Demo Abstract: Online Training and Inference for On-Device Monocular Depth Estimation

Allen-Jasmin Farcas, Geffen Cooper, Hyun Joon Song, Afnan Mir, Vincent Liew, Chloe Tang,
Prithvi Senthilkumar, Tiani Chen-Troester, Radu Marculescu
The University of Texas at Austin
Email: {allen.farcas, geffen, radum}@utexas.edu

***Abstract*—A central challenge in machine learning deployment is maintaining accurate and updated models as the deployment environment changes over time. We present a hardware/software framework for simultaneous training and inference for monocular depth estimation on edge devices. Our proposed framework can be used as a hardware/software co-design tool that enables continual and online federated learning on edge devices. Our results show real-time training and inference performance, demonstrating the feasibility of online learning on edge devices.**

***Index Terms*—Federated Learning, Continual Learning, Hardware/Software Co-Design, Edge Devices, Internet-of-Things**

I. Introduction

Training and deploying models in the real world is the de facto solution for enabling machine learning (ML) on edge devices. Currently, the common approach is to train a model on a cloud server, then optimize and deploy it on edge devices. However, deployed models are often static, resulting in degraded performance as the environment changes.

Continual learning (CL) can extend the learning process at the deployment stage, thus enabling models to adapt continually to changing conditions or tasks. A central challenge in CL is how to learn new tasks without forgetting and having a lower performance on older tasks. Practical approaches to CL must consider the memory and computational overhead involved in data, parameter storage, and model updates. However, the deployment of CL on edge devices is limited [1] and even recent surveys on CL [2] lack coverage of hardware-aware approaches with actual hardware validation on state-of-the-art CL methods. Furthermore, realistic data streams for real-time sensing, *e.g.,* a robot moving in indoor/outdoor space, are not thoroughly explored in existing approaches [2].

With growing privacy concerns, federated learning (FL) [3] has emerged as the de facto approach to enable on-device training on local data. However, the most common scenario is to run on-device training when the impact on user experience is minimal, usually at night while the device charges and is connected to Wi-Fi. Thus, in [4], the authors propose joint training and inference optimization for FL while providing *model as a service*, with a focus on maximizing inference performance. Since model as a service is a cloud-based framework, their approach requires constant internet connection. However, on-device training and inference (OTI) still represents a challenge.

In order to enable realistic continual FL, there is a clear need for developing OTI frameworks. Therefore, in this paper, we propose an OTI framework for monocular depth estimation (MDE) and validate it using real edge devices.

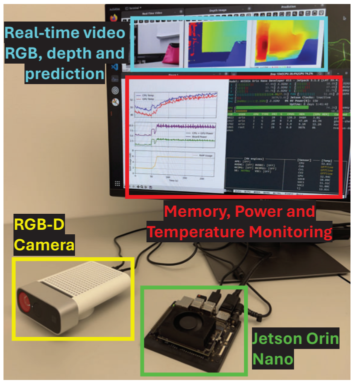
Real-time video RGB, depth and prediction
Memory, Power and Temperature Monitoring
RGB-D Camera
Jetson Orin Nano

Fig. 1: Online training and inference for on-device monocular depth estimation. We show the RGB-D camera and Jetson Orin Nano training. We monitor memory, power and temperature variation, while running real-time depth estimation inference and simultaneously training the model on current data.

II. Approach

Hardware Prototype Considerations: Our hardware prototype, shown in Fig. 1, consists of an NVIDIA Jetson Orin Nano edge device with 8GB RAM and a Microsoft Azure Kinect Developer Kit. We continuously monitor the memory consumption, power consumption and temperature of the edge device. For empirical evaluation, we show the real-time video feed and the depth from the RGB-D camera and the real-time predictions of the MDE model.

Software Framework for Online Training and Inference: We depict in Fig. 2 the practical approach towards OTI. First, we get the RGB-D image from the camera and preprocess the RGB image and the depth map for the model (*i.e.,* both inference and training). This includes inpainting to fill in any missing depth values. Next, we temporarily store on a data buffer (with a fixed size) the RGB and depth images. We use the data buffer as a blocking mechanism, *i.e.,* queue, enabling synchronized data exchange between the inference thread and training thread. Naturally, we limit the size of the data buffer due to the limited on-device memory. We use the entire buffer to create a data loader that is used to train the MDE model.

979-8-3503-7025-6/24/$31.00 ©2024 IEEE
DOI 10.1109/IoTDI61053.2024.00026

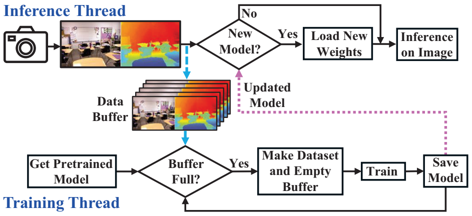

Fig. 2: Overall workflow of our software framework. We use one thread for inference and one thread for training. Both threads use the data buffer to temporarily store images that are used for training when the buffer is full.

TABLE I: GuideDepth quantitative performance on the NYUv2 dataset. Overall, structural pruning and fine-tuning are required to improve the model performance. We show metrics where lower is better (↓) and higher is better (↑).

	Full	30% Pruned w/o Fine-Tuning	30% Pruned w/ Fine-Tuning
Abs Rel ↓	–	2.406	0.145
Sq Rel ↓	–	13.354	0.105
RMSE ↓	0.501	5.187	0.599
RMSE *log* ↓	0.058	1.211	0.202
δ_1 ↑	0.823	0.015	0.796
δ_2 ↑	0.961	0.045	0.958
δ_3 ↑	0.990	0.121	0.990

Finally, the updated model is saved on-device and the inference thread loads the new model weights.

Optimizing for Online Training and Inference: In general, MDE models are quite large and heavily optimized for inference only. However, for OTI, we have to ensure that both inference and training are co-optimized. To this end, we use structured pruning [5] to keep the performance in terms of prediction accuracy, but optimize for on-device OTI. The structured pruning simultaneously reduces the model size (in terms of number of parameters), the memory consumption during training, and the computational complexity.

The continuous stream of RGB images and ground truth depth maps enable continual updating of the model at runtime. However, if the images and depth maps in the buffer only contain the most recently captured data, this can result in overfitting and catastrophic forgetting. Thus, a CL approach is required; we experiment with a simple replay-buffer [6] which adds a set of *previously* collected images and depth maps to diversify the data in the buffer and help mitigate forgetting.

III. Experimental Setup and Results

For all experiments, we use a state-of-the-art efficient and lightweight MDE model, namely GuideDepth [7]. We use the GuideDepth trained on 240×320 resolution images from NYUv2 dataset as a baseline. For OTI, we structurally prune 30% of all layers except the first four convolutional layers using the Torch-pruning library [5] with L1 norm importance. We then fine-tune the pruned model for 1 epoch on the NYUv2 dataset. We use a maximum size of 8 for the data buffer; therefore, the batch size for training is also 8.

TABLE II: Number of Parameters, FLOPs, MACs and size on disk for Full GuideDepth and Pruned GuideDepth. Overall, structured pruning reduces the memory consumption and computational requirements of the model.

	Params	GFLOPs	GMACs	Size on disk [MB]
Full	5.8M	5.32	2.66	22.9
Pruned (30%)	3.5M	3.74	1.87	13.7

TABLE III: Average latency, power and memory consumption during inference, training, and OTI for full and optimized (Opt.) GuideDepth. We show the average latency of OTI for both inference and training, separated by forward slash (/). Overall, OTI runs both inference and training in real-time.

	Avg. Latency [ms]	Avg. Power [W]	Avg. Memory [GB]
Inference (Full)	33.76±1.8	8.5	3.3
Train (Full)	521.07±13.23	11.6	4.6
OTI (Full)	**46.49±46.61 / 657.68±194.24**	**10.2**	**5**
Inference (Opt.)	34.07±1.37	8.3	3.2
Train (Opt.)	502.34±8.94	11.6	4.3
OTI (Opt.)	**45.88±43.26 / 617.69±84.37**	**9.8**	**4.6**

In Table I we use the evaluation metrics from [7] to evaluate the performance on the NYUv2 dataset. As shown in Table II, through structured pruning, we drastically reduce the number of parameters and consequently the size of the model on disk, FLOPs and MACs. Finally, Table III shows our proposed OTI framework consuming less power compared to continuously training either the full or optimized model (i.e., 30% pruned with fine-tuning), while running faster for the optimized model. Our demo shows that OTI improves performance over time. However, for larger performance leaps in latency, power and memory consumption, there is a clear need for more advanced OTI approaches.

IV. Conclusion

In this demo, we have proposed an online training and inference framework for monocular depth estimation. Our prototype demonstrates the feasibility of online learning on edge devices with live streaming data. We hope this will stimulate new research in on-device continual learning.

Acknowledgment

This research was supported by seed funding provided by the Chandra Family Department of Electrical and Computer Engineering at the University of Texas at Austin.

References

[1] Y. D. Kwon *et al.*, "Exploring system performance of continual learning for mobile and embedded sensing applications," in *2021 IEEE/ACM SEC*, pp. 319–332, 2021.

[2] L. Wang *et al.*, "A comprehensive survey of continual learning: Theory, method and application," *IEEE TPAMI*, 2024.

[3] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Aistats*, pp. 1273–1282, PMLR, 2017.

[4] P. Han *et al.*, "Federated learning while providing model as a service: Joint training and inference optimization," *arXiv preprint arXiv:2312.12863*, 2023.

[5] G. Fang *et al.*, "Depgraph: Towards any structural pruning," in *Proceedings of the IEEE/CVF CVPR*, pp. 16091–16101, 2023.

[6] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," *NEURIPS*, vol. 32, 2019.

[7] M. Rudolph *et al.*, "Lightweight monocular depth estimation through guided decoding," in *2022 ICRA*, pp. 2344–2350, IEEE, 2022.