Self-Supervised Learning and Network Architecture Search for Domain-Adaptive Landmark Detector

Takeda Koji

Tanaka Kanji

Abstract—Fine-tuning a deep convolutional neural network (DCN) as a place-class detector (PCD) is a direct method to realize domain-adaptive visual place recognition (VPR). Although the PCD model is effective, a PCD model requires considerable amount of class-specific training examples and class-set maintenance in long-term large-scale VPR scenarios. Therefore, we propose to employ a DCN as a landmark-class detector (LCD), which allows to distinguish exponentially large numbers of different places by combining multiple landmarks, and furthermore, allows to select a stable part of the scenes (such as buildings) as landmark classes to reduce the need for class-set maintenance. However, the following important questions remain. 1) How we should mine such training examples (landmark objects) even when we have no domain-specific object detector? 2) How we should fine-tune the architecture and parameters of the DCN to a new domain-specific landmark set? To answer these questions, we present a self-supervised landmark mining approach for collecting pseudo-labeled landmark examples, and then consider the network architecture search (NAS) on the LCD task, which has significantly larger search space than typical NAS applications such as PCD. Extensive verification experiments demonstrate the superiority of the proposed framework to previous LCD methods with hand-crafted architectures and/or non-adaptive parameters, and 90% reduction in NAS cost compared with the naive NAS implementation.

I. INTRODUCTION

Visual place recognition (VPR) aims to predict the viewpoint of a robot with respect to a pre-trained environment model or a map database, given a query on-board live image. It can be applied to benefit a wide range of autonomous-vehicle applications [1]–[3]; hence, it has attracted increasing attention in recent years. However, long-term VPR is one of the most challenging scenarios, where the VPR is affected by either domain shifts or the difference in appearances due to factors such as changes in weather, time of day, seasons, and (semi-)dynamic objects. The domain shifts deteriorate the VPR performance. Therefore, we must adapt the VPR system to a new target domain. Additionally, the training data for the adaptation must be collected by the robot self in a self-supervised manner without requiring human intervention.

A direct method to address the aforementioned challenge is to fine-tune a deep convolutional neural network (DCN)-based classifier or place-class detector (PCD), so as to classify N place classes in the target domain. Its training data, (viewpoint-annotated place-specific training images),

Our work has been supported in part by JSPS KAKENHI Grant-in-Aid for Scientific Research (C) 26330297, and (C) 17K00361.

The authors are with University of Fukui, 3-9-1, bunkyo, Fukui, Japan. tnkknj@u-fukui.ac.jp

Fig. 1. Proposed approach mines a set of landmark examples from an unlabeled target sequence and trains a domain-adaptive landmark class detector (LCD). Top: Input target map images (left) and feature points (right). Bottom: Landmark objects detected in paired target map (left) and live query images (right), respectively. It is evident that these objects are place specific and invariant against domain-shifts.

can be generated from a training view-sequence, in a self-supervised manner, by partitioning the robot workspace into N place classes [4], and reconstructing the viewpoint trajectory [5]. As a key advantage, the outputs of a DCN classifier can be viewed as an extremely compact (i.e., $\log N$ -bit), discriminative and semantic description of the input image, which can be directly used for performing database indexing and information retrieval. Recently, the PCD model has achieved considerable success in several state-of-the-art VPR systems [6].

Although the PCD model is effective, it might require a significant maintenance cost in the long-term VPR scenario. First, the PCD approach essentially requires the re-training of the DCN whenever a new appearance change is detected. Moreover, it requires a considerable amount of annotated training images to cover large-scale environments. Furthermore, it might be affected by the annotation noises during the self-supervised annotation process. Therefore, it would be unwise to rely solely on this approach.

Accordingly, we propose a new approach, called self-supervised domain-adaptive landmark-detector architecture

(SSDADA), that trains a DCN as a landmark class detector (LCD). Unlike the PCD approach, the proposed LCD approach detects a minimal set of multiple landmark objects in a given query/map image, on the basis of their object bounding boxes (OBBs) and class labels, and subsequently uses them to describe the image (Fig. 1). This new proposed LCD approach has several attractive characteristics. (1) Unlike PCD, an LCD can distinguish exponentially large numbers of different places by combining multiple landmarks. (2) An LCD inherits the discriminative power of a DCN, as evident from previous PCD approaches [6]. (3) The landmark ID is an extremely compact, discriminative, and semantic representation. (4) Unlike PCD, a stable part of the scenes (such as buildings) can be selected as landmark classes to reduce the need for class-set maintenance.

An obvious method to train an LCD system [7] would be to mine a minimal set of landmark-region examples with class labels from the training view-sequence, and subsequently fine-tune a DCN on the basis of these examples. However, the following important questions remain: 1) How we should mine such domain-specific landmarks even when we have no domain-specific object detector? 2) How we should fine-tune the architecture and parameters of the DCN to the target domain? To answer the first question, we present a self-supervised landmark-mining approach for collecting pseudo-ground-truth OBBs with landmark class labels from a given unlabeled target sequence. To answer the second question, we consider the problem of network architecture search (NAS) [8] on a novel application, an LCD.

Our main contributions are summarized as follows. 1) We address a new fully self-supervised fine-tuning of an LCD for long-term VPR, without requiring any labeled target data. 2) We formulate the fine-tuning of an LCD as an NAS problem [8], and the problem has a significantly larger design space than that of typical NAS applications such as classification tasks [8]. Thereby, we provide a feasible solution for this problem. 3) We conduct extensive verification experiments on VPR in new domains by using the publicly available RobotCar dataset [9]. We confirmed that the proposed method stably finds the optimal LCD network architecture and outperforms the baseline landmark detectors with hand-crafted architectures and/or non-adaptive parameters.

II. SELF-SUPERVISED LCD FRAMEWORK

A key challenge in mining a training set, i.e., domain-specific object examples, from an unlabeled map is: “how to mine domain-specific object examples even when we do not yet have a domain-specific object detector?”. We address this challenge by exploiting generic object-segmentation techniques, such as that in [10]. The basic idea is to aggregate domain-agnostic object segments into domain-specific landmark regions by using the domain-specific spatial context information as a guide (Fig. 2). In summary, our approach comprises the following two stages: 1) domain-agnostic object segmentation and 2) domain-specific landmark aggregation.

The aforementioned stages are detailed in the remainder of this section.

A. Generic Object Region Proposal

We use the semantic segmentation network in [11] to obtain generic object-region proposals. The segmentation network is pre-trained on the cityscapes dataset [12], and it predicts pixel-wise semantic labels with region contours. The semantic labels include sidewalks, buildings, walls, fences, poles, traffic lights, traffic signs, and vegetation. However, the aforementioned predicted semantic labels are frequently affected by the domain shift between the training and testing sets. Nevertheless, we observed that the predicted semantic labels were often useful for clustering in-domain regions, as demonstrated in Section II-B.

B. Domain-Specific Region Aggregation

We exploit the domain-specific optical flow (OF) for performing landmark aggregation. This approach is supported by the recent findings in [13] that semantic regions that are tracked for a sufficiently long period are probably stable parts of the scenes. OFs are extracted using the KLT tracker [14], and OF point trajectories shorter than 10 frames are discarded. Approximately 1000 KLT features are obtained for a 512×512 image.

Subsequently, the OFs are refined in the following two steps: (1) selection of reliable OFs and (2) selection of reliable OBBs. In the first step, each point on the spatio-temporal point trajectory of an OF is assigned the semantic label of the segment to which the point belongs, and we check whether semantic labels change for more than 30 frames. Subsequently, the candidate OBBs are computed for those regions to which at least one reliable OF belongs. In the second step, the candidate OBBs in the successive frames are linked to an OBB sequence if the intersection-over-union (IoU) between an OBB and its predecessor OBB (i.e., a previous OBB in the sequence) exceeds 0.7. The length of the search window for the predecessor is fixed to five frames. Short OBB sequences with lengths shorter than 60 frames are excluded. Each of the remaining OBB sequences is assigned a unique object ID.

III. DETECTOR ARCHITECTURE SEARCH

Generally, an NAS algorithm is the iteration of the following two distinct steps: proposal and evaluation. The proposal step aims to generate a network-architecture candidate. To that end, we follow the literature and use the DCN provided in [15] as the backbone network and, subsequently, employ an efficient initialization of it. The evaluation step aims to evaluate the performance of the candidate architecture by testing it on object-detection tasks. The overall cost is dominated by the evaluation step, and it is proportional to both the number of candidate architectures proposed and complexity of the test detection tasks. To reduce the infeasible computation cost, we employ a lightweight proxy task in place of the original LCD task.

Fig. 2. Self-supervised annotation of target sequence.

A key issue is “how to design a good proxy task?”. A proxy task should take the same input-output format as that taken by the original task and perform satisfactorily on the proxy task if and only if they perform satisfactorily on the original task as well, thereby requiring empirical methods to evaluate the translation quality appropriately. Therefore, we investigated several possible translation strategies in terms of quality, and they will be discussed in Section III-C.

A. FPN Design Space

An FPN takes multiscale feature layers as inputs and generates output-feature layers in identical scales. The output of the first pyramid network is the input to the next pyramid network. An FPN architecture merges any two input-feature layers into an output-feature layer. An FPN consists of N different “merging cells”, where the value of N is given during the search. Each merging cell takes two input-feature layers, applies processing operations, and then combines them to produce one output-feature layer of a desired scale. We take as input features 5 scales, namely, C2, C3, C4, C5, and C6 with the corresponding feature strides of 4, 8, 16, 32, and 64 pixels. C6 is created by applying stride 2 max pooling to C5. The input features are then passed to a pyramid network, which consists of a series of merging cells that introduce cross-scale connections. Subsequently, the pyramid network outputs augmented multiscale feature representations (P2, P3, P4, P5, and P6). Fig. 5 depicts an example of such an FPN architecture.

B. NAS Formulation

The NAS module finds a satisfactory controller that determines how to construct the merging cells. We follow the work in [16] and chose a recurrent neural network (RNN) as the controller. The RNN controller selects any two candidate feature layers and a binary operation to combine them into a new feature layer, where all the feature layers may have different resolutions. Each merging cell performs four prediction steps as follows. **Step-1:** Select the i -th feature layer from the candidates. **Step-2:** Select another j -th feature layer from the candidates. **Step-3:** Select the output-feature resolution. **Step-4:** Select a binary operation to combine the i -th and j -th layers and, subsequently, generate a feature layer with the resolution selected in Step-3.

For Step-4, we follow the NAS-FPN approach in [16] and implement two different binary operations, namely, sum pooling and global pooling. The input-feature layers are adjusted to the output resolution via nearest-neighbor upsampling or max pooling, if required, before applying the binary operation. The merged feature layer is always followed by a ReLU, 3x3 convolution, and batch-normalization layer. The

input-feature layers to the pyramid network form the initial list of the input candidates of a merging cell.

For Step-5, the newly generated feature layer is appended to the list of the existing input candidates, and it becomes a new candidate for the next merging cell. Notably, multiple candidate features might share the same resolution during the architecture search. Finally, the last 5 merging cells are designed to output feature pyramids, namely, P2, P3, P4, P5, and P6. The order of the output feature levels is predicted by the controller. Each output-feature layer is, then, generated by repeating Steps-1, 2, and 4 until all the layers are selected.

C. Proxy Task

Recently, a few studies have attempted to translate a given target task to a proxy one. The works in [17] and [18] proposed generating a proxy dataset and used it instead of the original dataset. The work in [18] proposed the use of a small backbone network and small input-image size. Such a method would be effective when the input image size is large. However, in our case, wherein the image size is previously small (i.e., 512×512), performance cannot be improved by the aforementioned method.

In this study, we propose to initialize the backbone network as a generic pretrained DCN. Specifically, the initial network weights are pre-trained on the public VOC2007 dataset [19]; subsequently, it is fine-tuned on the proxy task. This often avoids network-convergence problems and stabilizes the training within a considerably short training time.

We empirically observe that competitive performance can be achieved using such a subset of the available training set, instead of using an external evaluation dataset as used in [18].

To sample such a subset from the available training set, the equally spaced sampling of the entire training sequence is performed to preserve the overall inter-class spatial distribution.

For fine-tuning, a naive method is to initialize the network parameters with the pre-trained weights and set all the parameters as trainable. However, we find that upon freezing the parameters, we achieve better performance (in terms of the performance for 200 generations). Our strategy significantly reduces the number of iterations of the proxy task from 1120K to 3.6K with batch size 4, corresponding to the training time of 1 h per generation (1 GPU, Nvidia RTX 2080).

The controller RNN is trained using the policy gradient method. The controller samples the child networks with different architectures, which are trained on the proxy task. The resulting detection accuracy in average precision (AP)

on a proxy validation set is used as the reward to update the controller.

D. Implementation Details

The resnet-v1-50 [15] was used as the backbone network. The entire network was built using the following steps.

First, a base network was built. We implemented a 2D convolutional network as the base network, which has the output size of 64, kernel size of 7, and stride of 3. For the resnet-v1-50, the outputs of the C{2, 3, 4, and 5} have channel sizes 64, 128, 256, and 512, respectively. Subsequently, C6 was obtained by pooling C5. We then organized C{1, 2, 3, 4, 5, and 6} with the pyramidal shapes, and integrated outputs using a candidate architecture proposed by NAS.

An RPN consists of a convolutional layer with the kernel size of 3x3 and output channels of size 512. The class prediction layer has a convolution with 6 channels, a kernel size of 1x1, and stride of 1. The bounding box prediction layer has a convolution with 12 channels, and stride 1.

The anchor size is defined for each of C{2, 3, 4, 5, and 6}, where each size is initialized as 32, 64, 128, 256, and 512, respectively. Subsequently, each is searched during the training process by ratio of 1:1, 1:2, and 2:1 processes.

Our RPN uses a series of standard postprocessing techniques, such as decoding, clipping, NMS, and region of interest.

Subsequently, the candidate Fast-RCN architecture is built. Provided a candidate P-list, we compute the region of interest and use it to compute the class-prediction probability.

We designed a multi-task loss. For the classification task, the joint supervision of both the center loss and softmax loss is used. For the bounding box regression task, the smooth L1 loss was used. Consequently, the entire loss function becomes the following:

$$L(p, t, x) = L_c(p, p^*) + \lambda L_r(b, b^*), \quad (1)$$

where p denotes the prediction of the object class, p^* the ground-truth. b the prediction of OBB and b^* its ground-truth. The hyperparameters control the balancing weights among the three terms, and are determined by $\lambda=1$.

The network-initialization process is as follows. The convolutional layers of RPN, namely, the class-prediction layer, and class head of Fast-RCNN, are initialized as follows: the mean of the random values to be generated is set to 0, and the standard deviation of the random values to be generated is set to 0.01. One GPU (GeForce RTX 2080) was used for NAS, and four were used for training. The resnet block was frozen for training and searching (P2). The regression loss adopted in our method was smooth L1.

IV. EXPERIMENTS

To verify our framework, we test it on two public RobotCar datasets [9], both of which are first introduced in this section. These datasets cover challenges such as diverse object categories, confusing object appearances, occlusions, and dynamic objects. Based on the datasets, we investigated the LCD and VPR performance. Specifically, we compared

Fig. 3. Spatial distribution of LCD results. Shown are the viewpoints of true-positive detections overlaid on the top-view image (dataset: 2015-11-13-10-28-08). Left: NAS-LCD. Right: LCD.

Fig. 4. Visual images (a) and selected landmarks (b).

two different architectures with fine-tuning, the proposed NAS-optimized architecture and a hand-crafted architecture in [20], which are respectively denoted as “NAS-LCD” and “LCD”.

A. Dataset

The RobotCar dataset was used as the training data for the NAS task. The data were collected using the RobotCar platform repeatedly by traversing a route of approximately 10km in central Oxford, UK (see Fig. 3). To that end, we used the center camera (Fig. 4) of the vehicle-mounted Bumblebee front-faced camera (Point Grey Bumblebee XB3 trinocular

Fig. 5. NAS-optimized LCD architecture.

stereo camera). The image size was $(IW, IH)=1280 \times 960$. The camera was logged at the maximum frame rate of 16 Hz. We cropped the images with a fixed window $[100, IW-100] \times [0, IH-160]$ to remove a fixed bottom-side image region that is frequently affected by dynamic objects such as cars. We resized the images to 512×512 to be input to the cityscapes dataset. In the experiments, the following six image sequences from the RobotCar dataset were used: 2015-10-29-12-18-17 (“10/29”), 2015-09-02-10-37-32 (“9/2”), 2015-10-30-13-52-14 (“10/30”), 2015-11-10-10-32-52 (“11/10”), 2015-11-12-13-27-51 (“11/12”), and 2015-11-13-10-28-08 (“11/13”). Further, each sequence was $\times 4$ downsampled. The resultant sequences were 9387, 5413, 12049, 7338, 10368, and 10742 in length, respectively.

Trajectories are around 10km in length, and the 20m GPS-derived ground truth was used for evaluating VPR performance.

B. NAS Performance

In this section, we investigate NAS performance. The learning rate was fixed at 0.0004. The number of learning iterations was 3,600 per NAS generation. One GPU was used for the search, and the batch size was set to four. The NAS search was iterated for 500 generations. Notably, the NAS frequently proposes invalid P-lists, such as those that do not connect all the valid layers. Therefore, we checked whether a given network candidate was valid at each generation. Eliminating invalid network candidates resulted in up to 94% reduction in the total time cost for NAS. The total computation time was 50 h (1 GPU, Nvidia RTX 2080). The network with the highest score (for the 500 generations) was observed at 483 generations. This is a significant reduction in time cost compared with the naive NAS implementation, which takes approximately 20-30 GPU days.

Fig. 5 depicts the architecture detected by the proposed NAS-LCD. It is evident that the detected architecture is reasonably complex and considerably different from the hand-crafted one in [20].

C. Visual Place Recognition Performance

For the VPR, the top-1 accuracy was used. To improve robustness, each i -th query image is augmented by additional $(i \pm 2)$ -th images and the location with the highest votes is output as the final decision. The queries that provide no detection over the entire sequence were treated as false-negative predictions.

TABLE I
TOP-1 ACCURACY IN VPR

method	9/2	10/30	11/10	11/12	11/13
generic FPN	1.6	3.8	3.3	3.1	2.4
LCD	11.0	17.7	16.8	19.5	13.2
NAS-LCD	16.0	35.0	28.1	26.8	18.2

Table I presents the top-1 accuracy in VPR. It is evident that NAS-LCD outperformed LCD for all the dataset pairs considered. The difference exists because NAS-LCD is better than LCD in describing cropped OBB regions and capturing object-appearance distribution. In addition, Table I presents the VPR result with a generic FPN, which was pre-trained on the VOC2007 dataset for 20 object categories. It is evident that both the LCD and NAS-LCD methods, both of which were automatically fine-tuned on the target data, outperformed the generic FPN-based method.

In summary, the proposed SSDADA framework for mining effective collection of landmark examples and fine-tuning the network architecture and parameters is effective in significantly improving the VPR performance.

V. CONCLUSIONS

We proposed a lightweight and self-supervised method for long-term VPR. The proposed method avoids the costly annotation process by automatically mining landmark-object instances from the visual experience. The mined object instances are then used for designing an LCD. The proposed lightweight NAS algorithm is customized for LCD applications, and significantly reduces the architecture search and training costs. The experiments showed that the overall framework could learn effective LCD and yield very reasonable estimates of VPR. Importantly, the VPR performance was significantly improved using the proposed method.

REFERENCES

- [1] M. Cummins and P. Newman, “Fab-map: Probabilistic localization and mapping in the space of appearance,” *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.
- [2] M. J. Milford and G. F. Wyeth, “Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights,” in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 1643–1649.
- [3] E. Garcia-Fidalgo and A. Ortiz, “ibow-lcd: An appearance-based loop-closure detection approach using incremental bags of binary words,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3051–3057, 2018.
- [4] A. Chella, I. Macaluso, and L. Riano, “Automatic place detection and localization in autonomous robotics,” in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 741–746.
- [5] J. Straub, S. Hilsenbeck, G. Schroth, R. Huitl, A. Möller, and E. Steinbach, “Fast relocalization for visual odometry using binary features,” in *2013 IEEE International Conference on Image Processing*. IEEE, 2013, pp. 2548–2552.
- [6] G. Kim, B. Park, and A. Kim, “1-day learning, 1-year localization: Long-term lidar localization using scan context image,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1948–1955, 2019.
- [7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [8] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *arXiv preprint arXiv:1611.01578*, 2016.

- [9] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017.
- [10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [11] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018.
- [12] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3213–3223.
- [13] T. Naseer, G. L. Oliveira, T. Brox, and W. Burgard, "Semantics-aware visual localization under challenging perceptual conditions," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2614–2620.
- [14] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision," 1981.
- [15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.
- [16] Q. V. L. Golnaz Ghiasi, Tsung-Yi Lin, "Nas-fpn: learning scalable feature pyramid architecture for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 7036–7045.
- [17] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.
- [18] B. Chen, G. Ghiasi, H. Liu, T.-Y. Lin, D. Kalenichenko, H. Adams, and Q. V. Le, "Mnasfpn: Learning latency-aware pyramid architecture for object detection on mobile devices," *arXiv preprint arXiv:1912.01106*, 2019.
- [19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results."
- [20] T. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 936–944.