EdgeVisionBench
A Benchmark of Evolving Input Domains for Vision Applications at Edge

Zhang, Qinglong; Han, Rui; Liu, Chi Harold; Wang, Guoren; Chen, Lydia Y.

DOI
10.1109/ICDE55515.2023.00288

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 IEEE 39th International Conference on Data Engineering, ICDE 2023

EdgeVisionBench: A Benchmark of Evolving Input Domains for Vision Applications at Edge

Qinglong Zhang, Rui Han, Chi Harold Liu, Guoren Wang
Beijing Institute of Technology, Beijing, China
Beijing, China
{1120181200, hanrui, chiliu, wanggr}@bit.edu.cn

Lydia Y. Chen
TU Delft
Delft, Netherlands
lydiaychen@ieee.org

2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00288

***Abstract*—Vision applications powered by deep neural networks (DNNs) are widely deployed on edge devices and solve the learning tasks of incoming data streams whose class label and input feature continuously evolve, known as domain shift. Despite its prominent presence in real-world edge scenarios, existing benchmarks used by domain adaptation methods overlook evolving domains and under represent their shifts in label and feature distributions. To address this gap, we present EdgeVisionBench, a benchmark seeking to generate evolving domains of various types and reflect their realistic label and feature shifts encountered by edge-based vision applications. To facilitate evaluating domain adaptation methods on edge devices, we provide an open-source package that automates workload generation, contains popular DNN models and compression techniques, and standardizes evaluations with interactive interfaces. Code and datasets are available at https://github.com/LINC-BIT/EdgeVisionBench.**

***Index Terms*—Edge computing, vision applications, evolving domains, benchmark**

I. Introduction

The recent success of edge computing presents a new paradigm which features on-premise data processing on edge devices and provides the advantage of low response latency and privacy preservation in local data processing [15]. Thus, deep neural networks (DNN) based vision applications, ranging from image recognition [10], object detection [1] to semantic segmentation [12], are increasingly deployed on edge devices. Such applications solve learning tasks for incoming data streams, which may exhibit different distributions over time, known as **distribution/domain shifts**, and calls for adapting DNN models. However, as edge devices are limited in computation/storage resources, the on-device pre-trained DNN models are typically small/compressed and face the disadvantage of accuracy degradation [7]. It is exceedingly difficult to solve tasks encountering domain shifts, i.e., adapting the models of shallow architectures and small weights from the source domain to continuously changing label and feature distributions - **evolving target domains**.

Example of domain shift. This paper considers domain shifts in both *label* (y) and *input feature* (x). Figures 1(a) to (d) illustrate four scenarios of label distribution shift [6]. (I) *Close-set domain shift* where both the source and target domains have the same label space. (II) *Partial domain shift* where the target domain only contains a part of labels (e.g. car in Figure 1(b)) in the source domain. (III) *Open-set domain shift* where the target domain has unknown labels (e.g. rider in Figure 1(c)) in the source domain. (IV) *Universal domain shift* covers label shifts of both partial and open-set shifts. In each scenario, the target domain may have considerably different input features (e.g. light illumination, weather, and object densities) with the source domain.

Challenge and motivation. Despite the ubiquity of domain shift in real-world scenarios (e.g. out-of-distribution generalization [9] and detection [13]), the edge-based evolving domains are under represented in the benchmarks [4], [5], [8], [11], [14] that are widely used by the domain adaptation methods. Most of these benchmarks were designed for one **stationary** target domain. For instance, the adaptation methods developed for partial, open-set and universal domain shifts only study one prespecified target domain. Moreover, the online/test-time adaptation methods developed for evolving domains [6] just consider close-set domain shifts and a few datasets with synthetic transformations of input feature. While existing benchmarks provide the fundamental building blocks for studying domain adaptation, they fall short in reflecting and addressing the real edge scenarios - evolving domain shifts. For instance, an adaptation method that works well in the close-set domain shift can even consistently harm robustness for open-set and universal shifts [6]. To ensure the robustness of model transferring across shifts, the very first step is to design benchmarks that capture various shift types while supporting evolving domain generation at edge.

In this paper, we present EdgeVisionBenchmark, a benchmark seeking to generate four label shifts and various synthetic/realistic feature shifts, thus representing a broad array of evolving domains for vision applications. These DNN-based vision applications include: image recognition [10], object detection [1], semantic segmentation and action recognition [12]. The generated domains reflect natural distribution shifts arising from real-world scenarios on mobile and edge devices.

Contributions. We make following key contributions with our EdgeVisionBenchmark:

- **Extensive set of evolving domains**. A collection of the most prevalent vision datasets that can be used to generate evolving domains of any shift type. We expect to facilitate the

This work is supported by the National Natural Science Foundation of China (Grant No. 62272046, 62132019, 61872337) and Shandong Provincial Natural Science Foundation (Grant No. ZR2020MF034). Corresponding author: Rui Han.

2375-026X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDE55515.2023.00288

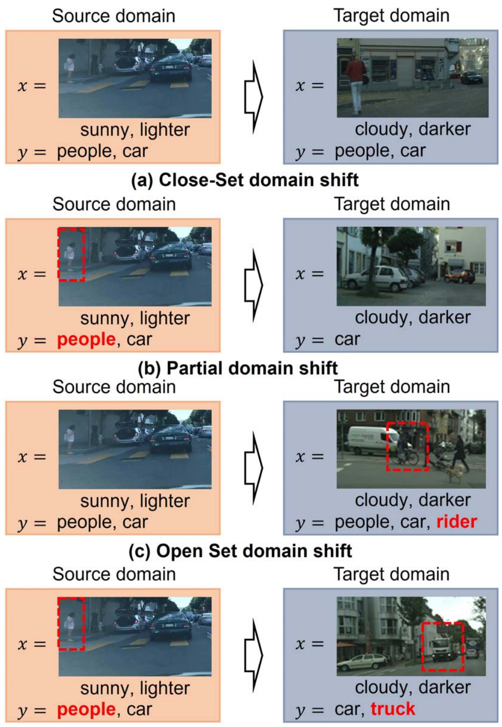


Fig. 1. Some example domain adaptation (DA) scenario to demonstrate the evolving domains with different types of label shifts. For a domain, we show a sample x with its domain information below and a label y representing appeared objects in x. Red font and dashed boxes illustrate novel categories of objects (unknown objects) which cause label shifts.

development of model adaptation techniques on challenging edge scenarios, e.g., drastic and continuously changing input data streams.

- **Edge testbed for on-device training**. An accompanying testbed that comprehensively considers different factors in testing a DNN model on edge devices, including workloads of diverse labels and input features, popular DNN models and compression techniques, and the latest domain adaptation methods. We further provide an API to the testbed for developing new adaption algorithms.
- **Demonstration scenarios**. Based on the generated workloads, we provide several evaluation scenarios of how adaption methods perform under different types of evolving domains, DNN models, compression techniques, and heterogeneous edge devices.

II. Design of EdgeVisionBench

EdgeVisionBench is designed to generate evolving domains for edge-based vision applications with four steps. As shown in Figure 2, these steps are implemented based on EdgeVisionBench's modules belonging to three layers: user interface, workload generation, and dataset repository.

TABLE I
Summary of collected datasets

Edge vision application	# Datasets (basic / adaptation)	# Samples (min to max)	# Categories (min to max)	Image resolution (min to max)
Image Classification	33 / 8	7.4k to 1281k	10 to 2000	32×32 to 224×224
Object Detection	40 / 3	0.5k to 279k	1 to 344	100×100 to 2464×2056
Semantic Segmentation	10 / 2	0.7k to 226k	1 to 66	100×100 to 2048×1024
Action Recognition	10 / 0	0.1k to 1000k	12 to 1000	320×240 to 1920×1080

A. Dataset preparation

Dataset collection. In EdgeVisionBench, we collect over 100 datasets belonging to four typical types of edge vision applications: (1) *image classification* applications aim to recognize the category of an image; (2) *object detection* applications aim to detect the category and location of each object in an image; (3) *semantic segmentation* applications aim to recognize the category of each pixel in an image; and (4) *action recognition* applications aim to recognize the category of an action in a video clip. As listed in Table I, these datasets have diverse numbers of samples, categories, and image resolutions.

Dataset preparation. The collected datasets can be divided two categories. First, all *basic* datasets denote real-world vision scenarios and they only have one domain. Second, *adaptation* datasets are derived from existing domain adaptation benchmarks and have multiple domains (most domains are synthetic). In preparation, each adaptation dataset is splitted into multiple sub-datasets, each one only contains samples from one domain.

B. Evolving domain construction

This step constructs evolving target domains by first generating multiple source-target domain pairs, where each pair corresponds to one randomly selected source and target datasets that have an overlap in their label space. Subsequently, it merges all source domains as one domain and randomizes the arrival sequence of the target domains. The constructed evolving target domains have shifts in both label and feature distributions. Specifically, given two original source and target datasets in the ith source-target domain, Figure 3 illustrates the process of constructing four *label shifts*. (1) Close-set domain shift: the construction operation is to remove all the labels that are not belong to either the source or the target dataset. (2) Partial domain shift: the operation is to remove the target dataset's labels not belonging to the source dataset. (3) Open-set domain shift: the first operation is to remove the labels that only belong to the source dataset and the second operation is to group all the unknown labels of the target dataset into one category. The second operation is also used in the construction of universal domain shift. Note that two different datasets naturally have considerable shifts in their feature distributions, so this step unifies the size/resolution of images in the datasets.

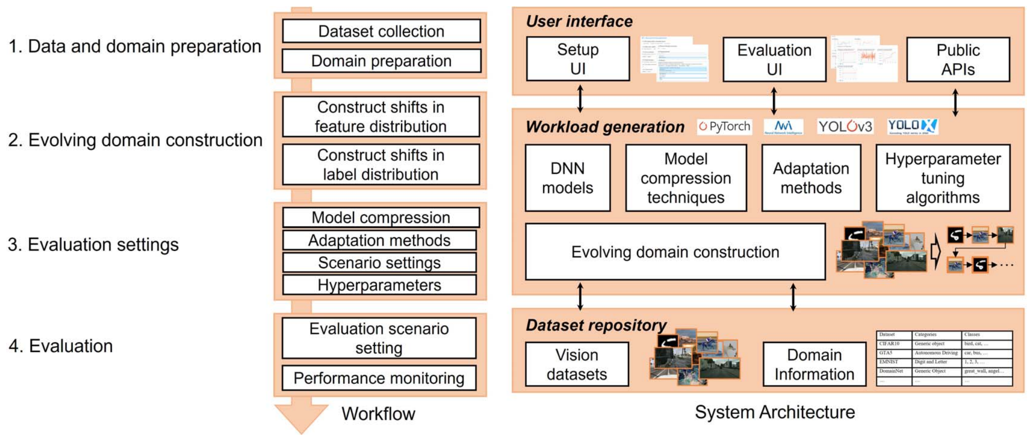


Fig. 2. The architecture and workflow of EdgeVisionBench

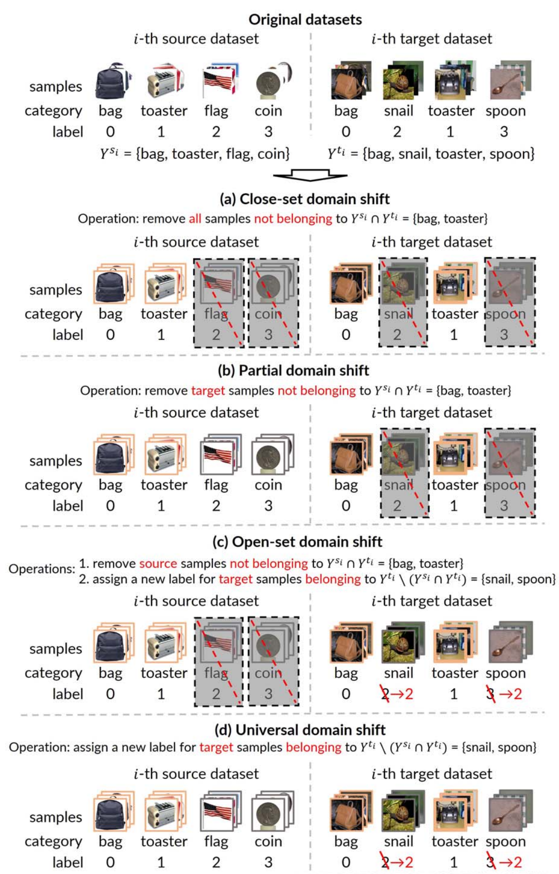


Fig. 3. Operations on source and target datasets to construct label shifts.

C. Evaluation settings

This step provides a list of optional configurations to evaluate how domain adaptation methods deal with evolving domain shifts.

- *DNN models and model compression techniques.* EdgeVisionBench supports common DNN models in vision applications and their compression techniques such as structured model pruning.
- *Domain adaptation methods.* EdgeVisionBench provides an abstract class to define the basic logic of an adaptation method.
- *Scenario settings.* These settings determine the characteristic of evolving target domains, such as the number of target domains and the period of domain occurrence.
- *Hyperparameters.* EdgeVisionBench applies validation image corruptions [3] on source datasets to generate the validation target domains. It then conducts random hyperparameter search [2] or user-defined searching algorithm to find the best hyperparameters for each workload.

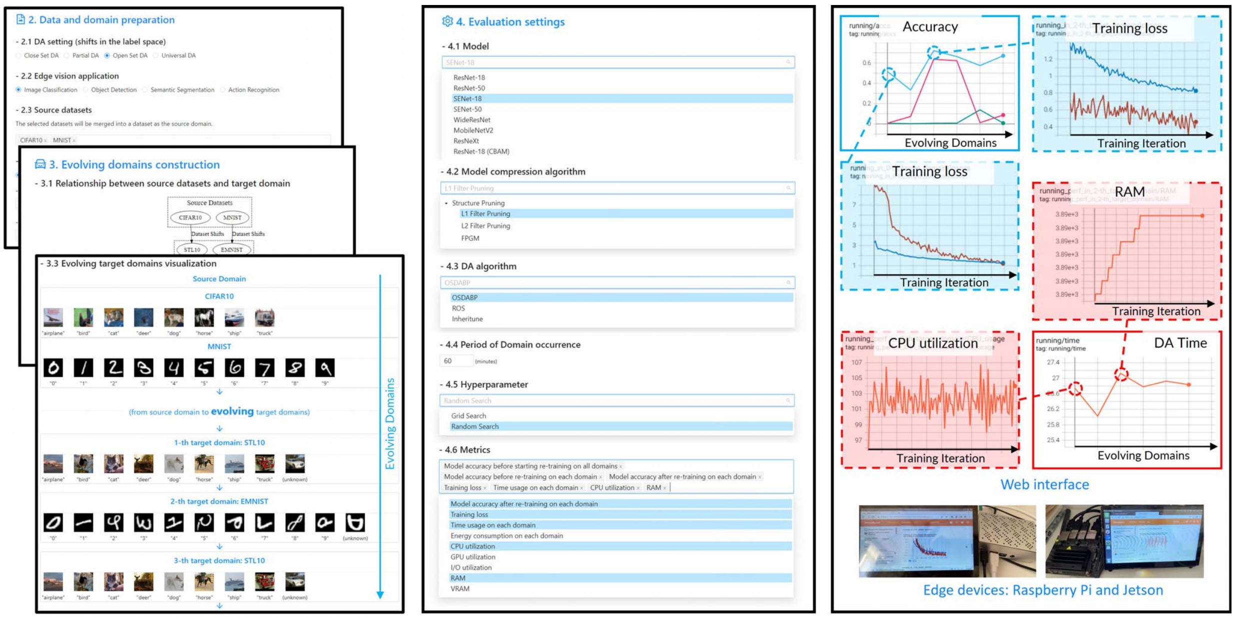
D. Evaluation

This step allows users to interact with the web interface to monitor the performance of the domain adaptation process on edge devices. Specifically, once a new target domain arrives, a domain adaptation method is applied to re-train the mode until the convergence. During iterative training, various metrics are monitored and displayed, including: (i) accuracy-related metrics in each target dataset: the model accuracy and training losses; (ii) performance-related metrics on edge devices: the time usage, energy consumption, CPU/GPU utilization, I/O utilization, and memory usage; and (iii) a summary report is presented when one evaluation completes.

III. Demo Description

In this demo, we plan to allow the audience to freely explore the example workloads via our web interface and experience how EdgeVisionBench can facilitate domain adaptation researchers to construct evaluation experiments. Figure 4 demonstrates the major steps in this evaluation process.

Evolving domains construction. The user begins by determining the basic settings of evolving domains including the type of label shift, the datasets in the source domain, and the target domains' arrival order. Subsequently, EdgeVisionBench



(a) Evolving domains construction (b) Evaluation settings (c) Monitoring of domain adaptation process

Fig. 4. Demonstration of EdgeVisionBench

loads and processes relevant datasets to construct the target domains, and feeds back them through the web interface. As shown in the bottom of Figure 4(a), users can examine the corresponding datasets of each target domain, easily see the shifts in input features and labels, and preview the sequence of target domains.

Evaluation settings. Specifically, users have the opportunity to: (1) determine the DNN model and its compression technique in producing the compressed model on edge devices; (2) choose the domain adaptation method and its corresponding hyperparameter tuning algorithm in model re-training; (3) set the frequency of domain occurrence to decide the model adaptation time in each domain; and (4) select evaluation metrics of interest.

Monitoring of adaptation process. The performance monitoring module of EdgeVisionBench is developed based on Tensorboard and it automatically tracks pre-specified metrics in evaluation. As shown in Figure 4(c), the web interface uses one subgraph to illustrate one evaluation metric such as training loss or CPU utilization. After the evaluation, the summary report is saved in readable JSON format so that the user can quickly examine the evaluation results. The monitor causes negligible overheads on edge devices such as Raspberry Pi or Jetson.

Demonstration scenarios. To ease the navigation of demo for the audience, we provide a list of example scenarios in the website: (i) image classification on Raspberry Pi and Jetson Nano; (ii) object detection on Jetson TX2; (iii) semantic segmentation on Jetson Xavier NX; and (iv) action recognition on Jetson AGX Orin.

References

[1] C. Wu et al. Machine learning at facebook: Understanding inference at the edge. In *HPCA'19*, pages 331–344, 2019.

[2] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *ICLR'21*. OpenReview.net, 2021.

[3] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *ICLR'19*, 2019.

[4] Bo Fu Mingsheng Long Junguang Jiang, Baixu Chen. Transfer-learning-library. https://github.com/thuml/Transfer-Learning-Library, 2020.

[5] Pang Wei et al. Koh. WILDS: A benchmark of in-the-wild distribution shifts. In *ICML'21*, 2021.

[6] Xiaofeng Liu, Chae Hwa Yoo, Fangxu Xing, Hyejin Oh, Georges El Fakhri, Je-Won Kang, and Jonghye Woo. Deep unsupervised domain adaptation: A review of recent advances and perspectives. *CoRR'22*, abs/2208.07422, 2022.

[7] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE TKDE*, 22(10):1345–1359, 2009.

[8] Steffen Schneider, Alexander S. Ecker, Jakob H. Macke, and Matthias Bethge. Salad: A toolbox for semi-supervised adaptive learning across domains, 2018.

[9] Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *ICLR'15*, 2015.

[11] Anne-Marie Tousch and Christophe Renaudin. (yet) another domain adaptation library, 2020.

[12] Zuxuan Wu, Xin Wang, Joseph E Gonzalez, Tom Goldstein, and Larry S Davis. Ace: Adapting to changing environments for semantic segmentation. In *CVPR'19*, pages 2121–2130, 2019.

[13] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[14] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Dassl.pytorch. https://github.com/KaiyangZhou/Dassl.pytorch, 2021.

[15] Zhi Zhou, Xu Chen, En Li, Liekang Zeng, Ke Luo, and Junshan Zhang. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.