WP522 (v1.0)

Optimization for Depth-Wise Convolution on Xilinx Devices

Depth-wise convolution can achieve better performance with fewer FLOPS compared to standard convolution among different tasks. Owing to its wide adoption for AI networks, Xilinx's adaptable platform offers a deep convolution engine, which achieves outstanding performance in edge and cloud accelerator cards.

ABSTRACT

This white paper explores depth-wise convolution deep learning operations implemented on Xilinx adaptive devices. This white paper aims to provide multiple optimization strategies for different Xilinx devices to meet a variety of task requirements. For the edge side, Xilinx implements a lightweight depth-wise convolution engine that supports the corresponding computing requirements. The white paper also describes how the top-ranking MobileNetv1 latency was achieved using specially designed DSAs on a Xilinx Alveo™ adaptive platform, targeting latency-critical autonomous driving applications.

Introduction

Deep learning brings incredible improvement in both accuracy and reliability compared with traditional algorithms in various computer vision tasks. Everyday, new AI models are being invented to achieve higher accuracy or lower hardware requirements. However, fixed AI accelerators such as GPUs and ASICs struggle to meet real-life performance due to increasing complexity in network layers, layer-to-layer dataflow, and lower mixed precisions. Xilinx programmable devices (FPGAs, SoCs, and ACAPs) can implement various domain specific architectures (DSAs) to optimize CNN, LSTM, and MLP in different vectors such as throughput, latency, and hardware utilization. In this white paper, the focus is on MobileNet, which contains depth-wise separable convolution (DSC).

DSC reduces computations compared with standard convolution (SC). The basic idea of DSC is to split standard convolution into two operations: Depth-wise convolution (Dwcv) and Point-wise convolution (Pwcv). Dwcv applies a single convolutional filter per input channel. The number of convolution filters is the same as the number of input channels. Pwcv is a 1x1 standard convolution following the depth-wise convolution in the model structure. It is used to aggregate feature information from different channels in the same spatial position. In this way, separable convolution can reduce computational cost and memory utilization of networks. See Figure 1.

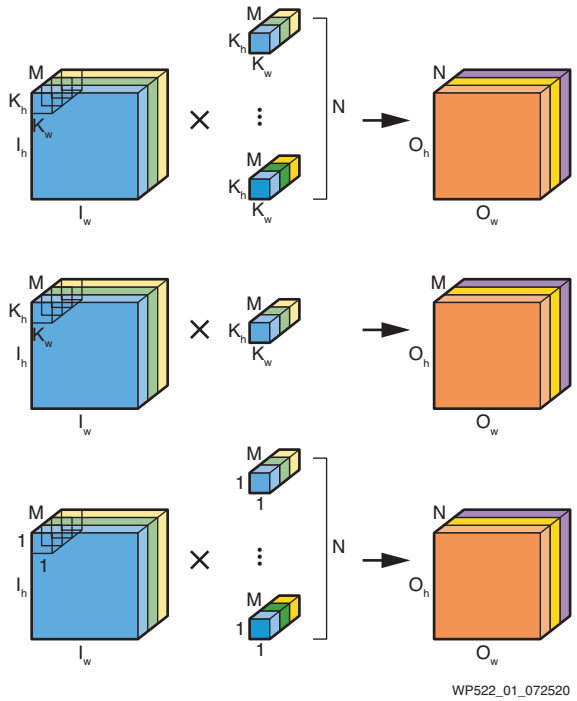

Figure 1: **Standard, Depth-wise, and Point-wise Convolution**

FLOPS and Parameters Analysis: Figure 1 demonstrates standard convolution and DSC. The size of input feature map is $I_w \times I_h \times M$, the kernel size is $K_w \times K_h$, and the output feature size is $O_w \times O_h \times N$, O is the output feature map, M is the number of input channels, N is the number of output channels. For a stride length of 1, the amount of operations is computed as:

Equation 1

$$O_{sc} = I_w \times I_h \times K_w \times K_h \times M \times N$$

For DSC, the amount of operations is:

Equation 2

$$O_{dsc} = I_w \times I_h \times K_w \times K_h \times M + I_w \times I_h \times M \times N$$

which is the sum of Dwcv and Pwcv. Splitting an SC into a Dwcv and a Pwcv permits the reduction operation of:

Equation 3

$$F_o = \frac{O_{dsc}}{O_{sc}} = \frac{1}{N} + \frac{1}{K_w \times K_h}$$

DSCs are the vital block in many efficient neural network architectures (e.g., MobileNet and EfficientNet series) for different compute vision tasks. Table 1 compares popular networks with standard convolution and depth-wise separable convolution in different tasks. In general, networks with DSC can achieve better performance with fewer FLOPS compared with SC among different tasks

Table 1: **Standard Conv to Depth-wise Conv Comparison**

	Model	Tasks	FLOPS (Billion)	Evaluation (%) (top / mAP / mIou)
Standard Conv	VGG16 [Ref 1]	Classification	15.3	71.5
	ResNet101 [Ref 2]	Classification	7.6	77
	Inceptionv3 [Ref 3]	Classification	5	78.2
	VGG16-SSD [Ref 4]	Object Detection	35.2	23.2
	YOLOV3 [Ref 5]	Object Detection	71	33
	RetinaNet (ResNet101) [Ref 6]	Object Detection	127	37.9
	DeepLabv3+ (ResNet101) [Ref 7]	Segmentation	81.02	77.21

Table 1: **Standard Conv to Depth-wise Conv Comparison *(Cont'd)***

	Model	Tasks	FLOPS (Billion)	Evaluation (%) (top/mAP/mIou)
Depth-wise Separable Conv	MobileNetv3 [Ref 8]	Classification	0.219	75.2
	EfficientNet-B0 [Ref 9]	Classification	0.39	77.3
	EfficientNet-B7 [Ref 9]	Classification	37	84.4
	MobileNetV3-SSD[Ref 8]	Object Detection	0.51	22.0
	EfficientDet-D0 [Ref 10]	Object Detection	2.5	32.4
	EfficientDet-D5 [Ref 10]	Object Detection	136	49.8
	EfficientDet-D7 [Ref 10]	Object Detection	326	51.0
	DeepLabv3+ (Xception) [Ref 7]	Segmentation	68	79.17

Implementation of Depth-wise Conv Engine

The Xilinx® Deep Learning Processor Unit (DPU) is a programmable engine of INT8 operations optimized for convolutional neural networks. The mult-add operations of CNN are processed in parallel in the computation engines including standard convolution (SC) engine, depth-wise separable convolution (DSC) engine, etc.

Pipeline Schedule Among Layers

The most common CNN structure is shown in Figure 2. The output result of one layer is the input feature of the following layer. It indicates that the following layer can start calculating after its sliding window (Kh × Iw) fills up on the outputs of the current layer. It is feasible to achieve the on-chip pipeline schedule to efficiently process the Pwcv layer and Dwcv layer in parallel, and to reduce the off-chip memory access. The Dwcv engine uses much less computing power than the original Conv engine and focuses on the processing of Dwcv layer. The Conv engine computes the SC and Pwcv layer.

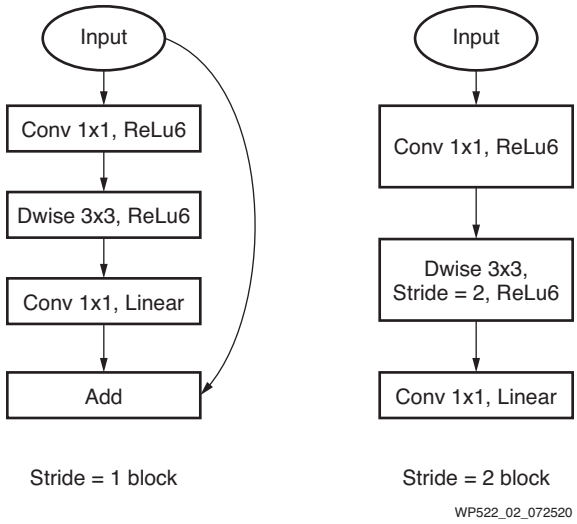

Figure 2: **Convolution Blocks in Model**

The pipeline schedule combines the three contiguous layers: Conv-Dwcv-Conv into one layer, which is shown in Figure 3(b). The input feature or weights of some layers can be too large to be filled into the on-chip memory. In that case, only the Conv-Dwcv layer is combined, while the next Conv layer is not incorporated, as shown in Figure 3(c). With the on-chip layer-combination, the Conv layer and Dwcv layer can be processed in parallel for higher efficiency, and the reduced off-chip memory access further decreases the run-time of latter layers. The schedule controls the data dependency among layers at row level. The process of each latter layer should wait for its Kh rows of input feature, i.e., the output feature of its previous layer. When profiling different strategies of the proposed parallel schedule, the height and width of the rectangle qualitatively indicate the contrasts of compute parallelism and computing time. Only the first layer in the combination needs read access to off-chip memory, and only the last layer needs write access. The profiling shows that the pipeline schedule of combined-layer can make the system more efficient.

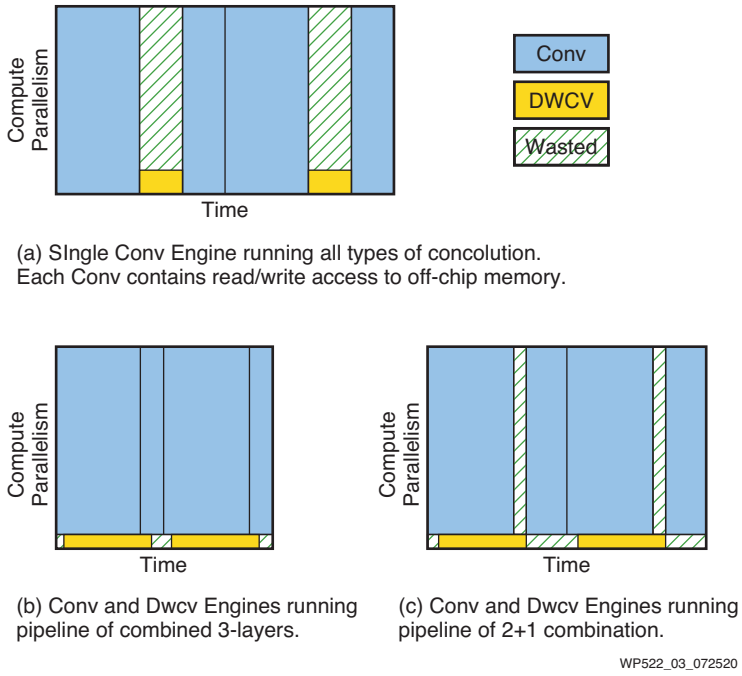

Figure 3: **Strategy Profiles**

Scalable Computation Engines

The Conv engine can process in parallel on the three dimensions: output channel parallel (OCP), input channel parallel (ICP), and parallel pixel (PP) (along lh direction), while the Dwcv engine only needs ICP and PP, and Dwcv does not combine the input channels to produce one output channel. The parallelism ratio Rp of the Conv engine to the Dwcv engine is calculated by the following:

$$R_p = \frac{OCP_{Conv} * ICP_{Conv} * PP_{Conv}}{ICP_{Dwcv} * PP_{Dwcv}}$$

Typically, the following are the default. $ICP_{conv} = ICP_{dwcv}$ and $ICP = OCP = CP$. Thus the R_p can be simplified as:

$$R_p = CP * \frac{PP_{Conv}}{PP_{Dwcv}}$$

In the case of MobileNets, the operation amount of Conv is nearly 32 times of that of Dwcv. Thus the R_p can be tuned to 32 to balance the run-time (along I_w dimension) between Conv engine and Dwcv engine to achieve the profiling illustrated in Figure 3. Typically, the system can be faster when CP and PP are larger, but the system is also limited by on-chip resources. The CP can be set to 16 and PP_{conv} to 8, because the number of IC in most layers in MobileNets are multiples of 16, and the height of most layers are no more than and close to multiples of 8. For the smaller chip, CP and PP_{conv} might need to be smaller. Figure 4 illustrates the scalable computing parallelism of Conv and Dwcv engines.

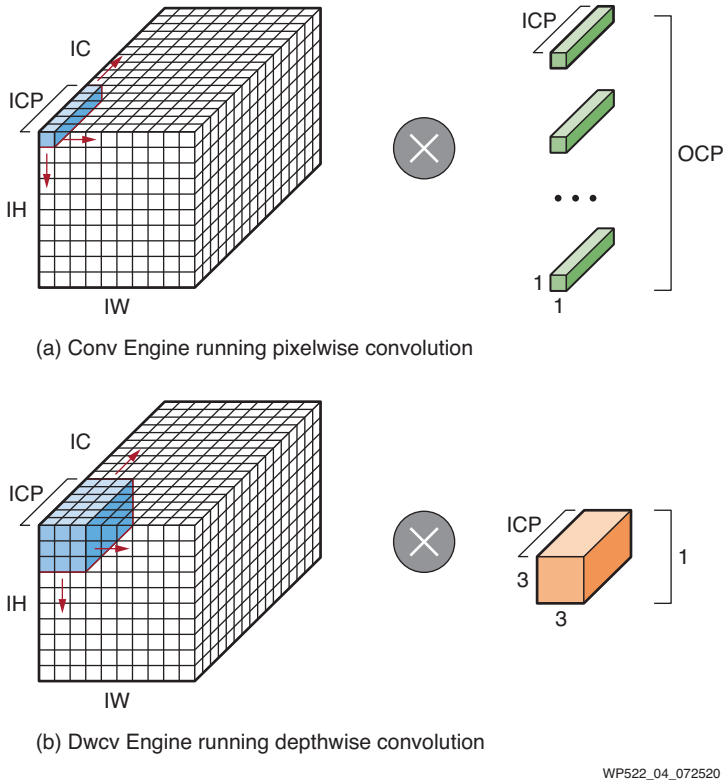

Figure 4: **Computing Parallelism of Conv Engine and Dwcv Engine**

Optimization of Depth-wise Convolution on Xilinx Edge Adaptive Devices

Data Flow Optimization

The on-chip memory (block RAM, distributed RAM, or UltraRAM) is partitioned into banks that are used as caches of feature, weight, and bias data. The bias data can be configured as channel-wise or layer-wise. When the accelerator starts to execute a layer, the feature, weight, and bias data are first loaded from off-chip memory to their corresponding banks. If the data in the banks is enough for an engine to work, the engine reads inputs from the banks, does the calculation, and then writes the result back to the banks. Finally, the result is saved from the banks to off-chip memory. The scheduling module controls the work timing of the engines. Figure 5 shows the data flow of features when Conv engine and Dwcv engine work in a pipeline. The Conv engine reads its input feature from bank0 and writes its result to bank1, Dwcv engine reads Conv engine's result as its input from bank1 and writes its result to the same bank.

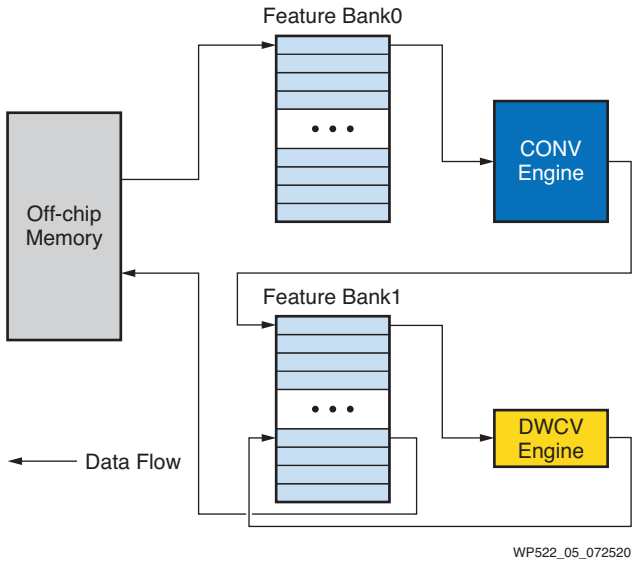

Figure 5: **Data Flow of Feature when Conv and Dwcv Work in Pipeline**

There is no need to instantiate a new bank for Dwcv to store its result because the output data of Dwcv is much less than that of Conv. Reuse of bank1 could save the on-chip memory, but when there is enough on-chip memory on the device, a new bank can be used to achieve high performance. The configurable bank utilization makes the design more scalable.

Performance and Competitive Analysis on the Edge Side

The DPU runs at 430MHz frequency on the Zynq® UltraScale+™ ZU2EG MPSoC and runs at 333MHz on the ZCU102 Evaluation Board. The ZU2EG device can achieve higher frequency because it has more routing resources.

Classification Results for Zynq UltraScale+ ZU2EG and ZU9EG MPSoCs: Table 2 provides a comparison between the Xilinx results and other similar accelerators on an ImageNet classification. The input image size of the Xilinx MobileNetV2 on the Zynq UltraScale+ ZU2EG and ZU9EG MPSoCs is 224 x 224; the operation is 608M. The input image size of the Xilinx MobileNetV2+SSD is 360 x 480; the operation is 6.57G. The Xilinx module is based on a Caffe model. The Xilinx accelerator achieves 205.3fps on the ZU2EG MPSoC, which is 2.13X faster than that of Synetgy on the Zynq UltraScale+ ZU3EG MPSoC[Ref 14]. Even though Xilinx's ZU9EG MPSoC and Intel's Stratix-V and Arria 10 SoCs have comparable logic resources, the Xilinx accelerator's performance on the ZU9EG device is 3.04X faster than the competitors.

Table 2: **Performance Comparison in Classification**

Design	Network	Platform	Speed (fps)	Top-1	Prec.(W/A)
[Ref 11]	MobileNetV2	CPU	13.3	72.0	32b
[Ref 12]	RR-MobileNet	ZU9EG	127.4	64.6	8-4b
[Ref 13]	MobileNetv1	Stratix V	231.7	–	–
[Ref 14]	DiracDeltaNet	ZU3EG	96.5	68.47	1-4b
[Ref 15]	MobileNetV2	Arria10	266.2	–	16b

Table 2: **Performance Comparison in Classification *(Cont'd)***

Design	Network	Platform	Speed (fps)	Top-1	Prec.(W/A)
Xilinx	MobileNetV2	ZU2EG	205.3	68.1	8b
Xilinx	MobileNetV2	ZU9EG	809.8	68.1	8b

Detection Results on Zynq UltraScale+ ZU2EG and ZU9EG MPSoCs: MobileNets can also be applied as an effective base network in modern object detection system [Ref 16]. Xilinx also implemented DPU for object detection in MobileNetV1 + SSD model on ZU2EG and ZU9EG devices. Rare neural network accelerators provide results for object detection in a MobileNet-like network, so Xilinx just compared the performance of object detection between different sizes of DPU. As described in Table 3, the Xilinx accelerator achieves 31fps on ZU2EG devices and 124.3 fps on ZU9EG devices compared to CPU.

Table 3: **Runtime and Frame Rate in Detection**

Framework	Platform	Speed (fps)	Prec. (W/A)
MobileNetV1 + SSD	Xilinx Zynq UltraScale+ ZU2EG MPSoC	31.0	8b
MobileNetV1 + SSD	Xilinx Zynq UltraScale+ ZU9EG MPSoC	124.3	8b

Optimization of Depth-wise Convolution on Xilinx FPGA with HBM

DPUs implemented on devices without High Bandwidth Memory (HBM) often face restrictions with on-chip memory space. The device must frequently exchange data between on-chip storage and the DDR, and the bandwidth of the DDR limits the computing power of DPU, making it difficult to be used as a low latency solution. As shown in Figure 6, when implementing DPU on devices with HBM, full advantage can be taken of the high bandwidth of HBM and the on-chip feature map buffer banks can be allocated into HBM device. The HBM is so large that there is no need for data movement between other DDR devices, and the high bandwidth of HBM easily supports larger computing arrays, which provide high compute power. The DPU micro-architecture implemented on the Xilinx Alveo™ accelerator U50 and U280 cards can be used as an excellent low latency solution.

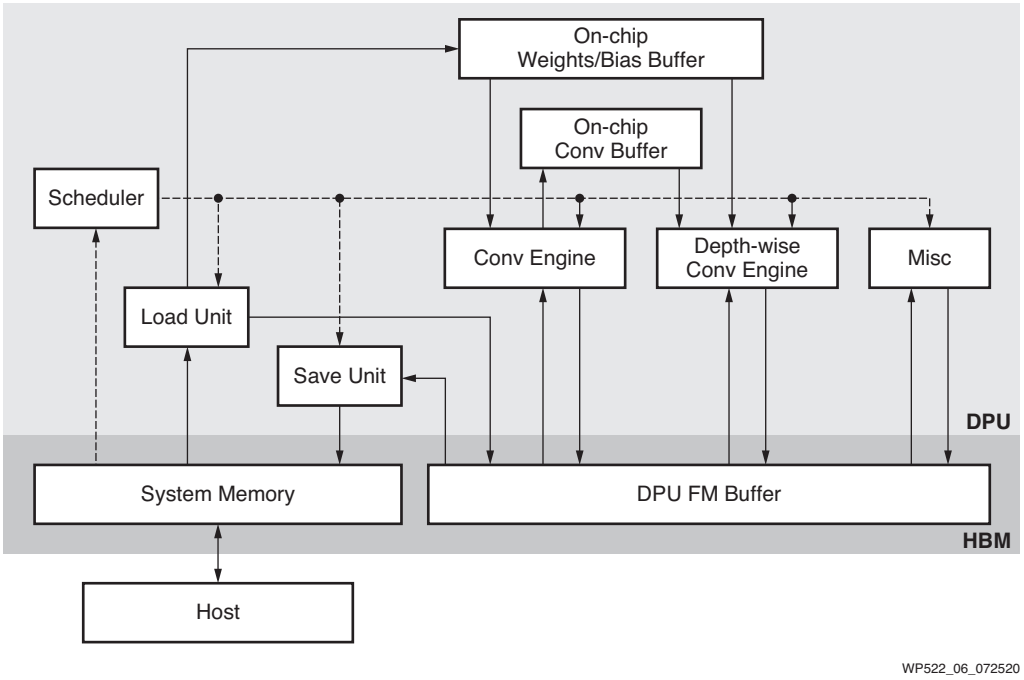

Figure 6: **Data Flow of Feature when Conv and Dwcv Work in Pipeline**

Performance of the Xilinx Design and Competitive Analysis

One V3ME DPU can provide up to 4.9 TOPS computation power for convolution, with a separate Dwcv engine for pooling and element-wise layers. The DPU is an excellent low latency solution for networks that have depth-wise layers or big image size. Some test data is demonstrated and compared with other solutions.

MLPerf SSD w/ MobileNet-v1 (224 × 224) Stream Test: The stream test results are shown in Table 4.

Table 4: **MLPerf Top 5 Test Results**

ID	Submitter	System	Benchmark Results / ms
Inf-0.5-22	Intel	Intel® Xeon® Platinum 9200 processors	1.4
Inf-0.5-28	NVIDIA	NVIDIA Jetson AGX Xavier	1.5
Inf-0.5-3	Dell EMC	Dell EMC R740xd with 2nd generation Intel® Xeon® Scalable Processor	1.54
Inf-0.5-32	Centaur Technology	Centaur Technology Reference Design v1.0	1.54
Inf-0.5-4	Dell EMC	Dell EMC R740xd with 2nd generation Intel® Xeon® Scalable Processor	1.69
	Xilinx	**DPU V3ME**	**1.59**

Other Non-MLPerf Benchmark (224 × 224) Test Results: Some non-MLPerf standard benchmark stream test results are shown in Table 5.

Table 5: **Non-MLPerf Benchmark Test Results (224 _ 224)**

Benchmarks	Stream Test Results / ms	Comment
MobileNetV2	0.8	–
Pruned MobileNetV1-SSD	0.84	MLPerf benchmark MLPerf SSD w/ MobileNet-v1 with pruning

High Resolution Benchmark (1920 × 1080) Stream Test Results: Table 6 shows the test results of a subset of the high-resolution (1920 x 1080) test sets on the V3ME DPU. Networks with and without depth-wise layers (such as Renet50 and YoloV2) can be accelerated, but the performance of networks with depth-wise layers is much more significant, such as MobileNetV2, which is as short as 13ms.

Table 6: **High Resolution Benchmark Test Results**

Benchmarks	Stream Test Results / ms
ResNet50	108.00
MobileNetv2	13.00
YoloV2	77.00

Conclusion

This white paper explores how depth-wise convolution deep learning operations are optimal on Xilinx devices. Edge devices (e.g., Zynq UltraScale+ ZU2EG and ZU9RG MPSoCs) and Alveo Data Center accelerator cards with HBM (e.g., U50, and U280) achieve excellent performance gains. Xilinx is well suited for Dwcv workloads for deep learning applications (e.g., image classification and object detection). Xilinx is continuing to innovate new hardware- and software-based methodologies to accelerate deep learning applications.

For more information on Xilinx's AI acceleration solution, go to: https://github.com/Xilinx/Vitis-AI/

Acknowledgments

The following Xilinx employees have authored or contributed to this white paper: Jinzhang Peng (Staff Software Engineer, AI Algorithm), Tianyu Zhang (Senior Design Engineer 1, AI Edge-Computing), Xianpei Zheng (Staff Design Engineer, AI Cloud-Computing), Dong Li (Senior Software Development Manager, AI Algorithm), Lu Tian (Software Development Director, AI Algorithm), Dongliang Xie (Design Engineer Director, AI Edge-Computing), Zhongmin Chen (Design Engineer Director, AI Cloud-Computing), and Yi Shan (Senior Director, AI).

References

1. Kour, G., & Saabne, R. (2014, September). Real-time segmentation of on-line handwritten arabic script. In 2014 14th International Conference on Frontiers in Handwriting Recognition (pp. 417-422). IEEE.
2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
3. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).
4. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.
5. Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.
6. Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).
7. Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European conference on computer vision (ECCV) (pp. 801-818).
8. Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Le, Q. V. (2019). Searching for mobilenetv3. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1314-1324).
9. Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946.
10. Tan, M., Pang, R., & Le, Q. V. (2019). Efficientdet: Scalable and efficient object detection. arXiv preprint arXiv:1911.09070.
11. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).
12. Su, J., Faraone, J., Liu, J., Zhao, Y., Thomas, D. B., Leong, P. H., & Cheung, P. Y. (2018, May). Redundancy-reduced MobileNet acceleration on reconfigurable logic for ImageNet classification. In International Symposium on Applied Reconfigurable Computing (pp. 16-28). Springer, Cham.
13. Zhao, R., Niu, X., & Luk, W. (2018, February). Automatic Optimising CNN with Depthwise Separable Convolution on FPGA: (Abstact Only). In Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 285-285).
14. Yang, Y., Huang, Q., Wu, B., Zhang, T., Ma, L., Gambardella, G., ... & Keutzer, K. (2019, February). Synetgy: Algorithm-hardware co-design for convnet accelerators on embedded fpgas. In Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 23-32).
15. Bai, L., Zhao, Y., & Huang, X. (2018). A CNN accelerator on FPGA using depthwise separable convolution. IEEE Transactions on Circuits and Systems II: Express Briefs, 65(10), 1415-1419.
16. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
01/28/2021	1.0	Initial Xilinx release.

Disclaimer

Automotive Applications Disclaimer