DRACO: Co-Optimizing Hardware Utilization, and Performance of DNNs on Systolic Accelerator

Nandan Kumar Jha[*], Shreyas Ravishankar[†], Sparsh Mittal[‡], Arvind Kaushik[§], Dipan Mandal[¶], Mahesh Chandra[§]

[*]IIT Hyderabad, [†]BITS Pilani Hyderabad, [‡]IIT Roorkee, [§]NXP Semiconductors,

[¶]Intel Labs, India. ([*]cs17mtech11010@iith.ac.in)

IEEE Computer Society Annual Symposium on VLSI 2020

July 8, 2020

Challenge: PE underutilization in memory-bound DNNs

Memory-bound DNNs: compute-efficient, but *low data reuse*. Requires *high bandwidth* hence most of PEs remain underutilized on a systolic array based DNN accelerator designed for large compute-bound DNN.

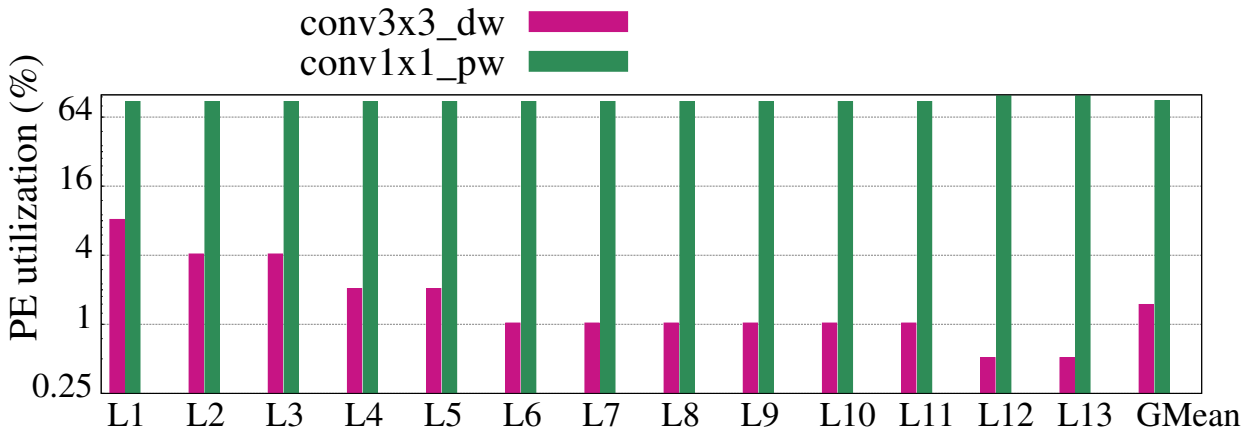


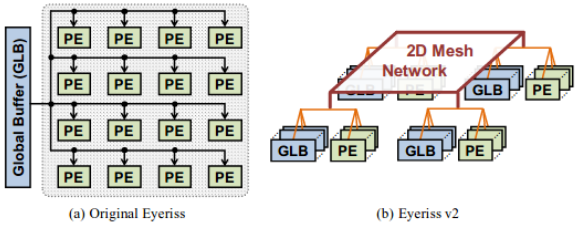
Figure 1 : MobileNet-V1 on Eyeriss accelerator with 16×16 array and row-stationary dataflow

PE utilization of 1×1 conv is ≈80% in almost all layers while in 3×3 depthwise conv is *≈4%* and decreases in deeper layers (even *lower than 1%*)

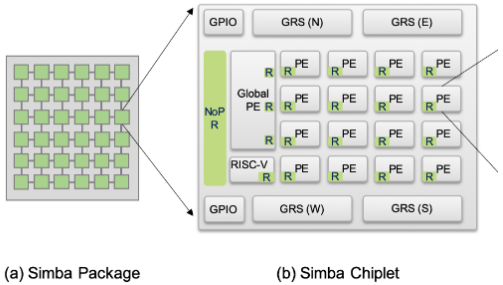
Need flexible DNN accelerators to efficiently support a diverse range compute-bound and memory-bound DNNs

Previous Work

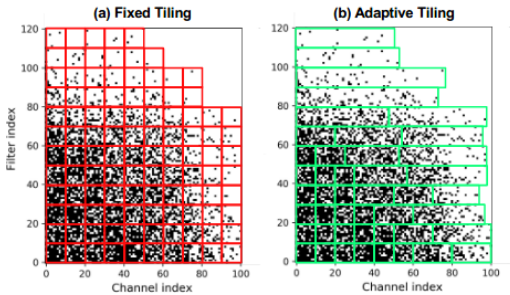
Proposed solution in previous works are either hardware-based optimization or dataflow modification.



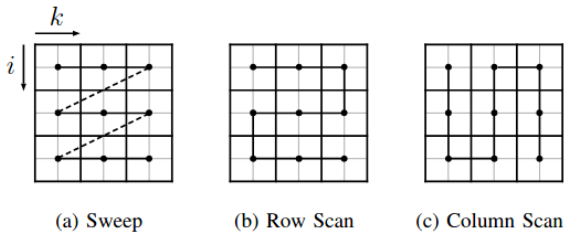
[Flexible NOC (Eyeriss-v2), Chen et al., JETCAS'19]



[Global PE (Simba), Shao et al., MICRO'19]



[Adaptive Tiling, Kung et al., ICPR'18]



[Flexible scanning, Wu et al., DATE'19]

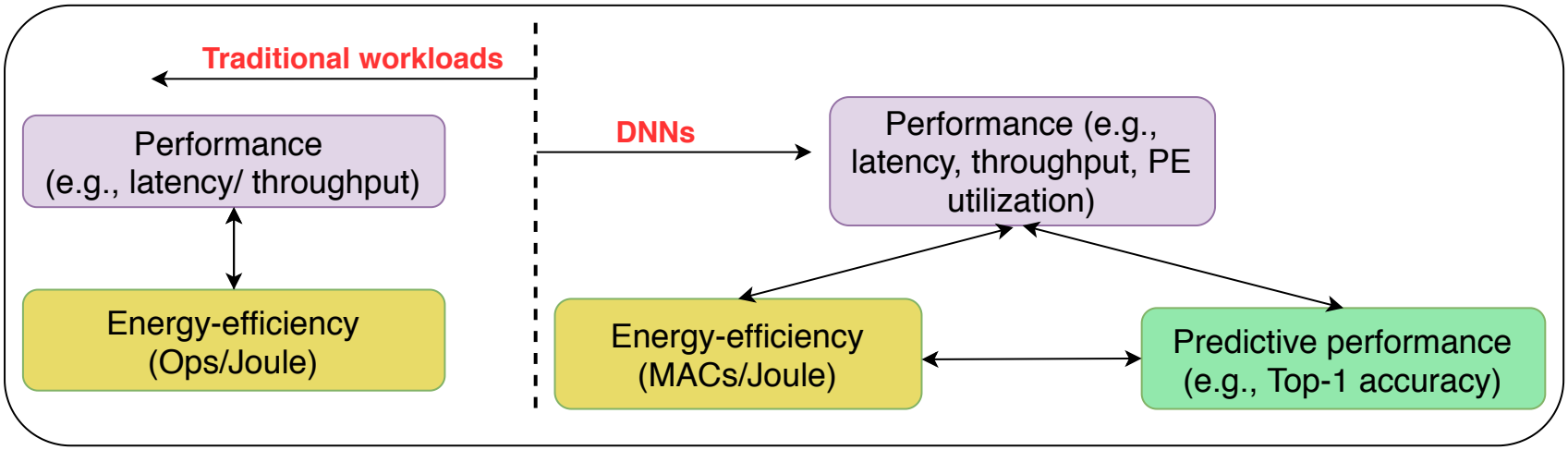
Pitfalls: Lack of flexibility (due to fixed functionality) in hardware based solution and the longer development cycles could render the proposed solution *ineffective* for SOTA DNNs.

Do we really need a specialized **microarchitecture** and/or **dataflow** to solve the low data reusability and PE underutilization challenges of depthwise convolution?

Solution?

Can we solve this problem at algorithm side (while preserving the predictive performance and energy efficiency of the network)?

Algorithmic optimization



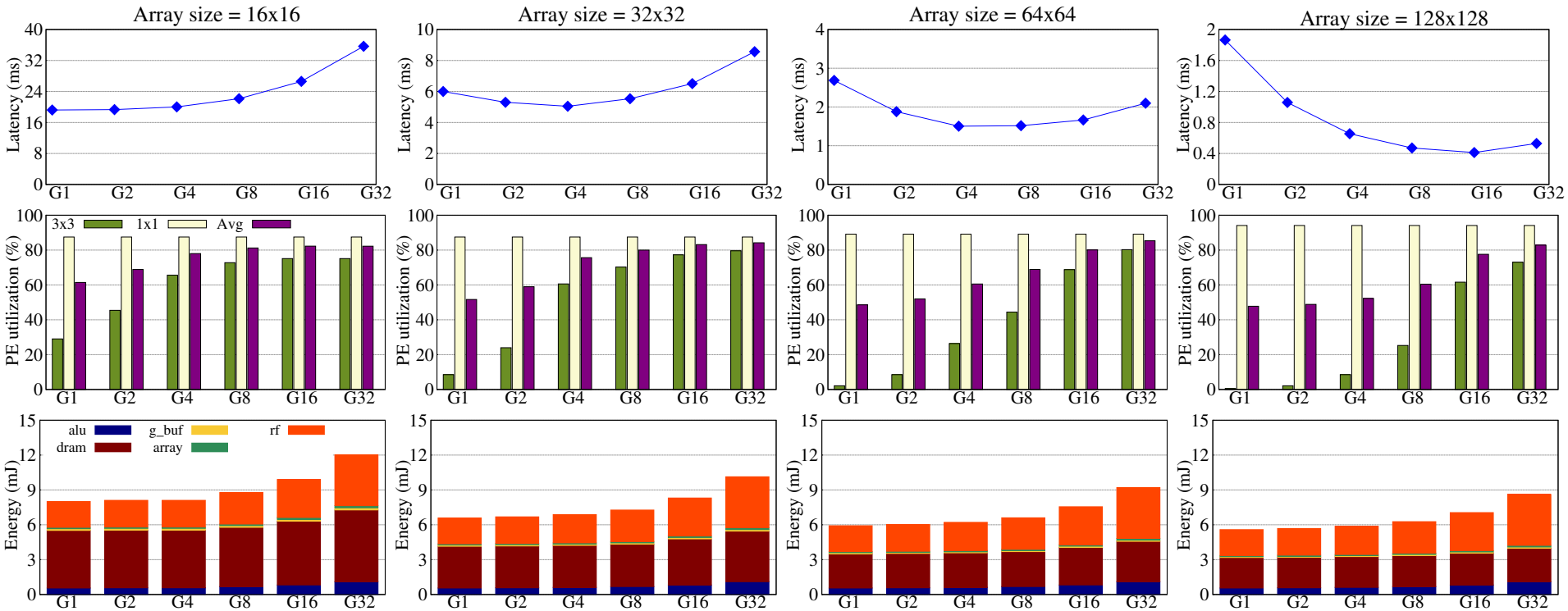
DRACO (Data Reuse Aware Co-Optimization)

Table: $d_k \times d_k$ and $d_f \times d_f$ are the spatial size of kernel (2D-filter) and feature map (ifmap/ofmap). m, n, and G are the #ifmaps, #ofmaps, and #channel per group

Metric	SConv	DWConv	DRACO
#MACs	$m \times n \times d_k^2 \times d_f^2$	$m \times d_k^2 \times d_f^2$	$\mathbf{G} \times (n \times d_k^2 \times d_f^2)$
$\#Param$	$m \times n \times d_k^2$	$m \times d_k^2$	$\mathbf{G} \times (n \times d_k^2)$
$WeightReuse$	d_f^2	d_f^2	d_f^2
$ActivationReuse$	$n \times \left(\frac{m}{m+n}\right) d_k^2$	$\left(\frac{m}{m+n}\right) d_k^2$	$\mathbf{G} \times \left(\frac{n}{m+n}\right) d_k^2$

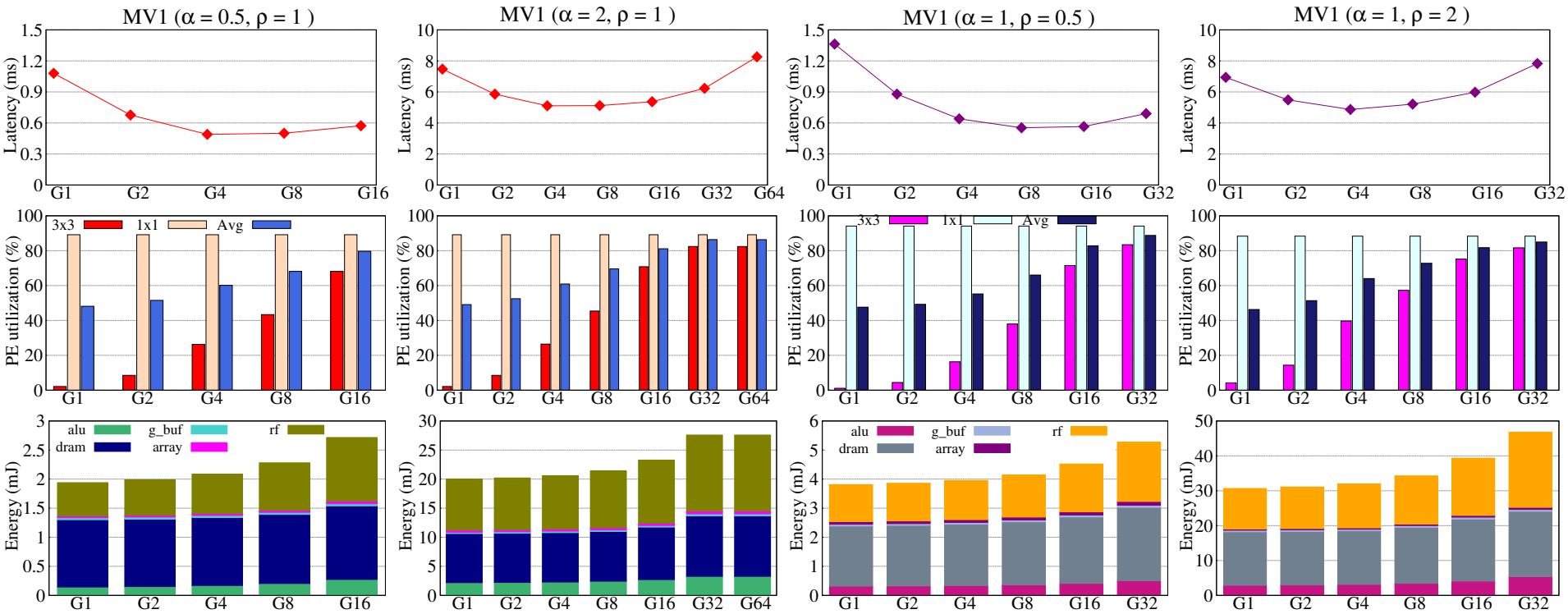
DRACO is a sweet point between SConv and DWConv and enables a **trade-off** between computational complexity and data reuse.

Experimental Results: MobileNetV1 on Eyeriss (1/2)



Takeaway 1: Optimum latency depends on both the PE utilization and the computational complexity of DNN, and the effect of PE utilization on latency depends on PE array size in the systolic array.

Experimental Results: MobileNetV1 on Eyeriss (2/2)



[α is width multiplier (↑ #filter-channels) and ρ is a resolution multiplier (↑ spatial size of fmaps). Array size here is 64×64]

Takeaway 2: Inference latency of DNNs with very few MACs depends only on PE utilization. Increasing PE utilization at the expense of a substantial increase in #MACs does not lower the latency as the effect of higher PE utilization is dominated by the #MACs.

Alternative for Latency Optimization

MobileNetV1 with α=0.5 and ρ=2: Reduces the computational overhead of improving PE utilization by reducing the #filter-channels

Performance comparison of MobileNetV1 versions $\{\alpha = 1$, and $\rho = 2\}$ and $\{\alpha = 0.5$, and $\rho = 2\}$

Model	Array size	Metric	G1	G2	G4	G8	G16
MV1 α=1, ρ=2	16x16	PE util. (%)	68	77	79	79	80
		Latency (ms)	66.5	67.7	72.1	81.1	99.2
		Energy (mJ)	59.7	60.1	61.0	63.7	69.3
	64x64	PE util. (%)	50	55	66	74	83
		Latency (ms)	6.9	5.5	4.9	5.2	6.0
		Energy (mJ)	30.6	31.1	31.9	34.3	39.3
MV1 α=0.5, ρ=2	16x16	PE util. (%)	68	76	79	79	80
		Latency (ms)	17.8	18.3	20.5	25.1	34.1
		Energy (mJ)	17.5	17.7	18.2	19.5	21.6
	64x64	PE util. (%)	49	54	66	73	81
		Latency (ms)	2.5	1.8	1.5	1.7	2.1
		Energy (mJ)	10.3	10.5	10.9	12.1	14.1

Latency is decreased by a factor of $\approx$**3.5**$\times$ on 16×16 systolic array and by $\approx$**3.27**$\times$ on 64×64 array size at G=4.

Predictive performance comparison

Top-1 accuracy (on Imagenette) for MobileNetV1 with different α and ρ

Models	G1	G2	G4	G8	G16	G32
MV1 (α=1 ρ=1)	84.08	84.55	84.65	83.46	83.40	79.94
MV1 (α=1 ρ=2)	84.76	84.55	84.17	84.81	83.29	82.90
MV1 (α=0.5 ρ=2)	82.61	83.54	83.70	82.71	82.29	-

- Changing G affects regularization.
  - Lower (higher) G $\Longrightarrow$ stronger (weaker) regularization.
- Changing G alters the **representational** power of network.
  - Higher G captures more variations of complex latent concepts.

Takeaway 3: With appropriate G we can get maximum predictive performance.

Conclusion

- Increasing PE utilization *does not* necessarily reduces the latency.
  - Effect of PE utilization on latency is dictated by the computational complexity of DNN and the systolic array-size.
- Increasing data reuse at the *expense* of higher computation does not affect energy efficiency significantly.
  - At lower G energy consumption is almost constant; however, at higher G it increases moderately.
- Predictive performance can be increased by fine-tuning G.

By fine-tunning G in DNNs with depthwise convolution, a **sweet point** for optimum latency, energy-efficiency, and accuracy can be achieved on a systolic DNN accelerator designed for large (compute-bound) DNNs

Thanks for Paying Attention..!!!