2021 IEEE International Symposium on Circuits and Systems (ISCAS) | 978-1-7281-9201-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAS51556.2021.9401468

Online Training Refinement Network and Architecture Design for Stereo Matching

Yu-Sheng Wu*, Sih-Sian Wu*, Tan Huang, Liang-Gee Chen, *Fellow, IEEE*
DSPIC Lab, Department of Electrical Engineering
National Taiwan University, Taiwan
gingerwu10@gmail.com, benwu@video.ee.ntu.edu.tw, danhuang0313@gmail.com, lgchen@ntu.edu.tw

***Abstract*—Sending local data to cloud servers is vulnerable to user privacy, and its long update latency. Meanwhile, the state-of-the-art stereo matching method is still computation demanding, fine-tuning the whole model on-device is not a practicable solution because of the limited power budget and computation ability on edge devices. In this study, we propose a two-stage online stereo matching refinement system, using an additional light-weight network to learn the domain gap between local data and cloud training data. We define a load-gain ratio to evaluate computer efficiency. This refinement system has a much better load-gain ratio than fine-tune. (0.2 v.s. 35.7 operation overhead/accuracy gain) Nevertheless, we only disburse 0.2% of additional parameters and 0.7% additional computation as set by inference the stereo matching model. Thus, it would be a suitable choice for an online training scenario. With re-scheduling the training pipeline, we use a patch-based layer fusion technique and reduce the off-chip memory bandwidth by 97%.**

***Index Terms*—stereo matching, online training, refinement network, device personalization, layer fusion, patch-based layer fusion**

I. Introduction

Depth estimation is the fundamental element of 3D vision. In computer vision tasks, such as autonomous driving, robotics, and AR, rely on dense and reliable 3D reconstructions of the sensed environment. With groundbreaking progress made by deep learning [1], current state-of-the-art (SOTA) stereo matching methods rely on deep convolutional neural networks (CNNs) taking a pair of left-right images to estimate the dense disparity map. Meanwhile, the necessity of additional training of deep neural networks (DNN) on edge device applications has increased due to the performance degradation when applying DNN models on the device. Different data distribution than the training one and approximated implementation degrades the performance when applying DNN-based stereo engines in real-world cases.

To enhance the performance in local scenarios, additional training with data received from the device can improve the inference performance by tuning the model to fit into its personal environment. **Device personalization** or **insitu personalization** is an essential function for IoT systems. Previous methodologies prefer off-line training by sending personal data to the cloud server and receiving updated models consistently, only provide inference mode in dedicated accelerators. However, sending data to cloud servers can have vulnerabilities which is a privacy concern for users, and its long update latency can be a severe weakness for real-time usage on edge devices. As a consequence, realize on-device DNN training has become an attractive solution to such problems.

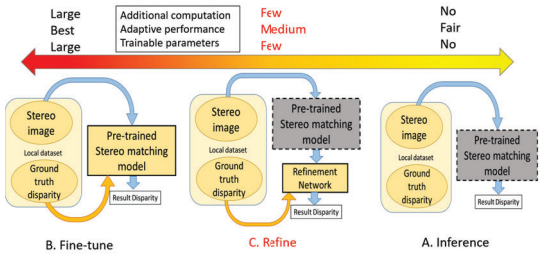


Fig. 1: Orientation of this work. We proposed an additional light-weight network to learn the domain gap without retraining the whole stereo matching model.

Due to the limited energy budget and computation ability on an edge device, retraining the whole stereo matching model on-chip is not a reasonable solution to realize device personalization within depth estimation tasks. For instance, retraining PSMNet [2] with full-HD images at 24 fps scenario, need processing about 548 T of operations per second. In current accelerators of machine learning, [3] got 3.6 TOPS and [4] got 0.3 TOPS peak performance, about 3 orders away from the required performance.

The proposed system in this study has orientation as shown in Fig. 1. We can organize edge stereo matching system form three aspects including **additional computation**, **adaptive performance**,and **trainable parameters**. The rightmost position of the spectrum would be **Inference**, we inference pre-trained stereo matching model without additional computation and trainable parameters but has the least adaptive ability. In the contrast, the leftmost position of the spectrum would be **Fine-tune**, the whole stereo matching model is trainable and has the best adaptive performance, but the computation overhead is large. Between these two positions of the spectrum would be **Refine**. We fixed the pre-trained stereo matching model and attached an additional light-weight network to learn the domain gap between local data and cloud training data.

In summary, the contribution of this work is twofold:

1) A two-stage online training refinement system for stereo

*Joint first author

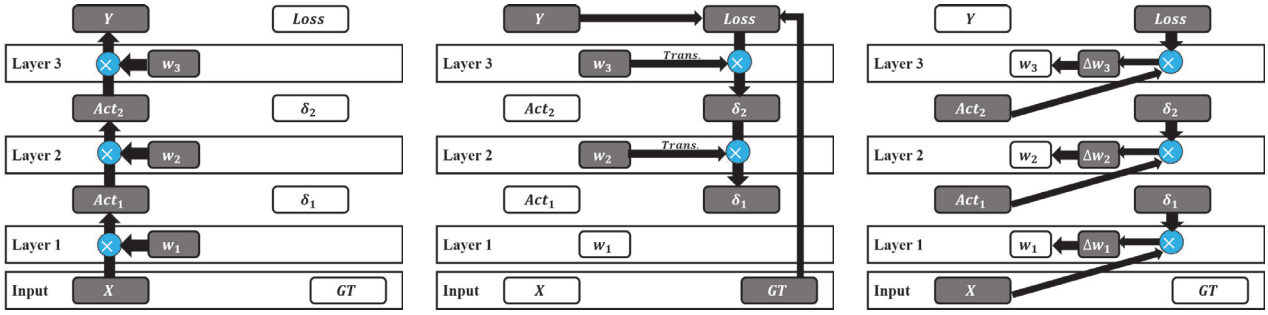


Fig. 2: Overview of training procedure of (a) feed forward propagation (b) back propagation and (c) weight gradient update.

matching is proposed with only 0.2 % additional parameter and 0.7 % computation compared to the inference stage.

2) The proposed architecture achieves 97 % bandwidth reduction compared to direct implementation.

II. Related Works

A typical way of training a deep learning model includes three computations. **1) Feed forward propagation (FF)**; **2) back propagation (BP)**; and **3) weight gradient update (WG)**. Fig. 2 shows the overview training procedure with a simple network.

WG has different dataflow from FF/BP, thus unified PE array may need additional control modules or having performance degradation, but a separate PE array may sacrifice flexibility. Bit precision affects the algorithm performance and we expect to find a proper bit width without accuracy loss. Also, these works simulate their result on a classification task, which has different network characteristics from the stereo matching task. The algorithm differences between training and inference lead to different requirements of systems and hardware architecture.

First, BP and WG operations are an additional process for training. WG phase needs input to generate from the FF phase. Besides, the WG phase has quite a different computation flow with FF/BP phase. The data dependency with more intermediate data and the irregular computation flow increase more consideration in training hardware design. Second, training usually proceeds in waves of the mini-batches, a set of inputs grouped to processed in parallel, which makes the memory footprints much higher because of the limited on-chip buffer. The above reasons implement a hardware accelerator that supports training a more difficult task than inference.

Yuan *et al.* [5] support multi-level sparsity convolution PE arrays, additional online tuning PE for FC layers bring out the pioneer hardware accelerator that tried to put learning feature on-chip. Fleischer *et al.* [6] proposed a processor core supporting full training and inference in the system. Lee *et al.* [4] present a energy-efficient on-chip learning accelerator. By exploiting sparsity inactivation, the sparsity-aware PE array gets higher throughput. Also, the input load balancer resolves PE utilization degradation due to sparsity-aware dataflow. Lu *et al.* [7] re-arrange the saving weight and reduce the processing latency. Also, they try to mix max-pooling into the ReLU module and reduce further memory requirements. Choi *et al.* [8], [9] design the hardware friendly lossless training procedure, applying the suitable batch normalization method with local maximum quantization, and the bit flexible PE array supporting different precision data computation.

III. Proposed Refinement Network

A. Proposed Hardware Friendly Refinement Network

The proposed network is composed of two stages, as shown in Fig. 3. The first stage is the pre-trained stereo matching model which provides the original disparity prediction, and the second stage is the refinement network which learns the residual signals from local dataset. The first stage network takes the stereo pair I_L and I_R as input and produces the initial disparity. The SOTA stereo matching model integrates whole stereo matching process, hence the network can generate nearly optimal disparities results on the target domain. Previous work [10] has shown that cascading additional networks on the initial disparity prediction network, learning multi-scale residual signals can get further improvement on trained dataset. The summation of the residual and the initial disparity is considered as the refined disparity map.

To train the proposed refinement system, we adopt the smooth L_1 loss function, which is applied in [2]. Smooth L_1 loss is widely used in regression for stereo matching because of its robustness and low sensitivity to outliers, as compared to L_2 loss. The loss function is defined as

$$L(d, \hat{d}) = \frac{1}{N} \sum_{i=1}^{N} smooth_{L_1}(d_i - \hat{d}_i)$$

in which

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise, \end{cases}$$

where N is the number of labeled pixels, d_i and $\hat{d}_i$ are the ground truth and predicted disparity of the pixel i, respectively.

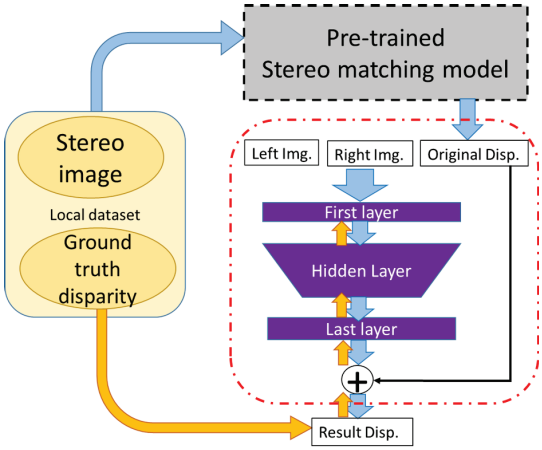


Fig. 3: Overview of the proposed framework. The propose light-weight refine network is blocked by red dashed lines.

The loss function is differentiable, hence the network can be trained end-to-end.

B. Proposed Hardware Architecture of Refinement Network

The direct implementation of the refinement network with structure of LNPU [4] is shown in Fig. 4 (a). The bandwidth requirement is 36.73 GB/s, and overall computation is 1.62 TFLOPs. DRAM access bandwidth is a relatively critical bottleneck with about 2 orders difference. Hence, the proposed architecture focuses on reducing DRAM access bandwidth.

1) Heterogeneous Layer Fusion: The layer fusion [12] is a widely adopted technique to reduce off-chip memory bandwidth, which concatenates operations of adjacent layers. The output activation of the previous layer is the input of the next layer, which is stored in on-chip memory instead of off-chip memory. The mini-batch serialization (MBS) CNN training approach [11] significantly reduces memory traffic by partially serializing mini-batch processing across groups of layers.

The timing pipeline after applying MBS is shown as Fig. 4 (b). The processing stage of one input sample in the FF phase is separated from the BP/WG phase. We explore the inter-layer reusable data between adjacent layers, which are indicated by the red dash line. We save the memory footprint of these intermediate data by separating the computation into blocks size, each block can directly compute through multiple layers in the FF phase with activation Act brought on-chip. However, in the BP and WG phase, we iterative process the block result through multiple layers by reusing the activation gradient δ on-chip.

However, the activations Act generated in the FF phase still need to be stored off-chip DRAM for data reuse and then reloaded in the WG phase since there is a long data reuse distance and there isn't enough on-chip buffer to keep the whole result.

To eliminate the long data reuse distance between the FF phase and the WG phase, we proposed a re-scheduled dataflow, patch-based layer-fusion training procedure. The core concept is that we merge three training phases into a cycle, and split the activation sample into further smaller partition – patch. Unlike the MBS method which illustrates in the last paragraph, each patch will not only finish processing multi-layer FF computation, but also multi-layer BP and WG computation at the same time. This approach will finish one patch's complete training procedure before doing the next patch's procedure, and the timing pipeline is referred to as Fig. 4 (c).

The benefit of this kind of dataflow is that we can directly reuse the generated activation and get the partial weight gradient by temporally buffering the reusable activation until the patch procedure is done. The split patch size is smaller than the block size in the aforementioned paragraph since we need to keep more intermediate data simultaneously, including the patch pyramid of activations, patch pyramid of activations gradients, the partial sum of weight gradients, and weights.

IV. Experimental Results

A. Performance Evaluation of Proposed Network

We choose PSMNet [2] fine-tuned network on KITTI 2015 [13] as our pre-trained network, and we choose different video sequences in DrivingStereo [14] as our target dataset. The overall performance on the different datasets is shown as Table I and visualization of one frame is shown in Fig. 5. This refinement system has a much better load–gain ratio k than finetune. For a flexible way to enhance the overall system performance, we can collect the failure examples and use idle time to fine-tune.

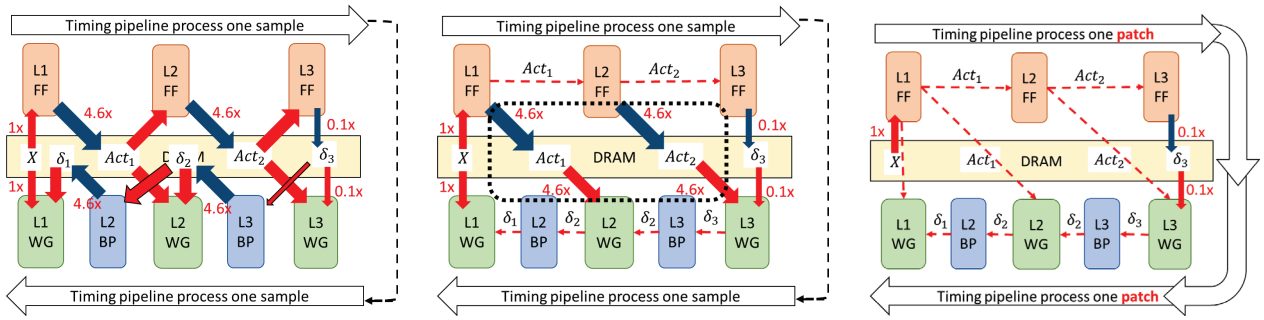
TABLE I: The performance on different dataset

	Accuracy			
	Foggy	Sunny	Rainy	Cloudy
Inference	94.2	95.3	92.5	97.3
Finetune	98.9	97.9	98.1	98.3
Refine (Proposed)	96.4	96.3	95.9	97.55
	$k = Op.\ Overhead/Acc.\ Gain$			
$k_{Finetune}$	42.5	76.9	35.7	200
k_{Refine}	0.3	0.7	0.2	2.8

Fig. 6 shows the comparison between the different simulation modes. The inference system has 5.48 M parameters and the refinement system has 5.49M parameters with 0.2% parameter overhead. In operation, processing 1 full-HD sample, we got 9.28 TFLOPs in inference mode, 27.8 TFLOPs in finetuning mode with 200% operation overhead. In the contrast, we got only 9.35 TFLOPs in refine mode with 0.7% operation overhead.

B. Ablation Studies for Network Configuration

We also tested different layer sizes and multi-scale techniques. Fig. 7 (a), shows the results of testing different hidden layer numbers and we found that the baseline network has saturated performance and thus it has the best cost-performance ratio. In Fig. 7 (b) and (c), although we can get better performance of disparities and receptive field with



(a) (b) (c)

Fig. 4: Timing pipeline of refinement network applying (a)LNPU [4] architecture, (b) MBS [11] architecture and (c) the proposed.

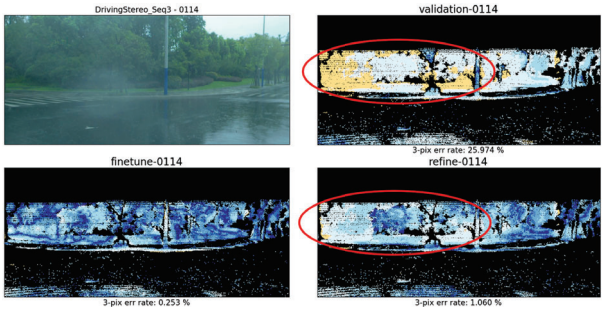
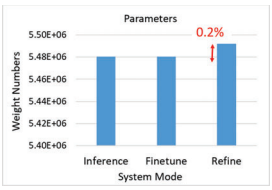
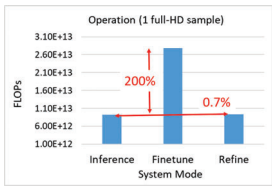


Fig. 5: Frame 114th of the raining scene in the driving stereo dataset [14].





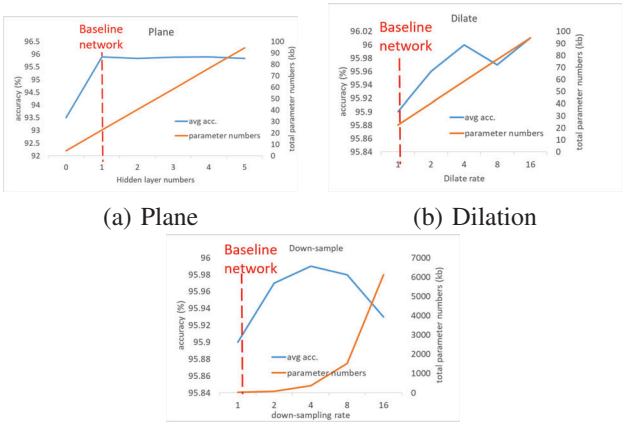
(a) Comparison of parameters (b) Comparison of operation

Fig. 6: Comparison of overhead between different simulation mode.

higher multi-scale rate, the cost is much higher than the bought out improvement. Over the above analysis, we finally choose the baseline model for the following up architecture design because of the best cost-performance ratio.

C. Bandwidth Comparison With Different Techniques

The proposed approach achieves the best bandwidth reduction, which is shown in Fig. 8. With the elimination of intermediate result off-chip memory footprint, we can achieve a 97% of bandwidth reduction and meet the hardware specification.



(a) Plane (b) Dilation

(c) Down-sampling

Fig. 7: Comparison of overhead between different simulation mode.

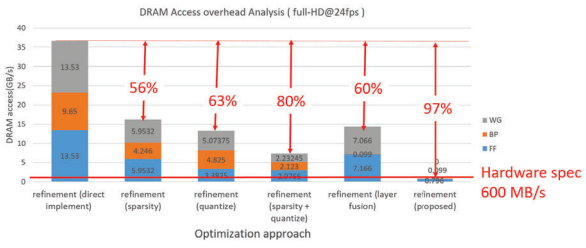


Fig. 8: Bandwidth comparison between different optimization approaches

V. Conclusion

We propose a two-stage online stereo matching refinement system. We fixed the pre-trained stereo matching model and attached an additional light-weight network to learn the domain gap between local data and cloud training data. This system

has a much better load–gain ratio than fine-tune. The proposed network disburses 0.2 % of additional parameters and 0.7 % additional computation as set by inference the stereo matching model. We exploited the network characteristics of refinement network and re-schedule the training procedure dataflow. A patch-based layer-fused training procedure is proposed, which can finally save 97 % of bandwidth.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5410–5418.

[3] C.-H. Lin, C.-C. Cheng, Y.-M. Tsai, S.-J. Hung, Y.-T. Kuo, P. H. Wang, P.-K. Tsung, J.-Y. Hsu, W.-C. Lai, C.-H. Liu *et al.*, "7.1 a 3.4-to-13.3 tops/w 3.6 tops dual-core deep-learning accelerator for versatile ai applications in 7nm 5g smartphone soc," in *IEEE International Solid State Circuits Conference (ISSCC)*, 2020, pp. 134–136.

[4] J. Lee, J. Lee, D. Han, J. Lee, G. Park, and H.-J. Yoo, "7.7 lnpu: A 25.3 tflops/w sparse deep-neural-network learning processor with fine-grained mixed precision of fp8-fp16," in *IEEE International Solid State Circuits Conference (ISSCC)*, 2019, pp. 142–144.

[5] Z. Yuan, J. Yue, H. Yang, Z. Wang, J. Li, Y. Yang, Q. Guo, X. Li, M.-F. Chang, H. Yang *et al.*, "Sticker: A 0.41-62.1 tops/w 8bit neural network processor with multi-sparsity compatible convolution arrays and online tuning acceleration for fully connected layers," in *IEEE Symposium on VLSI Circuits*, 2018, pp. 33–34.

[6] B. Fleischer, S. Shukla, M. Ziegler, J. Silberman, J. Oh, V. Srinivasan, J. Choi, S. Mueller, A. Agrawal, T. Babinsky *et al.*, "A scalable multi-teraops deep learning processor core for ai trainina and inference," in *IEEE Symposium on VLSI Circuits*, 2018, pp. 35–36.

[7] C.-H. Lu, Y.-C. Wu, and C.-H. Yang, "A 2.25 tops/w fully-integrated deep cnn learning processor with on-chip training," 2019, pp. 65–68.

[8] S. Choi, J. Sim, M. Kang, Y. Choi, H. Kim, and L.-S. Kim, "A 47.4 μj/epoch trainable deep convolutional neural network accelerator for in-situ personalization on smart devices," in *IEEE Asian Solid State Circuits Conference (A-SSCC)*, 2019, pp. 57–60.

[9] S. Choi, J. Shin, Y. Choi, and L.-S. Kim, "An optimized design technique of low-bit neural network training for personalization on iot devices," in *Design Automation Conference (DAC)*, 2019, pp. 1–6.

[10] J. Pang, W. Sun, J. S. Ren, C. Yang, and Q. Yan, "Cascade residual learning: A two-stage convolutional neural network for stereo matching," in *IEEE International Conference on Computer Vision (ICCV) Workshop*, 2017, pp. 887–895.

[11] S. Lym, A. Behroozi, W. Wen, G. Li, Y. Kwon, and M. Erez, "Mini-batch serialization: Cnn training with inter-layer data reuse," *arXiv preprint arXiv:1810.00307*, 2018.

[12] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer cnn accelerators," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–12.

[13] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3061–3070.

[14] G. Yang, X. Song, C. Huang, Z. Deng, J. Shi, and B. Zhou, "Drivingstereo: A large-scale dataset for stereo matching in autonomous driving scenarios," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 899–908.