CoDR: Computation and Data Reuse Aware CNN Accelerator

Alireza Khadem, Haojie Ye, Trevor Mudge
Computer Science and Engineering, University of Michigan
Ann Arbor, Michigan, USA
{arkhadem, yehaojie, tnm}@umich.edu

1. Introduction

In this work, we alleviate the computation resources and memory space required for the Deep Neural Networks (DNNs) inference. Figure 1 shows three complementary computation reuse techniques proposed in [2, 1, 4]. *CoDR* presents a novel CNN dataflow that employs scalar-matrix multiplication to exploit weight sparsity, repetition, and similarity simultaneously for the *Universal Computation Reuse* that exploits weight sparsity, repetition, and similarity simultaneously. Next, we customize *Run-Length Encoding* (RLE) scheme for the data values required for this technique. Finally, since weights are compressed, accesses to the on-chip weights are less costly than the accesses to the activations. Thus, we design the loop ordering of the *CoDR* dataflow to reduce the number of costly accesses to the input and output features.

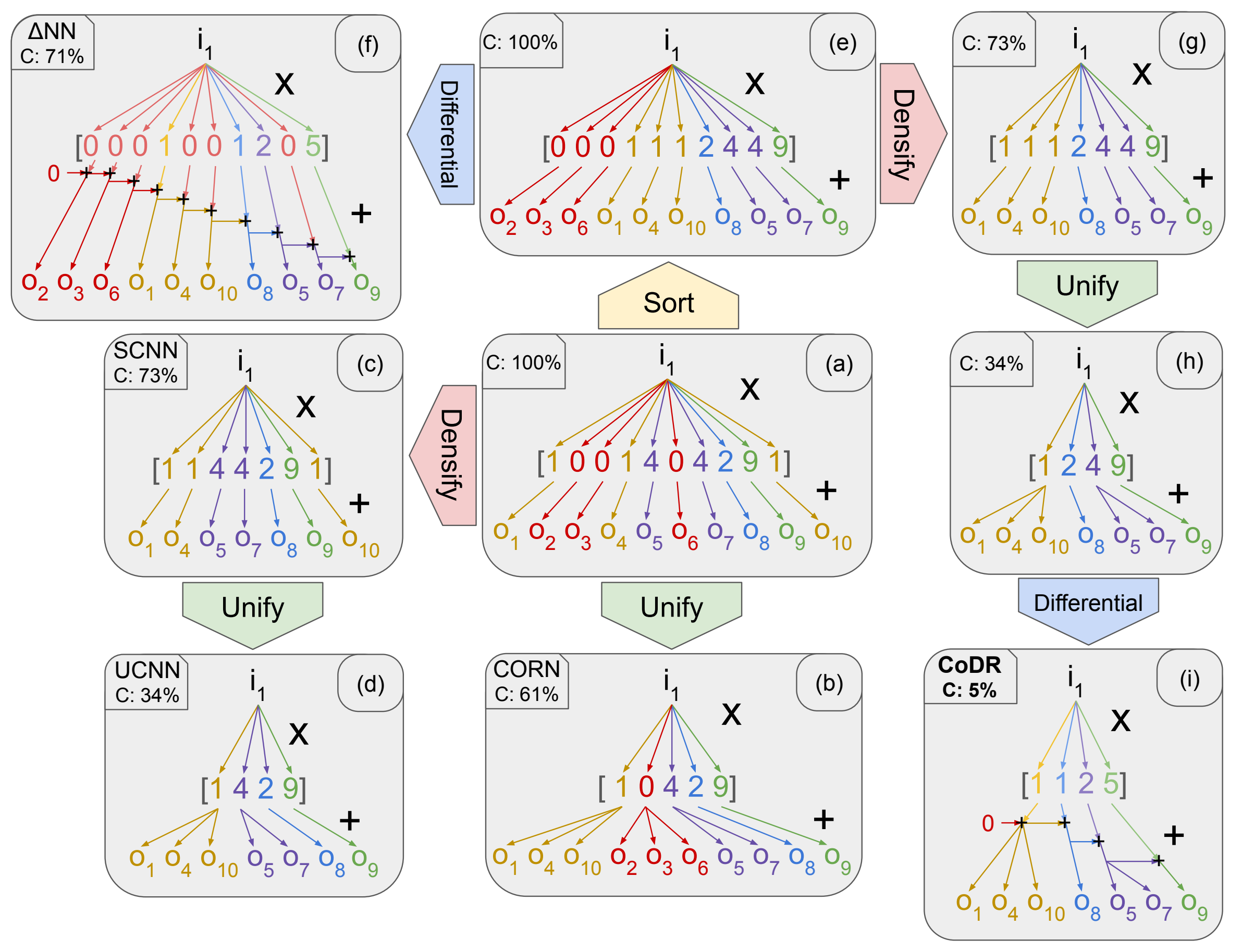

Figure 1: *CoDR eliminates ineffectual (Densify), repetitive (Unify), and similar (Differential) weights.* ***C*** *is the ratio of computation after applying each operation to the GoogleNet model [7].*

2. Computation Reuse

2.1 Weight Sparsity

Figure 1c shows that *SCNN* [1] exploits weight sparsity (94% in the 8-bit VGG16 model [6]) by removing zero weights (red colors).

2.2 Weight Repetition

While DNN inference requires millions of weights, unique weights are bounded by the data bit-length. This results in computation redundancy (39% in the 8-bit GoogleNet [7] weights) which is exploited by *CORN* [5] and *UCNN* [2] using **Unification** (same colors).

2.3 Differential Computation

Zero and redundant weights drop significantly to less than 10.0% in the 16-bit fixed-point weights. As an alternative, **Differential** computation operates on the differences of the weights rather than the absolute operands. ΔNN [4] exploits differential computation.

2.4 Universal Computation Reuse

We introduce *universal computation reuse* that employs three complementary computation reuse techniques, i.e., *Densification*, *Unification*, and *Differential Computation* simultaneously (Figure 1i).

3. Data Reuse

3.1 Scalar-Matrix Multiplication Dataflow

Figure 2 shows that CNN layer inference can be calculated by two operations. (a) **3D convolutions** is conventionally used by CNN accelerator. Instead, (b) *CoDR* employs **Scalar-matrix Multiplication** as it breaks the dependency between the individual weights and enables us to exploit the correlation in the linear weights (c).

3.2 Dataflow Loop Ordering

Since *CoDR* employs novel RLE schemes to compress the weights, accesses to the weights (1.69 bit/weight) are less costly than access to the input or output features (8 bit/feature). *CoDR* reduces the number of on-chip accesses to the input and output features by using an input and output stationary dataflow whose loop ordering is illustrated in Figure 4a. In contrast, *UCNN* [2] and *SCNN* [1] increase the number of costly accesses to the features.

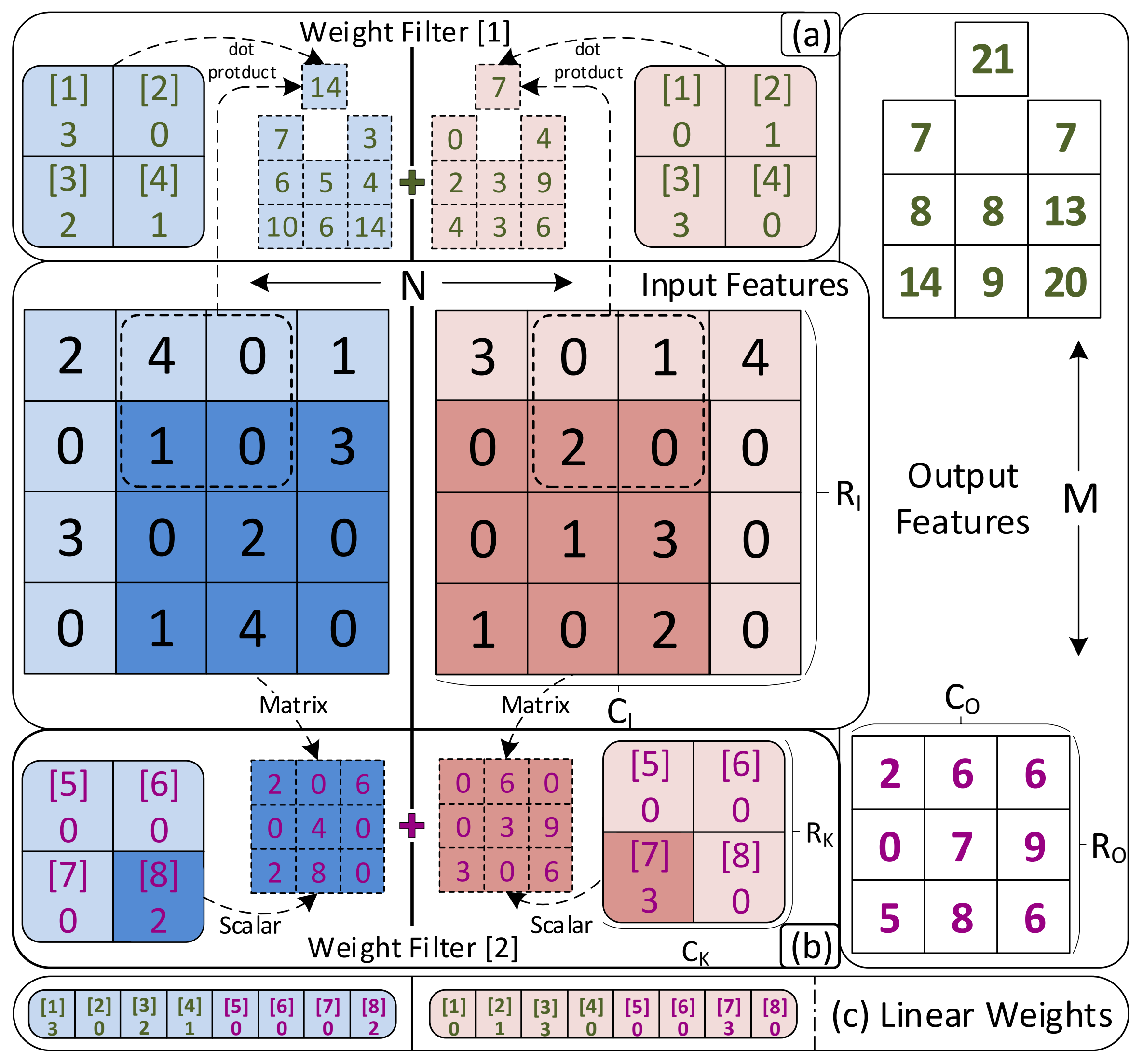

Figure 2: *(a) 3D convolution and (b) scalar-matrix multiplication.*

3.3 Run-Length Encoding

RLE Encoder compresses three types of data: weight Δ values, output indexes, and the count of unique weight repetitions.

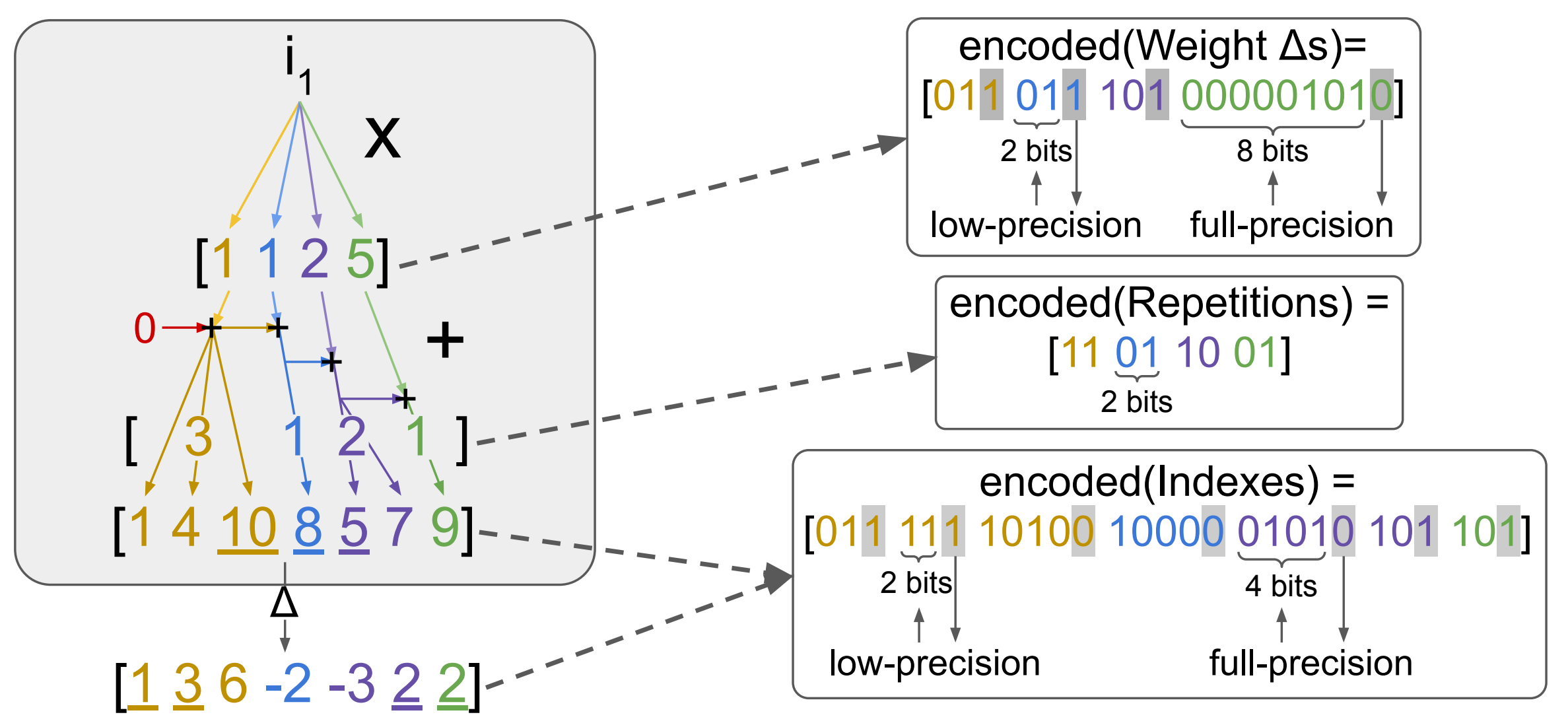

Figure 3: *Customized run-length encoding of the Figure 1i example.*

4. CoDR Architecture

4.1 High-level Architecture

Figure 4a illustrates *CoDR* architecture that contains input, weight, and output SRAM cells, and Processing Units (PUs). Since all PUs work on the same region of the input/output features, an input register file (RF) shared between all PUs caches input features.

4.2 Processing Unit Architecture

A PU (Figure 4b) has Multiplier and Accumulator Processing Elements (MPE and APE) for scalar-matrix multiplication and accumulation (Figure 4c), and an interconnection network to connect them.

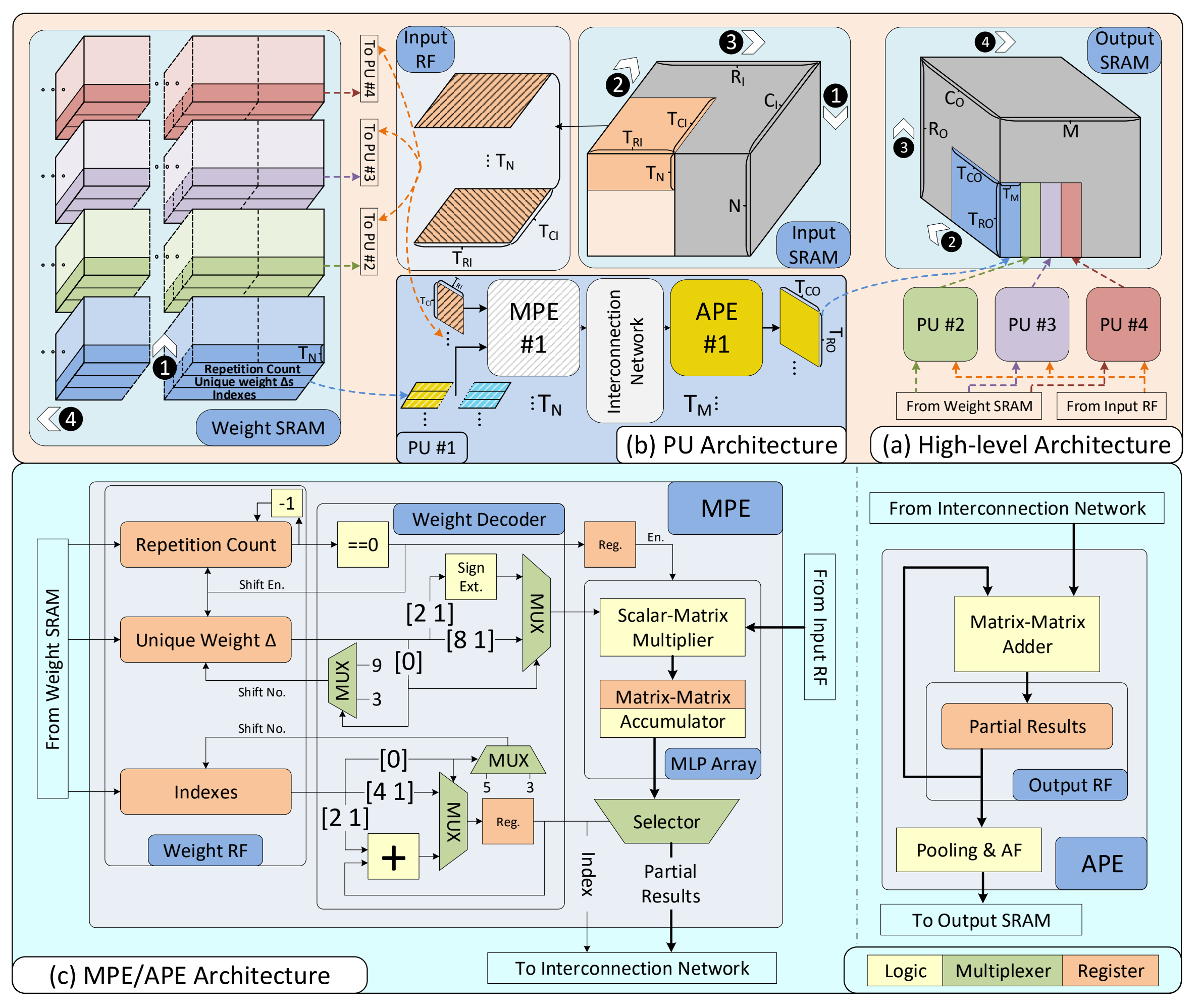

Figure 4: *(a) High-level, (b) PU, (c) MPE and APE architecture.*

5. Evaluation

We compare *CoDR* with two compressed CNN accelerators: *SCNN* [1] and *UCNN* [2]. We evaluate three CNN models [3, 6, 7] with different weight densities (**D**) and number of unique weights (**U**). Customized RLE encoder compresses the weights by $1.69\times$ and $2.80\times$ more than *UCNN* [2] and *SCNN* [1]. Besides, *CoDR* dataflow reduces SRAM accesses by $5.08\times$ and $7.99\times$ by decreasing the number of costly accesses to the input and output. Finally, Figure 5 shows that *CoDR* consumes on average $3.76\times$ and $6.84\times$ less energy relative to *UCNN* [2] and *SCNN* [1].

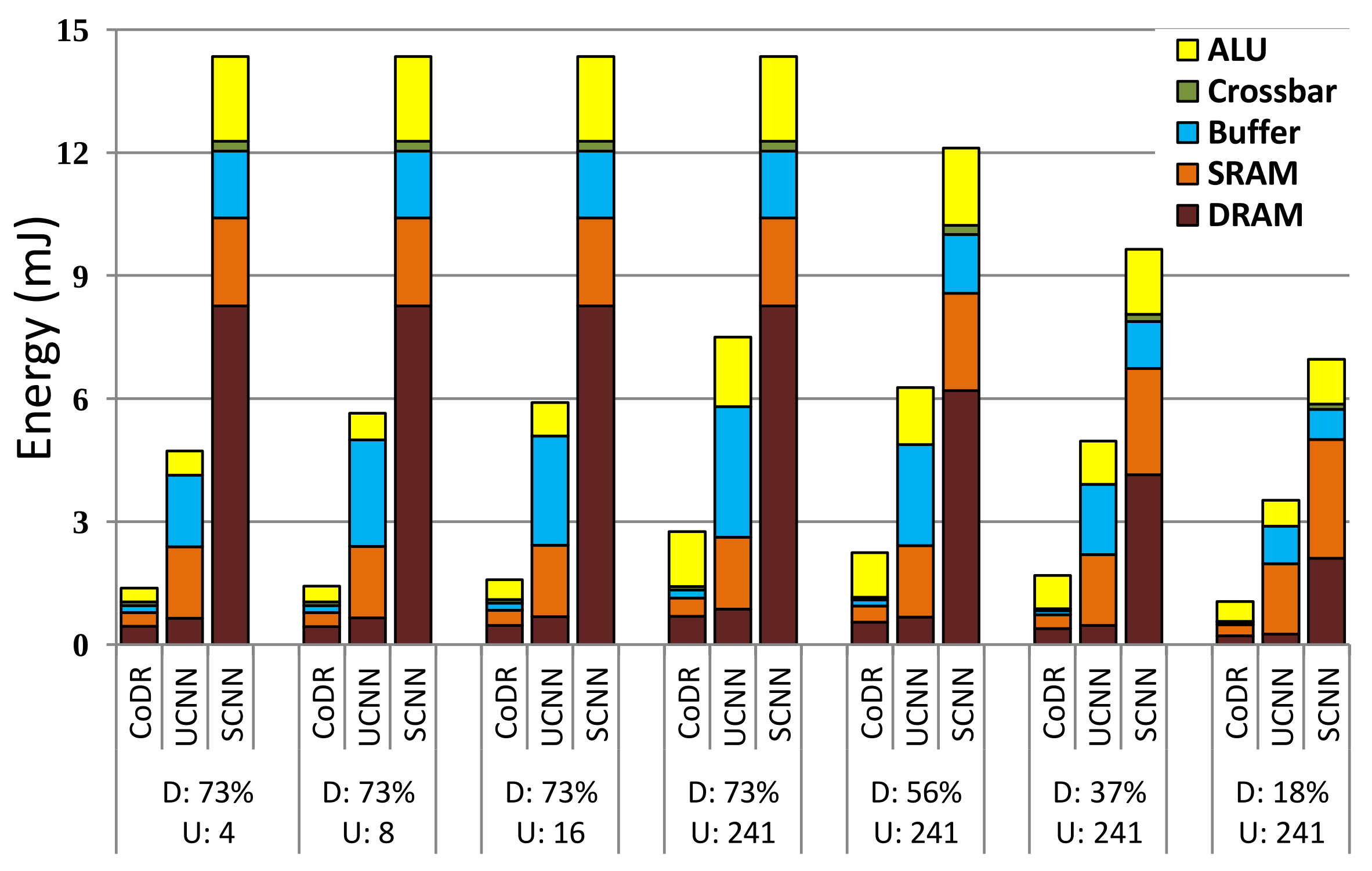

Figure 5: *Energy consumption analysis across different weight densities and number of unique weight of the GoogleNet model [7].*

6. Conclusion

In this work, we study three complementary computation reuse optimizations for the CNN accelerators and introduce *Universal Computation Reuse* that exploits weight sparsity, repetition, and similarity simultaneously. We propose a dataflow that employs scalar-matrix multiplication to apply *Universal Computation Reuse* to the convolutional layers. *CoDR* dataflow makes use of data reuse to minimize the on-chip memory access. We reduce the cost of each weight memory access by customizing run-length encoding based on the weight values. The loop ordering of the *CoDR* dataflow also reduces the total number of accesses to the input and output features by keeping them stationary in the processing elements. Our evaluation over three CNNs with different weight densities and number of unique weights shows that compared to two recent compressed CNN accelerators with the equivalent area of 2.85 mm^2, *CoDR* requires $1.69\times$ and $2.80\times$ less DRAM access, reduces SRAM access by $5.08\times$ and $7.99\times$, and consumes $3.76\times$ and $6.84\times$ less energy.

References

[1] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. Enright Jerger, and A. Moshovo, "Scnn: An accelerator for compressed-sparse convolutional neural networks," *ACM SIGARCH Computer Architecture News*, 2017.

[2] K. Hegde, J. Yu, R. Agrawal, M. Yan, M. Pellauer, and C. Fletcher, "Ucnn: Exploiting computational reuse in deep neural networks via weight repetition," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012.

[4] H. Mahdiani, A. Khadem, A. Ghanbari, M. Modarressi, F. Fattahi-Bayat, and M. Daneshtalab, "ΔNN: Power-efficient neural network acceleration using differential weights," *IEEE Micro*, 2019.

[5] H. Mahdiani, A. Khadem, A. Yasoubi, A. Ghanbari, M. Modarressi, and M. Daneshtalab, "Computation reuse-aware accelerator for neural networks," *Hardware Architectures for Deep Learning*, 2020.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint*, 2014.

[7] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.