applications with benchmark datasets and presented the existence of a trade-off between accuracy and computational attempt while executing VPR on a resource-constrained platform. Authors in [10] proposes a novel pipeline for viewpoint-tolerant place recognition utilising promising leads from existing works, enabling unprecedented robustness to a wide range of common challenges exploiting both efficient binary features and noisy estimates of the local 3D geometry, computed for visual-inertial odometry onboard the UAV. A lightweight CNN has been proposed by CALC [11] to address the problem of loop closure detection being trained with an autoencoder to create the HOG descriptor from geometrically distorted location images. Authors in [2] Cross-Region-Bow, the regional maximum activation of convolutions (R-MAC) [4], CAMAL [8], and Region-VLAD [7] concentrates on features pooling from a pretrained neural network since they are considered two separate stages namely feature extraction and aggregation. Recent state-of-the-art studies like Patch-

[1]This work has been partially funded by DRDO and the tools have been supported by MEITY, Government of India.

NetVLAD [5] and MultiRes-NetVLAD [9] are known to perform extremely well, however, they come at the cost of increased memory requirements due to the storage of detailed patches of generated features at multiple scales making it a perfect fit for systems with nominal resources constraints. While NetVLAD [1] with its simplicity solves this problem of storage requirement and consists of two stages trained end-to-end. The first stage consists of a benchmark CNN like AlexNet or VGG-16 network meant for image feature extraction followed by an aggregation layer combining them in a VLAD-like descriptor [6]. It comes with a powerful pooling mechanism consisting of the learnable parameters with every differentiable function that can be easily integrated into any CNN architecture. As mentioned earlier due to its simplicity and convenience of usage together with its good performance, it is still preferred in visual place recognition tasks by the vision community.

However, since its base architecture is still a CNN, it has an inherent high complexity making it difficult to be used in lightweight avionics systems. Also in the deployment phase when the terrain covers a larger area, it results in a large database of gallery images, which increases the search time for the query image to find its closest match from the gallery. This necessitates using parallel computing to accelerate these feature comparing and matching steps for obtaining a close match in hard real-time scenarios. To cater a solution to these existing problems, we have proposed an optimised network SqueezeNetVLAD algorithm. Our contributions are highlighted below:

- We propose an optimised feature extraction base network that has been integrated into the VLAD layer performing feature extraction while satisfying both a given constraint of accuracy and the resource budget of a platform. Both the accuracy, which is marginally better, the achieved model size reduction is better by 17 % and a nominal power reduction of 5 % w.r.t the SOTA models.
- Developed a customised CUDA kernel that dynamically compares the pair of query image feature vectors to the descriptor vectors of the already stored gallery image dataset of Pittsburgh250k [14] and TokyoTM [1] dataset in parallel with the maximum available tensor cores achieving real-time performance at 62.5 fps with input videos tested on Nvidia Jetson Xavier NX platform [13].

II. THE PROPOSED METHODOLOGY

Here we propose to design a lightweight but accurate visual place recognition algorithm and coined it squeezeNetVLAD. Motivated by the existing NetVLAD architecture on account of its simplicity in working principle. We proposed a lightweight version of the original NetVLAD architecture as illustrated in figure 1. The first 14 layers consist of a convolutional network with 3×3 homogeneous set of filters throughout with periodical activations ReLU and maxpooling. This is followed by the Vector of Locally Aggregated Descriptors (VLAD core) layer generating the feature descriptors of the image that needs to be matched with the feature descriptors/vectors of the stored images in the database.

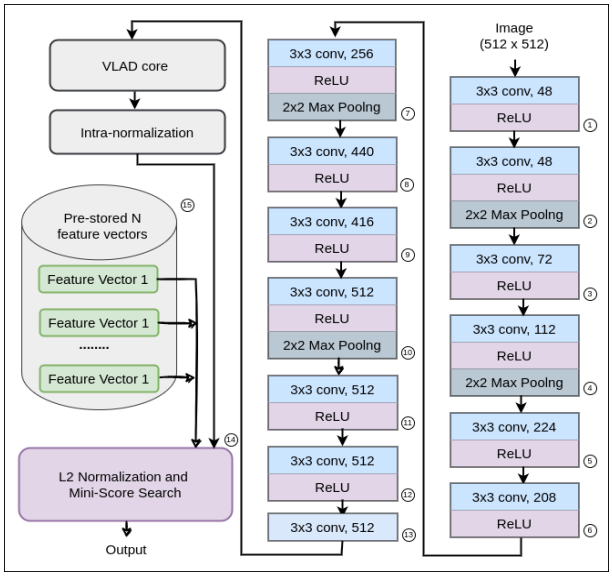

Figure 1: Network architecture of the proposed system architecture: SqueezeNetVLAD

We have selected the number of filters starting from layer 1 to 14 in the following manner as compared to its original counterpart as (Classical model arch $\rightarrow$ Optimised version) $64 \rightarrow 48$, $64 \rightarrow 48$, $128 \rightarrow 72$, $128 \rightarrow 112$, $256 \rightarrow 224$, $256 \rightarrow 208$, $512 \rightarrow 440$, $512 \rightarrow 416$ etc until it reaches the VLAD core where the descriptors are generated and matched with the stored gallery image descriptors in module 15 for finding a quick and close match for recognition in module 14 (ref Figure 1).
The flowchart is illustrated in figure 2. Let's describe it in conjunction with Algorithm 1. We aimed to maximise or retain the accuracy of the original model $Prediction_{Acc}$, however at a reduced model size given some budget in terms of resource $Resource_i > Thr$, $and\ Original_{Acc} \leq \underset{Max}{Predicted_{Acc}}$ while maintaining the constraints, where Thr is the given resource budget/threshold with some percentage magnitude, $Predicted_{Acc}$ is the predicted accuracy.

Overview of the algorithm:

As shown in the flowchart (figure 2), the available pretrained NetVLAD model is taken as the input. Upon analysing its architecture, we started optimising each individual layer of the model as follows:

- Selecting the number of filters at each layer of the input model empirically. And the maximum number of filters that can be fit given a resource threshold is chosen. The channels are also adjusted accordingly after the removal of filters.

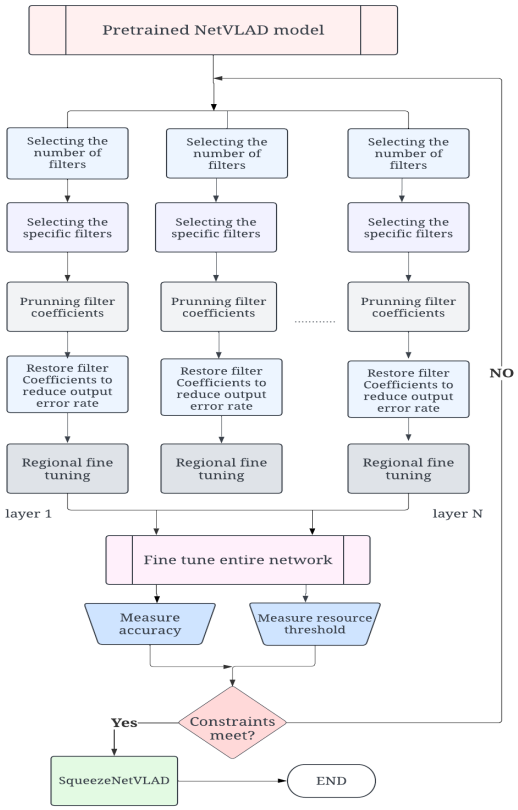

Figure 2: Flowchart of the proposed algorithm

- Filters which have the largest $l2$ norm are kept among all the filters in a layer.
- For a particular filter, its weights are pruned based on its magnitude, if it is less than a particular threshold, which is selected based on accuracy and complexity tradeoffs.
- Sometimes weights are pruned at a higher compression ratio than the target. In these scenarios, weights/coefficients are restored to meet a certain target compression ratio to minimise the output error.
- Having discussed the removal of filters and their weights, it is very significant to change the values of the remaining weights of the unpruned filters to further reduce the output error through least-square optimization.
- Having pruned and restored filters and their weights in all the layers, it is time to fine-tune the entire network to further increase the accuracy.
- Following this the achieved accuracy and the present resource in terms of the number of trainable parameters are measured against the user-defined constraints. If it is satisfied, the optimised net is achieved else further iteration is needed.

Relating to the Algorithm 1, pretrained NetVLAD model TN_i consisting of N no of layers including convolutional and VLAD layers. The present resource in terms of trainable parameters is measured in line 2. Until the resource in terms of trainable parameters and accuracy is met which is given as a condition in line 3, iteratively the algorithm is repeated.

Algorithm 1 Generating squeezeNetVLAD

Input: Trained Network NetVLAD: $TN0$ (consisting of N number of $Conv$ and $VLAD$ layers), Resource threshold: Thr, Resource Reduction Rate: Δ R
Output: Optimal Network meeting the threshold constraint:$squeezeNetVLAD$

1: $i = 0$
2: $Resource_i$ = Res(TN_i)
3: **while** (Resource$_i > Thr$, $Original_{Acc} \leq Predicted_{Acc}$)
4: $Con = Resource_i$ - ΔR_i
5: **for** $l = 1$ to N
6: TN_simple, No_of_filters(Num_l) = $Num(TN_i, l, Con)$
7: TN_simple, target_filters($Trgt_l$) = $Trgt(TN\_simple, l, Num_l)$
8: **for** $j = 1$ to $Trgt_l$
9: $Trgt_j$=Prune($W(Trgt_j - 1) \leq \eta_{Th}$)
10: $\widehat{Trgt_j} = arg\ min \left\| \widehat{F_y} - F_x.\widehat{Trgt_j} \right\|_n^n$ subject to $\left\| \widehat{Trgt_j} \right\|_0 \leqslant n$,
11: where n = number of nonzero weights we like to maintain.
12: $\overline{Trgt_{jR}} = \underset{\widehat{Trgt_{jR}}}{arg\ min} \left\| F_{yj} - F_{xjR}.\widehat{Trgt_{jR}} \right\|_2^2$, $\overline{Trgt_{jR}}\mathbb{C} = 0$ /* Fine-tune regional weights through least square optimisation
13: **End for**
14: $(TN\_simple, Resource_i, \overline{Trgt_l}) = \underset{\widehat{Trgt_l}}{arg\ min} \left\| F_{yl} - F_{xl}.\widehat{Trgt_l} \right\|_{conv}$, $\overline{Trgt_{iR}}\mathbb{C} = 0$ /* Fine-tune entire net
15: $Prdicted_{Acc} = Acc(TN\_simple)$
16: **End for**
17: $i = i + 1$
18: **End while**
19: squeezeNetVLAD = TN_simple
20: **return** squeezeNetVLAD

Lines 5 and 8 denote the total number of layers N and the target filters that are kept in each layer are denoted as $Trgt_l$. Line 4 denotes the rate at which the constraints tighten at every iteration on the network layers. Selection of the number of filters is performed empirically and which filters are to be targeted are also decided based on the target constraints in terms of both resource and accuracy as shown in lines 6 and 7.

For a particular target filter weights/its coefficients are pruned if it is below a particular threshold value η_{Th} in line 9. Say F_x and F_y are the input and output feature maps respectively and input features when convolved with target filters $Trgt_l$ produce the output features. So with a view to reduce the output error rate $l()$ minimisation is carried out. In this process, few weights are restored among multiple filters by minimising the large residuals (i.e. present output feature map difference) in every iteration by selecting the weight that reduces the $l2 - norm$ of the corresponding residual. This process continues until it reaches n which is the number of non-zero weights that need to be retained in all the filters as shown in lines 10 and 11 respectively. After pruning it is very important to update the values of the remaining weights of the unpruned filters to further reduce the output error through least-square optimization to fine-tune the regional weights of the pruned weights as shown in line 12. It has a closed-form solution and can be efficiently solved.

Finally, after the complete surgery of the entire network, the weights of the entire network are fine-tuned through backpropagation to attain the maximum possible accuracy. The final accuracy is predicted and resource in terms of trainable parameters is computed and compared with the given threshold constraints, based on which it is decided to terminate or iterate as shown in lines 18 to 20.

This obtained network including the VLAD core and

Table I: Illustrates the comparison w.r.t inference time, power consumption, model size and Recall accuracy of various variants of NetVLAD algorithm with different base architectures tested on Pittsburgh250K and TokyoTM datasets with our proposed model SqueezeNetVLAD

Base CNN for NetVLAD	Pittsburgh 250 K							Tokyo TM						
	Recall(R)				Inference	Power	Model	Recall(R)				Inference	Power	Model
	R@1	R@5	R@10	R@20	Time (ms)	Consumed(mW)	size (Mb)	R@1	R@5	R@10	R@20	Time (ms)	Consumed(mW)	size (Mb)
AlexNetBase [1]	55	68	73	77.5	7	14280	18	89	92	95	95	6	10480	9.9
VGG16Base [1]	76	88	91	91.5	20	20400	86	93	94	96	96	18	19400	54.9
SqueezeNetVLAD (Proposed)	**76.63**	**89**	**92.19**	**95.14**	16	19800	72	**94.26**	**95.53**	**96.7**	**96.7**	14	18600	45.57

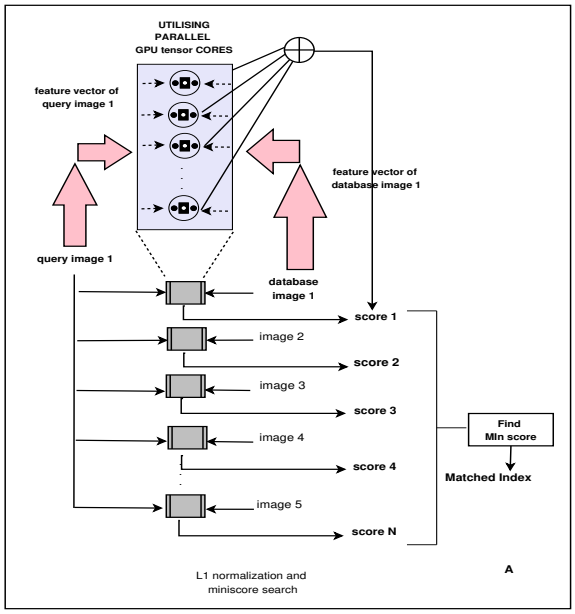

Figure 3: Overview of the proposed $l1$ normalisation and miniscore computation architecture module

the $l2$ normalisation and miniscore computation architecture module has been shown in the 14th module in figure 1 and detailed in figure 3. Referring to both figures 1 and 3, once the query image of resolution 512×512 is sent to the network, after feature extraction and descriptors being generated from layers 1 to 13 and through VLAD core, it enters into the $l2$ normalisation and miniscore computation block. Here a customised CUDA kernel has been developed that dynamically assigns copies of the query image feature vector to the available tensor cores where it compares to the descriptor vectors of the already stored gallery image dataset of Pittsburgh [14] and TokyoTM [1] dataset in parallel and the $l2$ normalisation is computed for every pair of the query and the gallery image features vectors. Out of these only the pair with the least $l2$ normalisation score value is selected as the best match index.

III. EXPERIMENTAL RESULTS

This architectural design entry has been coded using Python 3.7.8 with Pytorch machine learning framework [12]. The system configuration uses Intel Xeon(R) CPU ES-2650 v4 @ 2.20 GHz$\times$24, 32 GB primary memory with Ubuntu 20.04.3 LTS 64-bit. The inference has been done on the edge computing platform Nvidia Jetson AGX Xavier NX board [13]. Our design has been mapped to the 384-core NVIDIA Volta™ GPU with 48 Tensor Cores onboard the system has been utilised for the inference. Table I illustrates a comparison among the Recall values, inference time, power consumption and the model size of the various variants of the NetVLAD algorithm with different base architectures tested on Pittsburgh250K and TokyoTM benchmark datasets with our proposed model SqueezeNetVLAD. Like most of the state-of-the-art, we evaluated using the Recall evaluation metric as the sensitivity or true positive rate. Recall from 1, 5, 10 to 20 have been reported and the values of the proposed SqueezeNetVLAD have been shown in bold as we experimented on the benchmark datasets with existing state-of-the-art methodologies. We exceeded by marginal quantities with respect to the one with the VGG16Base version. However, our reported percentage with respect to the AlexNetBase version is far higher. Our proposed model size versions are 72 Mb and 45.57 Mb as trained and tested on the benchmark datasets which is 17 % less in comparison to the latest VGG16Base versions.

This model size reduction has brought about other advantages with reduced inference time and reduced power consumption by a maximum of 23% and 5% w.r.t the other SOTA models on both the benchmark datasets (viz Table I). The other benchmark models are also tested on the identical Nvidia Xavier platform along with similar datasets. The hyperparameter settings for our network are a Learning rate of 0.001, CrossEntropyLoss as the loss function, and Stochastic Gradient Descent as optimiser with batch size 32. We have kept the budget threshold as 40 % i.e. at max this percentage reduction is allowable without disturbing the accuracy of the model w.r.t the baseline version of VGG16Base [1].

IV. CONCLUSION

In this study, we propose an optimised SqueezeNetVLAD model performing satisfactory visual place recognition tasks on benchmark datasets like Pittsburgh250k and TokyoTM, maintaining the resource budget of a platform. The accuracy is marginally better and the achieved model size reduction is better by 17% w.r.t the SOTA models. This model size reduction has reduced inference time and power consumption by a maximum of 23% and 5% w.r.t the other SOTA models on both benchmark datasets. Also developed a customised CUDA kernel that dynamically compares the pair of query image feature to the descriptor vectors of the stored gallery image dataset in parallel with the maximum available tensor cores achieving real-time performance at 62.5 fps with input videos tested on Nvidia Jetson Xavier NX platform.

REFERENCES

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5297–5307, 2016.

[2] Zetao Chen, Fabiola Maffra, Inkyu Sa, and Margarita Chli. Only look once, mining distinctive landmarks from convnet for visual place recognition. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9–16, 2017.

[3] B. Ferrarini, M. Waheed, S. Waheed, S. Ehsan, M. Milford, and K. D. McDonald-Maier. Visual place recognition for aerial robotics: Exploring accuracy-computation trade-off for local image descriptors. In *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 103–108, Los Alamitos, CA, USA, jul 2019. IEEE Computer Society.

[4] Hervé Jégo Giorgos Tolias, Ronan Sicre. Particular object retrieval with integral max-pooling of cnn activations. In *ICLR 2016 - International Conference on Learning Representations, May 2016, San Juan, Puerto Rico.*, pages 1–12, 2016.

[5] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. Patch-netvlad: Multi-scale fusion of locally-global descriptors for place recognition. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14136–14147, 2021.

[6] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3304–3311, 2010.

[7] Ahmad Khaliq, Shoaib Ehsan, Zetao Chen, Michael Milford, and Klaus McDonald-Maier. A holistic visual place recognition approach using lightweight cnns for significant viewpoint and appearance changes. *IEEE Transactions on Robotics*, 36(2):561–569, 2020.

[8] Ahmad Khaliq, Shoaib Ehsan, Michael Milford, and Klaus D. McDonald-Maier. CAMAL: context-aware multi-scale attention framework for lightweight visual place recognition. *CoRR*, abs/1909.08153, 2019.

[9] Ahmad Khaliq, Michael Milford, and Sourav Garg. Multires-netvlad: Augmenting place recognition training with low-resolution imagery. *IEEE Robotics and Automation Letters*, 7(2):3882–3889, 2022.

[10] Fabiola Maffra, Zetao Chen, and Margarita Chli. Viewpoint-tolerant place recognition combining 2d and 3d information for uav navigation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2542–2549, 2018.

[11] Nate Merrill and Guoquan Huang. Lightweight unsupervised deep loop closure. *CoRR*, abs/1805.07703, 2018.

[12] Linux Foundation Meta AI. Pytorch: An open source machine learning framework, year = 2022, url = https://pytorch.org/ , urldate = 2022.

[13] Nvidia. Jetson AGX Xavier Series Modules, year = 2022, url = https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/, urldate = 2022.

[14] Akihiko Torii, Josef Sivic, Tomá Pajdla, and Masatoshi Okutomi. Visual place recognition with repetitive structures. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 883–890, 2013.