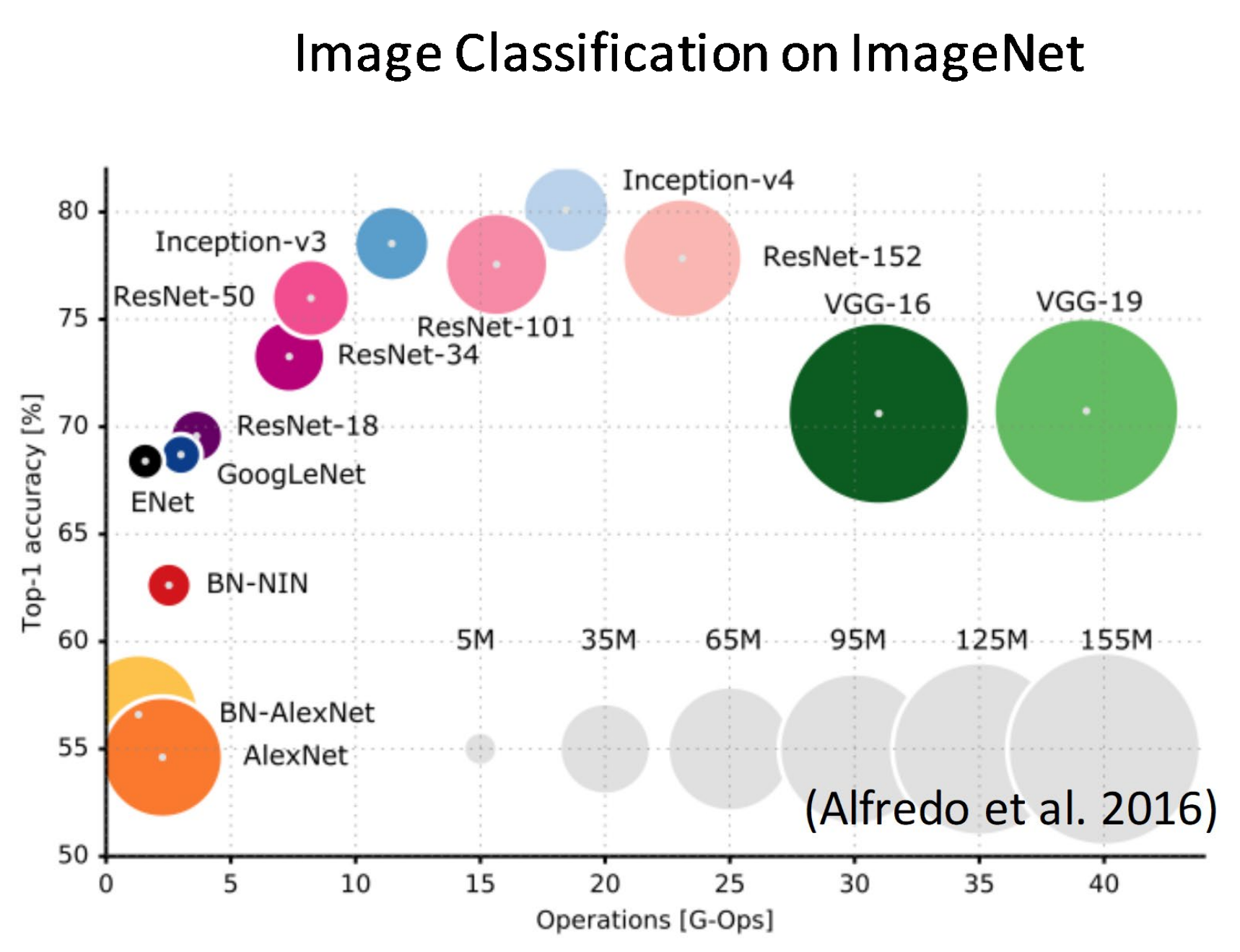
Exploring Bit-Slice Sparsity in Deep Neural Networks for Efficient ReRAM-Based Deployment

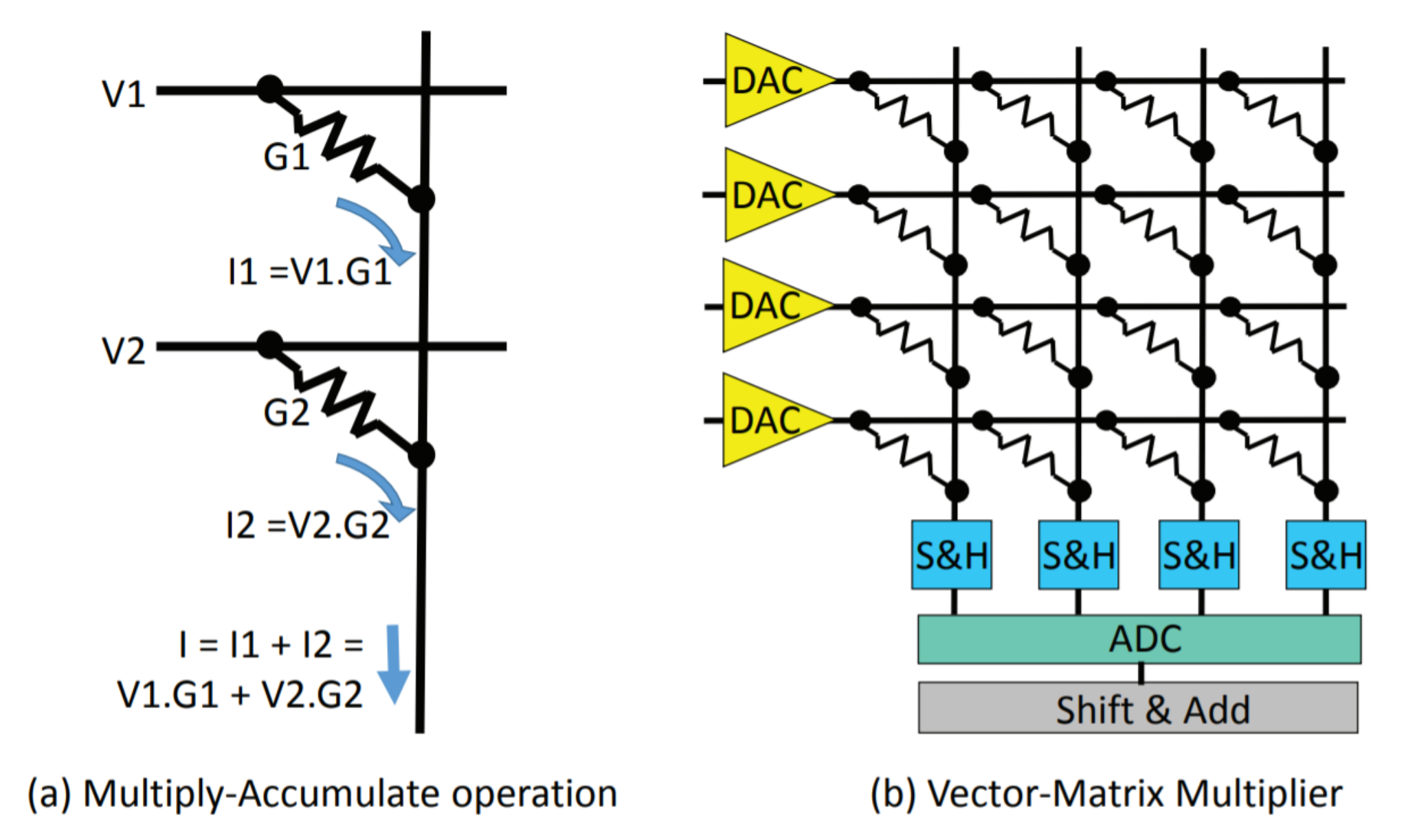
Jingyang Zhang[1], Huanrui Yang[1], Fan Chen[1], Yitu Wang[2], Hai Li[1]
[1]Duke University, [2]Fudan University

Motivation

- Deep Neural Network (DNN) models are powerful, but are costly to deploy.



- Resistive random-access memory (ReRAM)-based DNN accelerators (Shafiee et al. 2016, Song et al. 2017)
  - In-situ matrix-vector multiplications
  - Two-order magnitude advantage in energy, performance and chip footprint


V1
G1
I1 =V1.G1
V2
G2
I2 =V2.G2
I = I1 + I2 = V1.G1 + V2.G2
(a) Multiply-Accumulate operation
DAC
S&H
ADC
Shift & Add
(b) Vector-Matrix Multiplier

- Challenges of ReRAM-based accelerators
  - Limited cell bit density: operands (i.e. weights) are bit-sliced across multiple ReRAM bitlines
  - High bit-resolution ADC accounts for high power (>60%) and area (>30%) overhead

- ADC resolution dictated by accumulated currents on bitlines
- Need ***higher sparsity in each bit-slice*** to reduce the accumulated current on bitlines, therefore reducing ADC overhead

Methods

- Dynamic fixed-point quantization
  - For each layer, preserve dynamic range

$$S(W_l) = \lceil \log_2(\max_{w_l^i \in W_l} (|w_l^i|)) \rceil,$$

  - Uniform n-bit quantization after scaling

$$Q_{step} = 2^{S(W_l)-n}, \qquad B(w_l^i) = \lfloor \frac{w_l^i}{Q_{step}} \rceil.$$

    - $B(w_l^i)$ will be stored on ReRAM crossbar
  - Dynamic range recovery

$$Q(w_l^i) = B(w_l^i) \cdot Q_{step}$$

    - Recovery can be done efficiently with shifting
    - $Q(w_l^i)$ will be used for computation

Results

- Significant improvement on bit-slice sparsity

Table 1: Results on MNIST

Method	Accuracy	Ratio of non-zero wights $\hat{B}^3$	$\hat{B}^2$	$\hat{B}^1$	$\hat{B}^0$	Average
Pruned	97.99%	1.08%	5.87%	8.42%	17.42%	8.20±5.94%
ℓ_1	97.99%	1.19%	5.21%	7.01%	11.36%	6.19±3.65%
Bℓ_1	97.67%	**0.84%**	**4.02%**	**4.27%**	**9.58%**	**4.68±3.14%**

Table 2: Results on CIFAR-10

Model	Method	Accuracy	Ratio of non-zero wights $\hat{B}^3$	$\hat{B}^2$	$\hat{B}^1$	$\hat{B}^0$	Average
VGG-11	Pruned	88.93%	0.86%	28.30%	34.14%	33.39%	24.17±13.65%
	ℓ_1	**89.39%**	0.39%	9.37%	18.43%	22.19%	12.59±8.45%
	Bℓ_1	89.33%	**0.21%**	**3.57%**	**7.09%**	**10.71%**	**5.40±3.92%**
ResNet-20	Pruned	89.22%	1.10%	8.07%	21.92%	43.96%	18.76±16.36%
	ℓ_1	**90.62%**	0.44%	4.71%	14.37%	33.16%	13.17±12.60%
	Bℓ_1	89.66%	**0.31%**	**3.34%**	**11.99%**	**31.39%**	**11.76±12.12%**

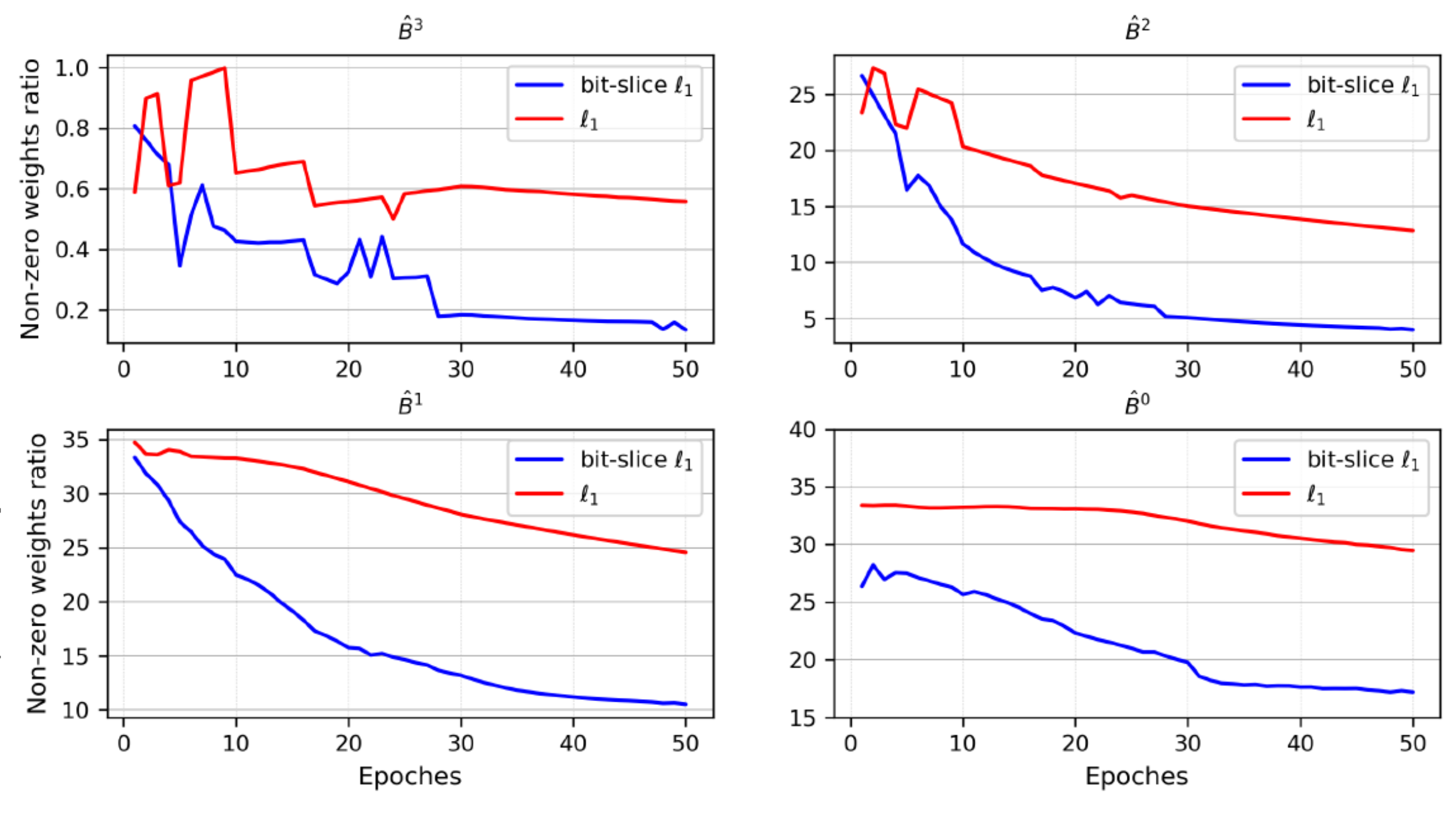


Figure 2: Bit-slice sparsity of VGG-11 on CIFAR-10 during training.

- Reducing ADC overhead on ReRAM-based accelerators
  - Map to 128 x 128 ReRAM crossbars (XBs)

Table 3: ADC Overhead Saving with Bit-Slice Sparsity

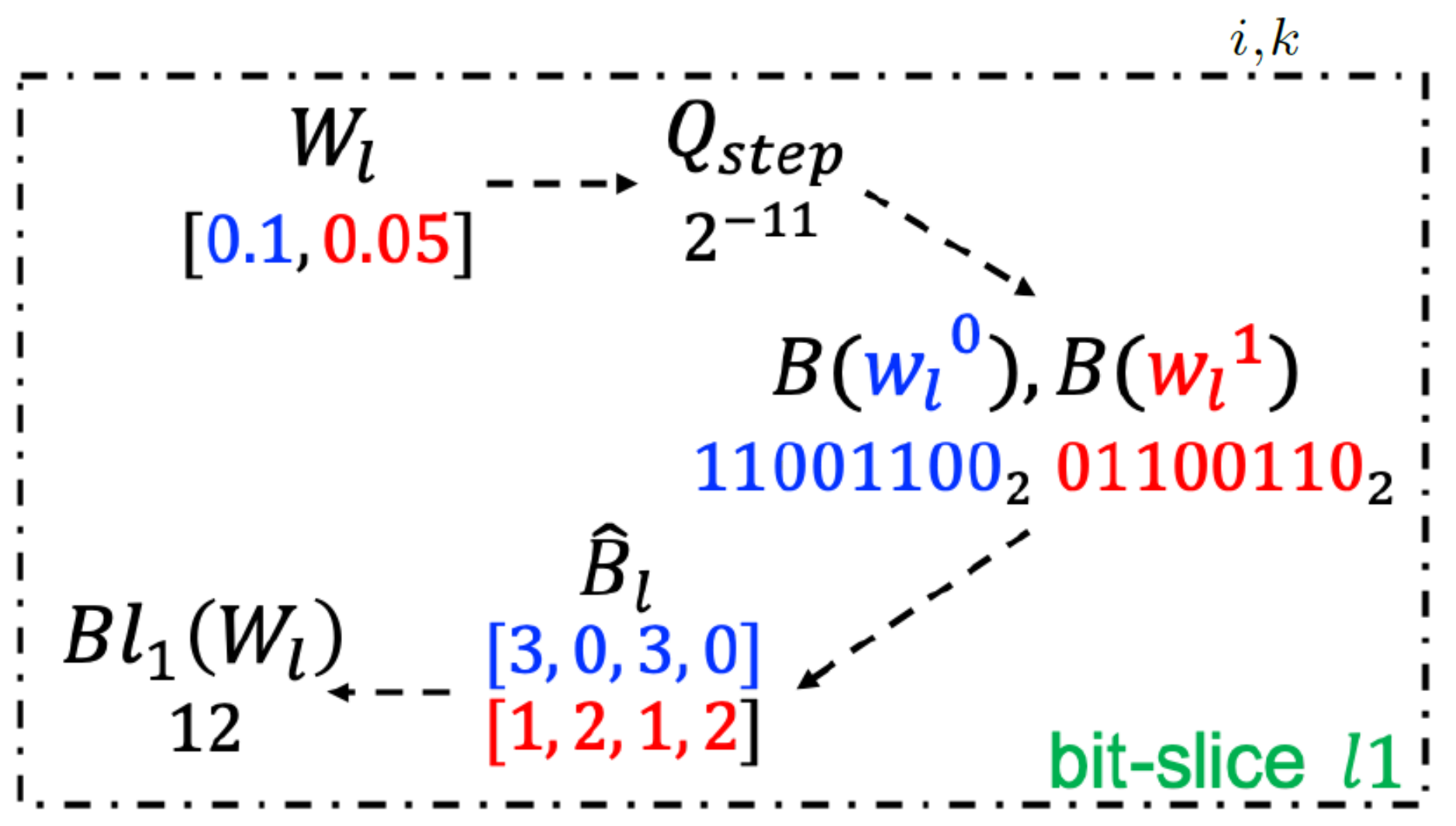
	w/o Bit-Slice Sparsity	w/ Bit-Slice Sparsity			
	Resolution	Resolution	Energy Saving	Speedup	Area Saving
XB$_3$	8 bit	1 bit	28.4×	8×	2×
XB$_{2,1,0}$	8 bit	3 bit	14.2×	2.67×	2×

Codes available at: https://github.com/zjysteven/bitslice_sparsity

Methods (cont'd)

- Bit-slice L1
  - Binary representation of the quantized weight
  - Slice into 2-bit slices
  - L1 regularization across all bit-slices of all elements within a weight matrix/tensor

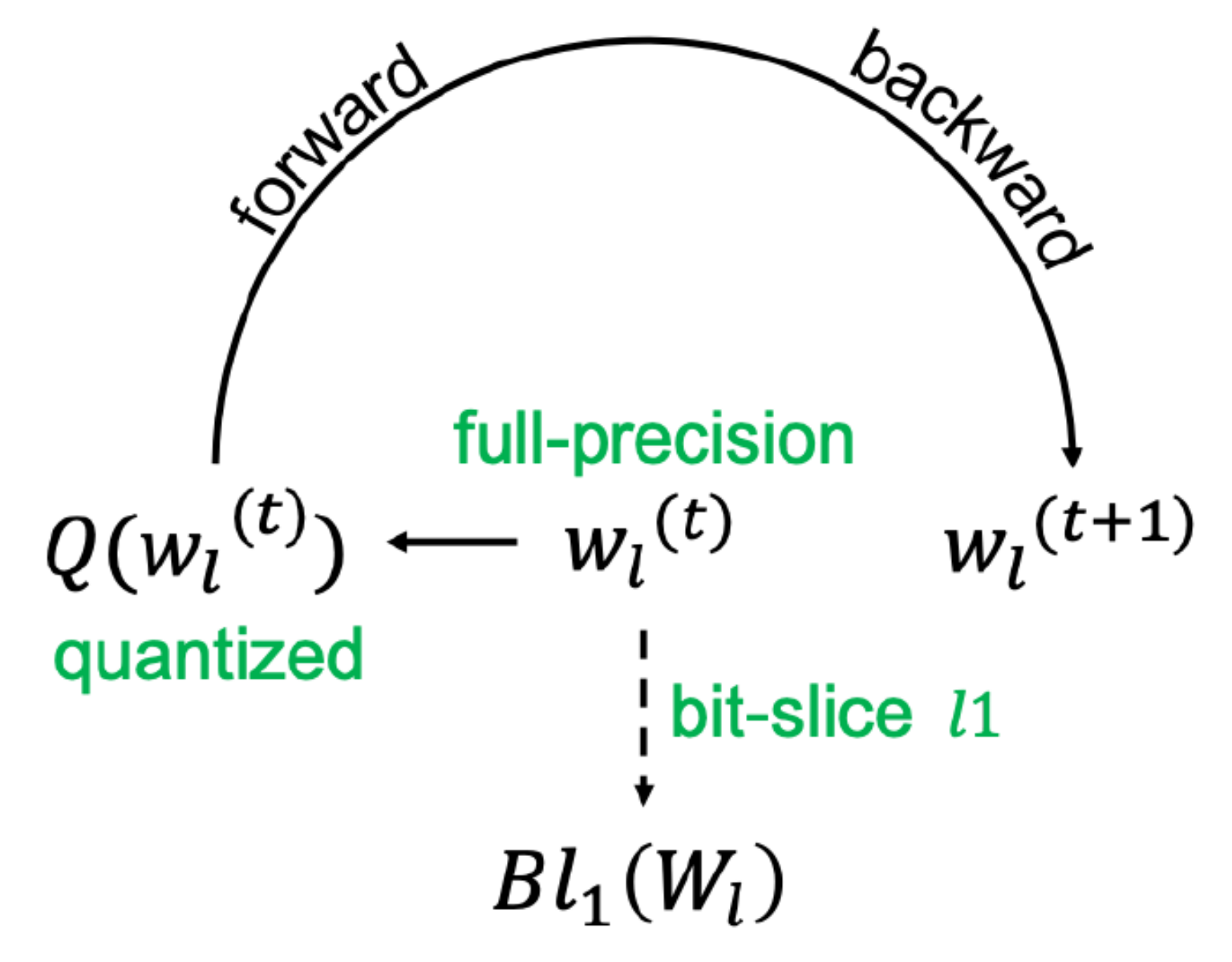
$$B(w_l^i) = \sum_{k=0}^{3} \hat{B}_l^{i,k} \cdot 2^{2k} \qquad B\ell_1(W_l) := \sum_{i,k} \hat{B}_l^{i,k}.$$



- Overall training routine
  - Forward and backward pass with quantized weight, gradient update on full-precision weight
  - Add bit-slice L1 to the objective

$$q^{(t)} = Q(w_l^{(t)}),$$

$$w_l^{(t+1)} = q^{(t)} - lr \times (\nabla_q \mathcal{L}_{CE}(q^{(t)}) + \alpha \nabla_q B\ell_1(q^{(t)}))$$



Conclusion

- We propose ***bit-slice L1*** regularizer, the first algorithm to induce bit-slice sparsity during the training of dynamic fixed-point DNNs
- ***Up to 2x or more*** sparsity improvement on bit slices comparing to traditional L1 regularizer
- When deployed on ReRAM-based accelerator, the achieved bit-slice sparsity allows the ADC resolution to be ***reduced to 1-bit*** of the most significant bit-slice and ***down to 3-bit*** for the others bits, which significantly reduces power and area overhead.

Acknowledgement

This work was supported in part by NSF CNS-1822085 and NSF CSR-1717885.

References

- A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar. Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars . In Proceedings of ISCA, 2016.
- L. Song, X. Qian, H. Li, and Y. Chen. Pipelayer: A pipelined reram-based accelerator for deep learning. . In Proceedings of HPCA, 2017.
- T. Yang, H. Cheng, C. Yang, I. Tseng, H. Hu, H. Chang, and H. Li. Sparse reram engine: Joint exploration of activation and weight sparsity in compressed neural networks. In Proceedings of ISCA, 2019.
- P. Gysel. Ristretto: Hardware-oriented approximation of convolutional neural networks. arXiv preprint arXiv:1605.06402, 2016.
- M. Saberi, R. Lotfi, K. Mafinezhad, and W. A. Serdijn. Analysis of power consumption and linearity in capacitive digital-to-analog converters used in successive approximation adcs. IEEE Transactions on Circuits and Systems I: Regular Papers, 2011.